ESCUELA POLITÉCNICA SUPERIOR

Departamento de Ingeniería Industrial II

ESTUDIO DE LA CO-EVOLUCIÓN ASÍNCRONA SITUADA PARA LA RESOLUCIÓN DE PROBLEMAS DINÁMICOS DESCENTRALIZADOS EN INGENIERÍA

AUTOR
ABRAHAM PRIETO GARCÍA

DIRECTORES
FRANCISCO J. BELLAS BOUZA
RICHARD J. DURO

FECHA
MAYO, 2009

Francisco J. Bellas Bouza, Profesor Contratado Doctor del Departamento de Computación de la Universidade da Coruña,

Richard J. Duro Fernández, Profesor Titular de Universidad del Departamento de Computación de la Universidade da Coruña,

CERTIFICAN:

Que la memoria titulada 'Estudio de la Co-evolución Asíncrona Situada para la Resolución de Problemas Dinámicos Descentralizados en Ingeniería' ha sido realizada por D. Abraham Prieto García bajo nuestra dirección en el Departamento de Ingeniería Industrial II de la Universidade da Coruña, y constituye la Tesis que presenta para optar al grado de Doctor.

Fdo. Francisco J. Bellas Bouza
Codirector de la Tesis Doctoral

Fdo. Richard J. Duro Fernández
Codirector de la Tesis Doctoral

A todos los que me habéis apoyado,
por haber hecho de esta tesis
una tarea distribuida.

Agradecimientos

A Blanca, por tanto tiempo de trabajo y espera compartido, por su indispensable ayuda y apoyo cuando más lo necesitaba. . . y por la motivación de los momentos que están por venir.

A Fran, por haber despertado mi interés en Inteligencia Artificial desde el primer día, por sus consejos, su paciencia y tolerancia con mi modo de trabajo y sobre todo por su preocupación y atención tanto a nivel personal como profesional.

A Richard, por la confianza depositada en mí y en mi trabajo durante estos años, por lo que he aprendido de su experiencia y de sus contribuciones, por el apoyo recibido en estos últimos meses para cumplir los plazos marcados y finalmente, por haberme brindado la oportunidad de divertirme trabajando.

A Pilar, por sus muchas aportaciones, compañía y ánimo durante el desarrollo de esta tesis, por ayudarme a combatir mi desorganización, por todas esas horas de encerado, y por las que aún quedan.

A mis compañeros del GII. A Vicente por haber dado el oportuno disparo de salida para la escritura de esta tesis, a Juan Monroy y a Alex por su ayuda en el proceso de diseño e implementación del WaspBed, y a todos por sus continuos ánimos.

Quiero agradecer también a Jose Santos las provechosas clases de su curso de doctorado sobre Vida Artificial para mi posterior trabajo y a Martijn Schut y demás miembros de la Artificial Intelligence Section de la VU University de Amsterdam el haberme permitido realizar una estancia en su grupo.

Por último y nunca menos importante, a mis padres y a mi hermano, por el valor que tiene saber que siempre están ahí.

Publicaciones

Durante la realización de esta tesis se han desarrollado los siguientes trabajos:

- Prieto, A., Caamaño, P., Bellas, F., Duro, R. J., (2009). Population Dynamics Analysis in an Agent-based Artificial Life System for Engineering Optimization Problems. *Proceedings CEC 2009*.
- Schut, M., Haasdijk, E., Prieto, A., (2009). Is Situated Evolution an Alternative for Classical Evolution?. *Proceedings CEC 2009*.
- Prieto, A., Bellas, F., Duro, R. J., López-Peña, F., (2008). An adaptive approach for the progressive integration of spatial and spectral features when training ground based hyperspectral imaging classifiers. *Aceptado en IEEE Transactions on Instrumentation and Measurement*.
- Prieto, A., Bellas, F., Caamaño, P., Duro, R. J., (2008). A Complex Systems Based Tool for Collective Robot Behavior Emergence and Analysis. *Lecture Notes in Artificial Intelligence*, **5271**, 633-640.
- Prieto, A., Bellas, F., López-Peña, F., Duro, R. J., (2008). Automatic Preprocessing and Classification System for High Resolution Ultra and Hyperspectral Images. *Computational Intelligence for Remote Sensing*, 313-340.
- Prieto, A., Bellas, F., Duro, R. J., López-Peña, F., (2007). Auto adjustable ANN-based classification system for optimal high dimensional data analysis. *Lecture Notes in Computer Science*, **4507**, 588-596.

- Caamaño, P., Prieto, A., Becerra, J. A., Duro, R. J., Bellas, F., (2007). Evolutionary Tool for the Incremental Design of Controllers for Collective Behaviors. *Lecture Notes in Computer Science*, **4527**, 587-596.

- Caamaño, P., Prieto, A., Bellas, F., Becerra, J. A., Duro, R. J., (2007). Una Herramienta Evolutiva para la Creación de Comportamientos Colaborativos. *Actas Congreso Español sobre metaheurísticas, algoritmos evolutivos y bioinspirados*.

- Prieto, A., Bellas, F., López-Peña, F., Duro, R. J., (2006). Integration of Spatial Information in Hyperspectral Imaging for Real Time Quality Control in an Andalusite Processing Line. *Lecture Notes in Computer Science*, **4253**, 292-299.

- Bellas, F., Faiña, A., Prieto A., Duro, R. J., (2006). Adaptive Learning Application of the MDB Evolutionary Cognitive Architecture in Physical Agents. *Lecture notes on artificial intelligence*, **4095**, 434-445.

- Prieto, A., Bellas, F., Duro, R. J., López-Peña, F., (2006). An Adaptive Visual Gesture Based Interface for Human Machine Interaction in Intelligent Workspaces. *Proceedings of 2006 IEEE International Conference on Virtual Environments, Human-Computer Interfaces and Measurement Systems*.

- Prieto, A., Bellas, F., Duro, R. J., López-Peña, F., (2006). Adaptive Spatio-Spectral Hyperspectral Image Processing for Online Industrial Classification of Inhomogeneous Materials. *Proceedings of 2006 IEEE International Conference on Computational Intelligence for Measurement Systems and Applications*.

- Prieto, A., Bellas, F., Duro, R. J., López-Peña, F., (2005). A Comparison of Gaussian Based ANNs for the Classification of Multidimensional Hyperspectral Signals. *Lecture Notes in Computer Science*, **3512**, 829-836.

Índice

Capítulo 1

Introducción

Esta tesis doctoral propone una técnica de computación evolutiva denominada Co-evolución Asíncrona Situada (CeAS) y estudia su aplicación en la resolución de un tipo particular de problemas de optimización dentro de la ingeniería caracterizados, fundamentalmente, por ser dinámicos y descentralizados. Esta técnica surge como combinación de las técnicas que propone el campo de los Sistemas Complejos para la generación y análisis de sistemas dinámicos y de los procedimientos que estudia el campo de la Inteligencia Artificial Distribuida para coordinar un grupo de agentes hacia un objetivo común.

Como es bien sabido, las técnicas de computación evolutiva tienen su base en las sencillas ideas de Darwin sobre selección natural en las especies, y originalmente fueron planteadas como técnicas avanzadas de búsqueda en espacios n-dimensionales que emplean una población de potenciales soluciones y explotan las altas capacidades de cómputo permitidas por los computadores. Su utilización en tareas reales no es nueva, y ya desde principios de los años 60 se empezaron a utilizar dichas técnicas para realizar tareas de optimización y aprendizaje automático (Box y Chanmugam, 1962; Friedman, 1959; Holland, 1962). Con el paso de los años se fueron desarrollando en profundidad y se demostró su capacidad de solucionar problemas con alto nivel de ruido y discontinuidades (De˜Jong, 1975). Fue a mediados de los 80 cuando su aplicación ya se había extendido a diversas áreas, desde problemas matemáticos

abstractos hasta problemas reales como el control de flujo en una línea de producción, análisis de patrones y otras muchas (Goldberg, 1989).

Hoy en día, la computación evolutiva es un campo ya maduro a nivel científico y está resolviendo problemas 'de interés cotidiano' en áreas de estudio tan diversas como la predicción en el mercado de valores y la planificación bursátil, ingeniería industrial, diseño de componentes electrónicos, programación de horarios en aeropuertos o líneas de montaje. Y esto es así porque, aunque el campo tenga ya más de medio siglo de vida, el incremento de las capacidades de cálculo de los ordenadores ha llevado parejo el incremento de la potencia de este tipo de algoritmos.

Desde el punto de vista de la optimización, una de las principales ventajas de las técnicas de la computación evolutiva es que no presentan unos requerimientos matemáticos muy formales con respecto al problema que están tratando, como por ejemplo, el cálculo de derivadas (o incrementos) de ningún orden tan común en las técnicas de descenso de gradiente. Las técnicas evolutivas son métodos de orden cero, es decir, todo lo que necesitan es la evaluación de la función objetivo. Sin embargo, pueden utilizarse para resolver problemas no lineales definidos sobre espacios de búsqueda discretos, continuos o mixtos, con o sin restricciones.

Muchas de las actividades propias de las distintas ramas de la ingeniería integran problemas no estructurados de la vida real que son complicados de modelar, debido a que requieren la incorporación de factores inusuales (desde variables estocásticas, factores de riesgo de accidentes, variables de mercado, hasta otros completamente subjetivos, estéticos por ejemplo). Además, gran parte de ellos son ejemplos típicos de problemas NP-completos, tales como la programación de tareas, horarios, TSP o la ubicación de instalaciones.

En estos casos, la computación evolutiva se presenta como una opción muy prometedora y con muchas ventajas para obtener soluciones a este tipo de problemas. El carácter poblacional de estas herramientas hace que presente la capacidad de proporcionar muchas soluciones cercanas al óptimo al final de cada ejecución. Esto es de interés si, por ejemplo, el criterio de selección no está rigurosamente definido o si presenta algún

tipo de ambigüedad, ya que permite elegir la mejor solución a posteriori, tras haber realizado ya la evolución. Por otro lado, los algoritmos evolutivos pueden llegar a un nivel de eficiencia realmente competitivo debido a su relativa capacidad de hibridarse con heurísticas dependientes del dominio.

Todas estas características de la computación evolutiva ya se han dado a conocer en el campo de la ingeniería industrial y son conocidas varias aplicaciones exitosas (Goldberg, 1989; Powell y Skolnick, 1993; Surry *y otros*, 1995). Sin embargo, tras un análisis actualizado de los problemas de optimización que aparecen en casos reales, circunscritos a diversas ramas de la ingeniería, nos hemos encontrado con que hay una serie de características, a mayores de las que ya hemos comentado, que son comunes a muchos de ellos y que los hacen especialmente complejos para las técnicas existentes actualmente, incluso para las evolutivas tradicionales. El objetivo de esta tesis será desarrollar y estudiar la aplicación de una nueva técnica evolutiva que permita tratar este tipo de problemas.

Las características que definen los problemas tipo a los que nos enfrentaremos son las siguientes:

- **Dinamismo:** esta característica es típica de los problemas reales que dependen del tiempo, ya que en la mayoría de ellos se producen diversos cambios en tiempo real, que pueden afectar a las variables del sistema, a la función objetivo, a las restricciones, etc. Algunos de estos cambios son intrínsecos al problema y otros se derivan de fallos de funcionamiento.

- **Descentralización:** existe un gran número de problemas que, o bien requieren que para su resolución la tarea se lleve a cabo de forma distribuida, o bien este enfoque proporciona una mejora significativa en la utilidad de la solución. Dentro del tratamiento de los problemas que requieren una distribución de la tarea, la resolución de esta puede ser descentralizada o centralizada. En el marco de esta tesis, nos centraremos únicamente en problemas que requieran una resolución descentralizada, característica que engloba a dos típicamente relacionadas:

- **Conocimiento parcial:** en numerosos ejemplos de problemas de ingeniería se posee únicamente conocimiento parcial de las variables en juego, de forma que la optimización del objetivo no se puede asegurar mediante una resolución centralizada.
- **Interacciones locales:** por otro lado, también es común que la función objetivo que se pretende optimizar resulte de la combinación no trivial de interacciones locales, lo que implica que, de nuevo, un enfoque centralizado no sea posible.

Hablaremos pues de la resolución, no de cualquier tipo de problema de optimización en ingeniería, sino de aquellos que sean **dinámicos** y **descentralizados**. Como veremos en el apartado dedicados a antecedentes, tanto el dinamismo como la descentralización son características que se suelen 'relajar' en la definición de los problemas, dada su alta complejidad. En el marco de esta tesis, estas dos características serán imprescindibles y no pueden ser eliminadas.

La técnica de optimización que resuelva de forma distribuida problemas dinámicos descentralizados, deberá proporcionar:

- **Escalabilidad:** definimos la escalabilidad como la capacidad para una técnica de resolución de un problema de proporcionar soluciones válidas cuando se cambian algunas de las dimensiones del problema. El frecuente cambio en los recursos, necesidades, demandas, etcétera en los problemas reales, hace que muchos problemas requieran de una aproximación escalable para proporcionar una solución satisfactoria.
- **Robustez:** se define como la capacidad de un sistema para adaptarse a fallos, o ruido en partes del sistema, sin pérdida de funcionalidad. Ruido, fallos en componentes y demás problemas no previstos son inherentes a los problemas reales.
- **Adaptabilidad:** es la capacidad de un elemento para adaptarse a cambios en el entorno que le rodea. En nuestro caso, en cierto modo engloba a las dos características anteriores de un modo más general y representa la capacidad de una herramienta para trabajar con problemas cambiantes, obteniendo soluciones para nuevas situaciones a partir de las soluciones anteriores.

En cuanto a los tipos de problemas que son susceptibles de ser tratados con la técnica que se propone en esta tesis, a partir del estudio de la literatura relacionada hemos destacado las siguientes categorías básicas, aunque no son las únicas posibles, que nos permitirán definir más adelante los problemas que consideramos representativos para comprobar el funcionamiento de la técnica:

- **Problemas de abastecimiento, VRP o variantes del mismo:** este problema consiste en líneas generales en obtener un conjunto de rutas que permitan que un grupo de vehículos abastezcan a un grupo de clientes. Adicionalmente se define una base de suministro y una matriz de distancias/coste/tiempo entre clientes y clientes/base. El objetivo depende de las especificaciones de cada problema pero generalmente será, o bien minimizar el coste del reparto o minimizar los tiempos de espera o alguna función de utilidad definida sobre estas dos. Existen múltiples variantes del mismo añadiendo limitaciones o restricciones adicionales; ventanas de tiempo, capacidad de los vehículos, etc. Este problema ha sido ampliamente tratado mediante el uso de heurísticas y metaheurísticas en su definición clásica. Sin embargo, en sus aplicaciones reales este tipo de problemas tienen mucha tendencia a variar sus restricciones o parámetros mientras se están llevando a cabo los repartos, fallos en vehículos, corte de rutas, etcétera, adicionalmente la generación de demandas es altamente fluctuante y en ningún caso un valor fijo y por tanto se requiere un solucionador robusto.
- **Problemas de exploración, mapeado o de vigilancia:** estos problemas se definen mediante un entrono que posee serie de puntos o áreas que deben ser visitados por un observador, bien para mapear el entorno o bien para detectar algún tipo de elemento o irregularidad. El problema de vigilancia es un caso particular de esta tipología en el que se realiza una búsqueda continua de irregularidades. La función objetivo está basada en el tiempo que necesita el algoritmo para recorrer todos los puntos de control (o superficie de control en el caso continuo). Este tipo de problemas tienen gran cantidad de aplicaciones reales, tales como la búsqueda de náufragos, la detección de focos de fuego en una zona forestal o la vigilancia de recintos de seguridad. Variaciones en zonas de riesgo, numero de observadores, fallos en observadores, fallos en sensores, etcétera.

- **Problemas de optimización con grafos:** existen múltiples problemas de optimización combinatoria que han sido modelados mediante grafos, el problema del viajante, el problema de la coloración, el problema del camino más corto, etc, etcétera. La mayoría de los problemas reales son modelables mediante el uso de grafos, por tanto estos problemas inicialmente teóricos presentan una gran cantidad de aplicaciones reales, obtención de rutas óptimas, algoritmos de reparto, problemas de planificación, etcétera
- **Problemas de ubicación de instalaciones:** dado un conjunto de instalaciones, un conjunto de posibles emplazamientos y un conjunto de clientes que deben ser servidos definimos varios objetivos: ¿Qué instalaciones sirven a cada cliente para minimizar los costes del transporte? ¿Dónde situamos cada una de las instalaciones para minimizar los costes de transporte?

Todos estos tipos de problemas están sujetos a cambios en tiempo real, todos son problemas de optimización combinatoria y su resolución en forma distribuida aporta ventajas intrínsecas en cuanto al cumplimiento de los requisitos de robustez, adaptabilidad y escalabilidad si estas soluciones se basan en interacciones locales, y por tanto, se resuelven de forma descentralizada. Esta es la aproximación que se plantea en esta tesis.

La técnica que se desarrolla y analiza a lo largo de esta memoria se ha denominado CeAS: Co-evolución Asíncrona Situada. Esta denominación resume sus tres características fundamentales a nivel algorítmico. Por un lado, como hemos adelantado, el CeAS se inspira en los estudios de Sistemas Complejos en cuanto al uso de un sistema dinámico con interacciones locales entre sus constituyentes. De un modo más concreto, la Vida Artificial, cuyo estudio se enmarca dentro de los Sistemas Complejos, se basa en la simulación de entornos en los que los individuos están sujetos a un proceso de coevolución descentralizada y asíncrona.

Así, el CeAS pertenece a la categoría de algoritmos de *coevolución* debido a que permite una competencia entre individuos de una población para guiar el proceso de búsqueda de soluciones. Adicionalmente, la coevolución implica que el proceso evolutivo es abierto (del inglés open-ended), con lo cual no existe un criterio de parada basado en un nivel de

utilidad absoluto, sino que se utiliza un número de iteraciones fijo o un criterio basado en la estabilidad evolutiva de la población.

Por otro lado, a diferencia de muchas de las aproximaciones tradicionales bioinspiradas como es el caso de los algoritmos genéticos, estrategias evolutivas, etc., en los que la selección de los individuos para la reproducción se lleva a cabo mediante una evaluación y comparación centralizada en base a una función objetivo y cada cierto número de pasos de simulación preestablecido, en el CeAS la selección siempre se realizará de forma local y supeditada a determinadas condiciones que irán asociadas a la eficiencia en la tarea individual y que se generarán, por tanto, de forma *asíncrona*. En consecuencia, en el CeAS la evolución es *situada*, es decir, todas las interacciones entre los individuos de la población son locales y están supeditadas a una serie de relaciones de proximidad o coincidencia espacio-temporal, lo cual implica una descentralización intrínseca.

Sin embargo, y a pesar de las coincidencias expuestas, la principal diferencia entre el CeAS y los algoritmos evolutivos open-ended existentes en el campo de Vida Artificial, estriba en que estos no buscan la optimización de una función objetivo, sino el estudio del sistema resultante una vez que se alcanza el equilibrio. En el caso de una ejecución mediante el uso del CeAS, esta no se reduce a la simulación de un sistema dinámico complejo sino que se orienta la evolución hacia la búsqueda de un objetivo concreto. Esto implica realizar una serie de modificaciones en el procedimiento de evolución que permitan introducir estos objetivos, y para esto nos vamos a servir de las técnicas que se han venido estudiando en el campo de la Inteligencia Artificial Distribuida. Más concretamente, estudiaremos el establecimiento de unas funciones de utilidad privada y global que guíen la evolución para distribuir el comportamiento objetivo de la población en una serie de comportamientos individuales.

El trabajo llevado a cabo en esta tesis ha seguido los siguientes pasos metodológicos:

En primer lugar se ha fijado con claridad el objetivo final de la presente tesis doctoral y los subobjetivos que han llevado a su consecución. Estos objetivos aparecen en el capítulo 2 de la presente memoria.

A continuación se realiza una revisión general de los problemas de optimización en ingeniería y de las principales técnicas de resolución que se han venido aplicando. A partir de este estudio se establecen una serie de características reales que presentan grandes dificultades para estas técnicas y que, tal y como comentamos anteriormente, definen un tipo de problemas dinámicos descentralizados que serán el objetivo de resolución del CeAS. Como conclusión de este apartado, queda patente la necesidad de aportar un nuevo enfoque en forma de técnica computacional para resolver estos problemas. La primera parte del capítulo 3 de esta memoria, titulado *Antecedentes*, se dedicará a este estudio.

Posteriormente, se analizan las principales aportaciones a la resolución de problemas intrínsecamente distribuidos en dos campos: el de la Inteligencia Artificial Distribuida (o Sistemas Multiagente) y el de los Sistemas Complejos. De ambos análisis se concluyen una serie de ideas de partida para el planteamiento de esta nueva técnica. Así, se extraen, del campo de los Sistemas Complejos, la generación, control y análisis de simulaciones con múltiples elementos e interacciones, y del campo de la Inteligencia Artificial Distribuida, las ideas para la obtención de un comportamiento coordinado que resuelva de forma distribuida la tarea que establece el diseñador. Esta revisión se lleva a cabo en la segunda parte del capítulo de *Antecedentes*, alcanzándose unas especificaciones básicas de los requerimientos para la técnica que se aborda.

En el capítulo 4 se lleva a cabo el diseño básico del CeAS, se genera también una metodología para la transformación de un problema real en un escenario evolucionable mediante el CeAS y para el establecimiento de los mecanismos que nos servirán para guiar la evolución hacia el objetivo global predefinido. Finalmente, se establecen también unas pautas de análisis de la técnica durante el transcurso de la evolución.

A partir de las características del CeAS, y del análisis de las opciones existentes en la actualidad, se concluye la necesidad de desarrollar una herramienta de simulación computacional propia. Esta herramienta se denomina WaspBed (World-Agent Simulation Platform for BEhavior Design) y se presenta como un simulador sencillo, flexible y de fácil uso, que permite modelar Sistemas Multiagente complejos en el marco de la evolución asíncrona situada de forma rápida, y analizar la dinámica de este tipo de

sistemas con gran número de elementos e interacciones. WaspBed ha servido también para diseñar una serie de estructuras formales (plantillas de configuración) para la definición y modelización de cualquier escenario en el simulador, lo que permite establecer un procedimiento organizado de planteamiento de un problema particular en CeAS. En el apéndice I se puede encontrar una especificación detallada del WaspBed en cuanto a las clases que lo componen.

Una vez obtenida una versión probada del WaspBed, pasamos a estudiar en detalle el comportamiento del CeAS mediante este simulador. Para ello, primeramente se plantea un problema dinámico descentralizado de obtención de la *ruta óptima en un grafo* de manera descentralizada. En el capítulo 5 se muestran las especificaciones concretas del ejemplo así como los resultados más destacados que se han obtenido. El principal es que el CeAS obtiene una solución satisfactoria, proporcionando un algoritmo de búsqueda muy similar a otros muy comunes en la literatura, pero con la propiedad añadida de la adaptabilidad mostrada por la técnica antes cambios en el sistema, algo que no se puede obtener con la metaheurística original. Una de las consecuencias de este primer estudio es la necesidad de llevar a cabo un análisis energético del sistema, de cara a poder fijar a priori las regiones de funcionamiento estable, estudio que se realiza para este caso analizando sus implicaciones.

El siguiente problema que se plantea para analizar el CeAS es un *problema de exploración*, en concreto de vigilancia, otro de los ejemplos típicos dinámicos descentralizados que hemos encontrado en la bibliografía. El problema se aborda considerando todas las restricciones que implicaría su aplicación en un sistema robótico real de tamaño fijo de manera que se pueda tratar el estudio del CeAS como algoritmo de operación en tiempo real en entornos físicos.

Este ejemplo permite realizar un estudio comparativo de esta técnica frente a otras ya existentes y constatar la mejoría aportada por el CeAS. Además, también se ha comprobado el funcionamiento del CeAS en un problema con tamaño de población fija, lo cual implica que el balance energético no es necesario y tambien que el algoritmo permite obtener soluciones a problemas con restricciones propias de soluciones reales como es la existencia de una cantidad fija de vigilantes. Este análisis se va a

centrar, por tanto, en las funciones de utilidad privada y global que guían la evolución hacia la solución deseada y en el estudio de los parámetros relevantes para el control de la evolución. Los detalles sobre este segundo ejemplo de aplicación se encuentran en el capítulo 6 de la memoria.

Finalmente, se utiliza el CeAS en un problema dinámico descentralizado que aglutina todas las características que pretendemos tratar en este tipo de problemas: el problema de la asignación óptima de *fletes de buques* en paralelo con la definición de la flota ideal. Este problema pertenece a la categoría de problemas de abastecimiento y presenta una aplicación real directa en cuanto a que los criterios de utilidad se basan en parámetros económicos. Se diseña para ello un escenario en el WaspBed que contempla una serie de puntos de suministro y de demanda de mercancía y una flota de buques de tamaño y configuración variable. Los resultados pueden ser consultados en el capítulo 7 de la memoria.

El capítulo 8 de la memoria presenta las principales conclusiones y aportaciones que se pueden extraer de todo el trabajo realizado además de presentar las líneas que han quedado abiertas y que deberán ser tratadas en el futuro.

Cabe destacar que el desarrollo de esta tesis se enmarca en una serie de proyectos del Grupo Integrado de Ingeniería de la Universidade da Coruña relacionados con sistemas distribuidos y cooperativos y financiados tanto por el MEC, el MICINN y la Xunta de Galicia cuya financiación queremos agradecer. Por otra parte, el trabajo realizado se ha llevado a cabo en los laboratorios del Grupo Integrado de Ingeniería en Ferrol y en la Artificial Intelligence Section, Department of Computer Science, VU University of Amsterdam.

Capítulo 2

Objetivos

El principal objetivo de esta tesis doctoral es el siguiente:

Diseñar, implementar y estudiar en detalle las capacidades y limitaciones de una nueva técnica de computación evolutiva que denominaremos Co-evolución Asíncrona Situada (CeAS) y que surge como alternativa a las técnicas de optimización existentes en un dominio de aplicación concreto: los problemas de optimización dinámicos descentralizados que aparecen en los distintos campos de la ingeniería.

De cara a alcanzar el objetivo global, este se desglosa en una serie de sub-objetivos que se exponen a continuación:

- Definir una técnica que permita la resolución de problemas de ingeniería que no son abordables por técnicas tradicionales y que se caracterizan por su alto grado de dinamismo y por implicar la resolución de una tarea de forma distribuida mediante una estructura descentralizada. Se busca también que esta técnica presente las características de adaptabilidad y escalabilidad necesarias para obtener soluciones efectivas a problemas reales.

  El proceso de definición se descompone en los siguientes puntos a realizar:

  - Definición formal del funcionamiento del CeAS: generación de un escenario, estructura secuencial del algoritmo, procesos de

evolución y/o adaptación, criterios de parada y cualquier otra consideración necesaria para la descripción precisa del proceso.

- Definición de nuevos operadores de selección y de cruce para la reproducción, teniendo en cuenta el carácter descentralizado y poblacional de las soluciones requeridas.
- Definición del procedimiento de utilización del CeAS: creación de una metodología de diseño para llevar a cabo la transformación de un problema real a un escenario de evolución válido para el CeAS. Esto implica:
  - Establecimiento de requisitos de modelado distribuido.
  - Introducción de las funciones de utilidad global e individual para guiar el proceso de evolución.
  - Gestión energética para el control de la evolución.
  - Definición de los procesos de selección reproductiva y reemplazo de la población.
- Definición del procedimiento de análisis: establecimiento de una estrategia que permita extraer información a partir del comportamiento del escenario.
  - Estudio del proceso evolutivo.
  - Extracción de soluciones.
  - Análisis de robustez/sensibilidad de una configuración.

- Crear un software de apoyo para el uso del CeAS: con el objeto de implementar la estructura planteada por el CeAS, debemos utilizar una herramienta software que nos vaya a permitir las tareas de diseño, ejecución de los procesos de evolución y análisis de un modo cómodo y flexible. Para ello es necesario:
  - Definir las necesidades de un software de apoyo al uso del CeAS: permitir y facilitar la definición de los constituyentes de un escenario, definición de las interacciones, flexibilizar las modificaciones en el modelo inicial, capacidad de reutilización de herramientas de configuración y análisis desarrolladas, etcétera.
  - Implementar el software de apoyo: WaspBed.

  - Comprobar el funcionamiento del software de apoyo mediante la implementación de un escenario de prueba que cubra las funcionalidades requeridas.

- Comprobar qué ventajas e inconvenientes plantea CeAS con respecto a otras aplicaciones y definir así el ámbito de aplicación: la definición de un nuevo concepto de técnica evolutiva ha de presentar una serie de ventajas inherentes a su estructura. Estas capacidades han de comprobarse y debemos estudiar si aportan una mejora respecto a otras aproximaciones. Por otro lado, existirán también una serie de inconvenientes que acoten el alcance del CeAS que debemos detectar y definir.

- Estudiar la aplicabilidad de la técnica CeAS a varios problemas distribuidos y dinámicos representativos dentro de la ingeniería. Estudiar su capacidad frente a las características que definen los problemas reales de ingeniería, según se extrae del estudio de estos problemas:

  - Estudiar la aplicabilidad del CeAS a problemas de optimización combinatoria.
  - Estudiar la aplicabilidad en problemas que presenten restricciones, bien en el modelo del entorno o bien en la definición de los individuos de la población.
  - Estudiar la aplicabilidad en problemas definidos mediante variables estocásticas.
  - Estudiar la aplicabilidad en problemas que requieren de grupos heterogéneos de individuos para llevar a cabo la tarea de forma eficiente.

Capítulo 3

Antecedentes

En este capítulo se lleva a cabo una revisión de los campos de conocimiento en los que se fundamenta esta tesis doctoral, fijando los aspectos más relevantes de los mismos en cuanto a sus principios básicos y en sus principales desarrollos hasta la fecha. El objetivo de esta revisión es, por un lado, establecer claramente la necesidad de la técnica propuesta, y por otro, justificar la idoneidad de las aproximaciones en las que se inspira.

El capítulo se ha estructurado en tres partes fundamentales. La primera, relacionada con las técnicas de optimización en ingeniería, concluirá con una definición clara y concisa del tipo de problema marco al que nos enfrentaremos en el resto de la tesis. La segunda parte revisa la resolución distribuida de problemas dinámicos en el campo de la Inteligencia Artificial Distribuida y en el de los Sistemas Complejos. De este apartado se extraerán una serie de conceptos básicos de cada campo que se integrarán en la técnica evolutiva propuesta. Finalmente, en la tercera parte se lleva a cabo un resumen global de todo el capítulo que dé paso al desarrollo propiamente dicho.

3.1. Problemas de optimización en ingeniería

3.1.1. Introducción

La *optimización matemática* se refiere a un proceso mediante el cual se busca solucionar un problema que podemos definir de forma genérica

como:

Hallar el máximo (o mínimo) de una función

$$f(x), x \in \varphi \in \mathbb{R}^n \tag{3.1}$$

siendo n, el número de variables del problema y φ el subespacio definido por el conjunto de restricciones.

Por otro lado, la Real Academia Española define la *ingeniería* como:

Estudio y aplicación, por especialistas, de las diversas ramas de la tecnología.

Otras definiciones posibles del campo de la ingeniería son:

La ingeniería es la profesión que aplica conocimientos y experiencias para que mediante diseños, modelos y técnicas se resuelvan problemas que afectan a la humanidad.

La ingeniería es el arte de aplicar los conocimientos científicos a la invención, perfeccionamiento o utilización de la técnica en todas sus determinaciones.

De las anteriores definiciones se puede derivar un concepto más concreto de la ingeniería como aquel campo en el que el conocimiento de las matemáticas y de las ciencias naturales, obtenido mediante estudio, experiencia y práctica, se aplica para desarrollar estrategias óptimas de utilizar los recursos disponibles en la naturaleza para beneficio de la humanidad.

Podemos entonces definir los problemas de optimización en ingeniería como 'la búsqueda de la mejor solución para un problema técnico definido en el mundo natural'. El término *mejor* en esta definición es dependiente del contexto en el que se enmarque dicho problema y de los objetivos establecidos para cada caso particular. Esto podría implicar tanto la búsqueda de una solución que minimiza variables tales como los costes o los riesgos, la resistencia al avance, etc., como la de otra que maximiza los beneficios, la eficiencia, etc. De cualquier forma, la característica fun-

damental del tipo de problemas de optimización que se trata en el marco de esta tesis es que provienen del mundo real, no siendo el objetivo de este trabajo el tratar con funciones de optimización sintéticas. Estos problemas se caracterizan por su alta complejidad al depender de variables estocásticas, parcialmente conocidas, continuas, etc., que corresponden a funciones dinámicas, discontinuas, multimodales, etc. Es decir, resultan un 'banco de pruebas' ideal para el desarrollo y la validación de nuevas técnicas de optimización, como es el caso que nos ocupa.

A nivel práctico, resolver un problema de optimización en una aplicación de ingeniería requiere de dos etapas fundamentales: primero, determinar el modelo matemático que rige el problema, y segundo, realizar el proceso de optimización de una variable objetivo utilizando algún tipo de algoritmo. En el marco de esta tesis, vamos a trabajar sobre problemas con modelos ya conocidos, y por tanto, nos centraremos en la segunda etapa.

Basándonos en los manuales clásicos de optimización en ingeniería, encontramos la siguiente división en categorías (Nocedal y Wright, 1999; Rao, 1996):

- *Minimización uni/multidimensional lineal sin restricciones:* este tipo de problemas pueden solucionarse fácilmente mediante técnicas clásicas de descenso de gradiente o incluso mediante una solución analítica, que en estos casos, no suele ser difícil de obtener.

- *Programación lineal con restricciones:* muchos de los problemas reales de optimización pueden ser expresados como problemas de programación lineal, y es debido a esto que se ha generado abundante investigación en algoritmos para su solución.

  Adicionalmente, muchos algoritmos para la resolución de problemas no lineales funcionan resolviendo sub-problemas acotados del problema principal planteados como problemas lineales. Por otro lado, muchas ideas de la programación lineal han inspirado conceptos para la teoría de optimización en funciones no lineales y con restricciones, como es el caso de la dualidad, descomposición, convexidad,

etc. El método más conocido para resolver este tipo de problemas es el método simplex.

- *Minimización de funciones no lineales con restricciones:* es en esta categoría, que comprende a las dos anteriores, donde podríamos encuadrar a la gran mayoría de los problemas de basados en modelos reales. Las ecuaciones que rigen los procesos reales, relaciones entre variables y las funciones objetivo, presentan no linealidades si se modelan con cierta fidelidad. Por otro lado, los problemas reales suelen presentar un gran número de restricciones asociadas a: limitaciones en el rango de la mayoría de las variables (espacio, velocidad, tiempos de procesado, entregas, etc.), no homogeneidad del entorno (obstáculos, discontinuidades, etc.) o restricciones asociadas a combinaciones de variables (distancias, limitaciones de recursos, etc.). Estas restricciones deben ser consideradas a la hora de modelar este tipo de problemas, pues en muchos casos modifican completamente el tipo de soluciones factibles así como los procesos de búsqueda de las mismas.

- *Programación entera y mixta:* otra categoría es aquella en la que todas o parte de las variables que entran en juego en el proceso de optimización son enteras. Este tipo de problemas puede llegar a modelarse mediante funciones no lineales con restricciones, pero existen métodos de resolución específicos y constituyen un caso lo suficientemente importante como para ser diferenciados.

- *Optimización multiobjetivo:* es un tipo optimización en el cual no existe una única función objetivo que combine todos los sub-objetivos de una aplicación, y por tanto, se busca obtener lo que se denomina un frente de soluciones no dominadas en lugar de un óptimo global.

- *Optimización basada en elementos finitos:* es aquella optimización que no se basa en un modelo matemático, sino en los resultados obtenidos resolviendo un modelo físico mediante la técnica de elementos finitos. Se utiliza principalmente en problemas de diseño de estructuras o de piezas para minimizar los esfuerzos tensionales, gradientes de temperatura, para mejorar resistencia, conductividad, aislamiento, etc.

Debemos mencionar que, además de esta clasificación tradicional que ha sido ampliamente utilizada, podemos encontrar otras que utilizan criterios diferentes. Por ejemplo, He y Wang (2007) propone una división de los problemas de optimización en sólo cuatro tipos:

1. Optimización global
2. Multiobjetivo
3. Multidisciplinares
4. Probabilísticos

La novedad principal de esta taxonomía reside en la utilización de dos nuevos grandes grupos de problemas. Por una parte, la optimización multidisciplinar es aquella en la que intervienen restricciones o condiciones en el modelo provenientes de diferentes disciplinas científicas para llevar a cabo un modelado más real de un problema (variables económicas, variables físicas, variables sujetas a criterio subjetivo, variables de tendencias de mercado, etc.).

En cuanto a la optimización probabilística, es aquella en la que algunas de sus variables no poseen valores fijos sino que responden a algún tipo de distribución de probabilidad. Esta última distinción nos parece realmente importante y a tener en cuenta en los procesos de optimización de problemas reales. Existen muchos modelos que suponen una serie de valores fijos para algunas variables, cuando estas nunca muestran valores constantes, y el hecho de introducir variables estocásticas cambia sustancialmente el problema en cuanto a inestabilidad, índices de riesgo, etc.

Tras haber introducido el tipo general de problemas que se consideran como optimización en ingeniería, a lo largo del siguiente apartado trataremos de acotar el tipo de problemas de ingeniería que se tratarán en esta tesis, centrándonos en aquellos que se engloban dentro del ámbito de la Investigación Operativa.

3.1.2. Investigación operativa

Relacionada con la optimización en ingeniería existe una rama interdisciplinar de las matemáticas aplicadas y de la ciencia formal que se deno-

mina *Investigación Operativa*. Esta utiliza métodos matemáticos, estudios estadísticos y aproximaciones con heurísticas para obtener soluciones a problemas complejos en sistemas organizados. Su objetivo principal es el de la optimización en problemas reales, es decir, maximizar o minimizar una función objetivo con referente real (maximizar beneficios, eficiencia de una línea de montaje, ancho de banda o minimizar riesgos, pérdidas, etc.).

La Investigación Operativa se relaciona de forma muy cercana a la ingeniería industrial, y en este campo se considera como una parte importante de sus disciplinas. El objetivo, en último término, de la aplicación de la Investigación Operativa es apoyar en la 'toma óptima de decisiones' en los sistemas y en la planificación de sus actividades.

El enfoque fundamental de la Investigación Operativa posee muchas coincidencias con las ideas que inspiran la técnica cuya aplicación se va a estudiar en esta tesis en problemas reales. A saber, en oposición a aproximaciones más tradicionales, se busca incidir en el *estudio de un grupo de sub-sistemas que interaccionan entre sí buscando soluciones que beneficien al conjunto en global*.

En definitiva, debido a las múltiples similitudes en el campo de aplicación y en la tipología de soluciones perseguidas, entre la Investigación Operativa y la aproximación que vamos a presentar en este trabajo, vamos a centrar nuestro interés de forma más detallada, no en las técnicas que se utilizan para resolver problemas en esta disciplina, sino en los tipos de aplicaciones que se tratan y los modelos que estudia la Investigación Operativa para así poder identificar las características más representativas de los problemas que vamos a tratar en el marco de esta tesis doctoral. En este sentido, los tipos de problemas que contempla la Investigación Operativa y que nos resultan de interés son los siguientes (Taha, 2006; Bronson y Naadimuthu, 1997):

- **Problemas de optimización combinatoria:** en ellos, el gran número de combinaciones que se generan hace que enumerar todas las posibilidades sea inabordable. El conjunto de posibles soluciones es discreto o puede ser reducido a un conjunto discreto. El objetivo es encontrar la mejor combinación de opciones. Dentro de esta categoría podemos destacar los siguientes subproblemas:

- *El problema de la asignación:* en este problema se definen un conjunto de agentes y un conjunto de tareas y un coste para cada asignación agente-tarea, se debe asignar a cada tarea un agente de manera que se minimice el coste.
- *Problema del transporte:* dado un conjunto de puntos de producción y un conjunto de puntos de demanda y unos costes asociados al transporte entre cada par de puntos, se debe encontrar el plan de transporte que minimiza la suma de los costes.
- *Problemas de teoría de grafos:* en este campo se enmarcan todos los problemas cuya modelización mediante un grafo resulta de interés. Algunos problemas que se han podido resolver gracias a la teoría de grafos han sido, por ejemplo, la síntesis de circuitos secuenciales (Abdelrazik, 1993), el modelado de los trayectos de transporte público en las calles de una ciudad(Hay, 1993), etc.

- **Problemas de procesos en los que intervienen variables estocásticas:** en estos problemas, las variables del modelo no poseen valores fijos sino que han de ser modelados como distribuciones probabilísticas. Son problemas parcialmente dinámicos ya que existen variaciones dinámicas en las restricciones o en las variables que definen las condiciones del problema, pero estas siguen una distribución probabilística conocida. Ejemplos de este tipo de problemas son los problemas de líneas de espera o los problemas de demanda probabilística.

- **Problemas de procesos en los que aparecen situaciones de concurrencia:** son aquellos en los que es preciso optimizar procesos cuyas variables y/o restricciones provienen o están afectadas por diferentes ámbitos que hay que considerar para tratar el problema de forma completa. Por ejemplo, problemas de toma de decisiones en los que se obtiene información a partir de sistemas expertos, resultado de modelos de simulación, variables económicas, etcétera y se ven restringidos o afectados por variables económicas, criterios subjetivos, etc.

- **Problemas modelados mediante sistemas dinámicos:** son aquellos problemas que trabajan sobre modelos en los que la característica más relevante es la existencia de un número elevado de componentes y el gran número de interacciones que existen entre estos. Esto implica que el número de estados diferentes y transiciones entre estados es extremadamente elevado, y por tanto, son procesos en los que los métodos de análisis no son prácticos y en los que las simulaciones computacionales son la única forma de abordar el comportamiento del modelo.

Luss y Rosenwein (1997) citan algunas de las aplicaciones reales concretas pertenecientes a distintos campos que se resuelven dentro de esta disciplina de la Investigación Operativa:

- Diseño de las tablas de enrutamiento dependientes del tiempo para cada uno de los conmutadores de la red con el objetivo de minimizar el coste en redes de telecomunicaciones.
- Planificación de flotas aéreas: asignación del tipo de avión, tripulación, tipo de tarifa, coste del billete, etc.
- Diseño de una red de transporte de agua desde distintos puntos de suministro a los lugares de demanda con restricciones ambientales, políticas, etc.
- Planificación de la salida de agua de los embalses en una red hidroeléctica para el mantenimiento de múltiples plantas eléctricas.
- Industria petrolera: definición de los criterios de inversión en nuevas instalaciones y exploración y perforación entre diferentes campos petroleros.
- Planificación del suministro eléctrico para satisfacer las demandas en los picos de consumo.
- Predicción de la localización y evolución de incendios forestales. Gestión y emplazamiento eficiente de recursos para combatir incendios.
- Localización de almacenes y plantas de fabricación.

- Determinación de políticas de transporte en una red de distribución.
- Salud pública: planificación de las variaciones en las entradas de los pacientes para mantener la ocupación alta, etc. Detección y tratamiento de enfermedades.
- Servicios gubernamentales: planificación de la infraestructura de ciudades (construcción o expansión de carreteras, puentes, colegios, etc). Diseño de sistemas de emergencia eficientes. Diseño de rutas para tránsito masivo, autobuses escolares, ambulancias, etc.

Del análisis de las categorizaciones anteriores, podemos concluir que los problemas de ingeniería que se tratan en la Investigación Operativa presentan una serie de características relevantes y diferenciadoras, que se han destacado como dominantes en problemas reales (Michalewicz y Schoenauer, 1996):

1. En cuanto al tipo de variables que intervienen en los modelos de estos problemas, con frecuencia son variables continuas y discretas simultáneamente, es decir, son variables mixtas.
2. Gran parte de los parámetros que definen el problema no presentan valores fijos, sino que están representados por distribuciones estocásticas.
3. Las variables poseen también cierto nivel de ruido inherente a la captación experimental de medidas para el modelado.
4. Tanto las funciones objetivo como las variables que intervienen están sujetas a restricciones, típicamente no lineales.
5. Las variables que conforman el modelo están altamente interrelacionadas.
6. Además, este tipo de problemas suele estar sujeto a cambios en tiempo real, es decir, se requieren técnicas con capacidad de adaptación si se quiere alcanzar una solución completa.

Estas características, conjuntamente, producen problemas muy complejos de optimizar, con mucha interdependencia entre comportamientos

de los distintos elementos y que, en consecuencia, hacen que los métodos de análisis a priori no sean prácticos y sea necesario el uso de simulaciones computacionales para modelar los procesos (Schwabacher y Gelsey, 1998).

En conclusión, debemos decir en este punto que, en el marco de esta tesis doctoral, nos centraremos en el estudio de gran parte de los problemas que se resuelven en el campo de la Investigación Operativa, y que se caracterizan por su complejidad en cuanto al tipo de variables que se manejan, al gran número de elementos que interactúan y al dinamismo del sistema. Estas características los hacen susceptibles de ser tratados mediante técnicas distribuidas de resolución de problemas, como veremos más adelante.

Tras haber acotado un poco más el tipo de problemas al que nos enfrentamos, a continuación realizaremos una revisión de las técnicas y métodos de optimización de mayor relevancia en el campo de la optimización en ingeniería.

3.1.3. Métodos de optimización

Los problemas de optimización comunes en ingeniería - y los problemas concretos de la Investigación Operativa más aún - han sido adoptados con frecuencia en la literatura especializada en métodos de optimización para mostrar la efectividad de los nuevos algoritmos. Dentro del gran número de métodos existentes para problemas de ingeniería, a continuación realizaremos una revisión de las técnicas más utilizadas, agrupadas en tres grandes categorías: métodos analíticos, técnicas heurísticas y técnicas metaheurísticas.

3.1.3.1. Métodos analíticos

Previo al uso de técnicas de resolución aproximada, heurísticas o metaheurísticas, debemos mencionar las técnicas clásicas de resolución analítica de un problema de optimización. Estas se encuentran en el peldaño más alto en cuanto al nivel de información que se requiere del modelo de

un proceso, y se sirven de técnicas de análisis matemático para obtener soluciones exactas a un determinado problema.

Estas técnicas de análisis van desde la resolución de sistemas de ecuaciones lineales hasta la integración de sistemas de ecuaciones no lineales en derivadas parciales pasando por técnicas de estadística inferencial para programación estocástica. Para su uso se requiere, por un lado, la obtención de todas las ecuaciones (o modelos probabilísticos) que rigen un proceso, y por otro lado, que las técnicas de resolución existentes permitan encontrar una solución en un tiempo razonable al modelo propuesto. Sin embargo, se ha visto que la aplicación de este tipo de aproximaciones a problemas con cierto grado de complejidad, varias ecuaciones no lineales acopladas (nivel de interacción elevado) o modelos con restricciones, resulta enormemente difícil. La adaptación de estas técnicas a cada problema particular(Gill, 1980; Kosloff y Kosloff, 1983; Magyari y Keller, 2000), aún encontrando el modo de hacerlo, implica una la resolución tremendamente costosa.

De todas formas, dentro de las técnicas analíticas clásicas de optimización, podemos encontrar casos de éxito en la solución de los siguientes problemas:

- Problemas de optimización sin restricciones en donde el cálculo diferencial es la herramienta utilizada para localizar los extremos de una función.
- Problemas lineales con restricciones lineales que se enmarcan dentro de lo que se conoce como programación lineal y en la que encontramos técnicas como el método simplex o el método dual.
- Problemas no lineales con restricciones no lineales que cumplan las condiciones de Kuhn -Tucker (Kuhn y Tucker, 1951), para que la solución de la programación no lineal sea óptima. Estas técnicas, aunque datan de 1939 están todavía muy poco desarrolladas y, frecuentemente, fallan o no son suficientes para demostrar la existencia de extremos alcanzables.
- Problemas estocásticos, inspirados en su mayoría en problemas reales en los cuales, a diferencia de los problemas deterministas, las variables que definen los modelos son distribuciones de probabilidad y

se resuelven principalmente mediante técnicas de programación estocástica. Los modelos más extendidos de programación estocástica resuelven el problema mediante el uso de programación lineal en dos fases(Birge, 1997).

En conclusión, las técnicas analíticas resultan muy limitadas para tratar problemas reales y surge la necesidad del uso de otras herramientas que no van a generar soluciones exactas, sino aproximadas, pero en un tiempo razonable de cálculo. Estas son las técnicas heurísticas, que veremos en el próximo apartado.

3.1.3.2. Técnicas heurísticas

Aunque algunos tipos de problemas de optimización son relativamente fáciles de resolver mediante método exactos u obteniendo directamente la solución analítica, otros muchos, como es el caso de problemas reales que nos ocupa, son muy complejos, y han requerido del desarrollo de técnicas heurísticas como única posibilidad para obtener soluciones válidas en un tiempo de cálculo que permita su uso práctico. Así, durante las últimas cuatro décadas, se han desarrollado un gran número de algoritmos de este tipo para resolver problemas de optimización.

La mayoría de los algoritmos desarrollados entran en la categoría de las técnicas heurísticas, y están basados en los algoritmos de programación numérica lineal o no lineal pertenecientes a las técnicas analíticas, y por tanto, requieren todavía de una cantidad considerable de información del problema. Frecuentemente, las técnicas heurísticas utilizan una búsqueda intentando mejorar la solución en las cercanías de un punto inicial. En estos métodos, se busca tanto la rapidez de la búsqueda como la calidad de la solución obtenida.

Según Melián *y otros* (2003), en el campo de la Investigación Operativa, una heurística se define como: '. . . un procedimiento que proporciona un alto grado de confianza para encontrar soluciones de alta calidad con un coste computacional razonable, aunque no se garantice su optimalidad o su factibilidad, e incluso, en algunos casos, no se llegue a establecer lo

cerca que se está de dicha situación. Se usa el calificativo heurístico en contraposición a exacto . . . ’.

Como vemos, una heurística o algoritmo heurístico, es aquel algoritmo que es capaz de generar una solución aceptable con un coste computacional razonable en problemas reales. Sin embargo, estos algoritmos no nos garantizan ni un nivel de proximidad al óptimo global, ni tampoco un coste computacional máximo requerido para encontrar una solución dentro de un determinado nivel de optimalidad.

Aunque en comparación con técnicas analíticas, que requieren de un modelo matemático fiel, las técnicas heurísticas no requieren de un estudio tan profundo del modelo sobre el que se va a trabajar. Aunque sí es cierto que para desarrollar el algoritmo heurístico, se requiere una gran cantidad de conocimiento a priori y un desarrollo ad-hoc para cada tipología de problema. Para considerar que un problema está bien definido, debemos tener definidos los siguientes campos: todas las variables que entran en juego, las distintas restricciones que debe cumplir la solución al problema y la función objetivo. A partir de esta definición, se generará el algoritmo que se va a utilizar basándonos en alguna técnica ya conocida.

En general, se aplican aproximaciones heurísticas, frente a las soluciones analíticas en el caso de que: no se conozcan todas las ecuaciones que rigen el modelo con cierta precisión, no se conozca ningún método exacto para su resolución, sí exista un método exacto pero este sea muy costoso computacionalmente o que el método heurístico sea más flexible, es decir, que permita realizar variaciones en restricciones o parámetros del modelo con facilidad.

El método heurístico más representativo y simple es el de la *búsqueda exhaustiva* que consiste en: dado un problema, analizar todas las posibles soluciones y escoger la mejor. Por supuesto, este proceso será factible sólo para aquellos problemas simples que tengan un número pequeño de soluciones alternativas.

Aunque existe un gran número de heurísticas diferentes, pues se diseñan para cada problema específico, citamos a continuación los tipos más relevantes:

- Métodos de Descomposición: consiste en descomponer el problema principal en varios sub-problemas de más fácil resolución.

- Métodos Inductivos: parten de la resolución de un caso simple y generalizan hasta el problema completo.

- Métodos de Reducción: en este caso, se identifican propiedades que comparten muchas de las soluciones al problema y se introducen como restricciones para restringir el espacio de búsqueda

- Métodos Constructivos: el problema se resuelve paso a paso buscando el óptimo para cada iteración.

- Métodos de Búsqueda Local: estos procedimientos comienzan con una solución al problema y la van modificando ligeramente para ir mejorándola.

  Estos dos últimos métodos, constituyen la base de muchas de las aproximaciones metaheurísticas que veremos en el siguiente apartado.

- Metodos de gradiente: una última categoría de heurísticas que creemos tienen relevancia suficiente como para mencionar por separado, pero que también podríamos incluir dentro de los métodos de búsqueda local, son aquellas que se basan en el *gradiente de la función objetivo*. Estos métodos han sido enormemente utilizados para la resolución de problemas no lineales, y en muchos casos dan una solución muy cercana al óptimo en un tiempo muy reducido. En general, los métodos de gradiente para optimización presentan varias limitaciones, requieren problemas formulados explícitamente y modelos poco costosos computacionalmente. Sin embargo, en ingeniería, los problemas suelen conllevar una computación excesiva, como en los CFD[2] , FEA[3] u otros, y su carácter secuencial les lleva a depender de una solución inicial y a tener gran tendencia a detenerse en mínimos locales.

[2]Problemas de fluido dinámica computacional
[3]Análisis de elementos finitos

En definitiva, este tipo de aproximaciones heurísticas han demostrado gran éxito en su aplicación a múltiples problemas y han supuesto una alternativa efectiva a las soluciones analíticas para problemas reales, pero como cualquier técnica, presentan también sus limitaciones. Una de las principales es que se requiere una gran cantidad de información sobre los problemas a tratar, experiencia sobre otros casos anteriores o el conocimiento a priori de ciertas variables de los modelos. Es decir, presentan cierto grado de dependencia con respecto al problema que se va a solucionar. En el caso de que se requiera de parámetros que para algunos casos son desconocidos, estas técnicas ya no resultan aplicables.

Presentan, además, tendencia a detenerse en óptimos locales y son altamente dependientes de la solución inicial escogida. En el caso de búsquedas basadas en gradientes, cuando la función objetivo y/o las restricciones no son derivables o presentan cambios muy bruscos, la aplicación de estas técnicas puede resultar complicada e inestable.

En un intento de reducir de nuevo la dependencia de conocimiento a priori sobre el problema o la adaptación específica de las técnicas a cada escenario, surgen otras técnicas denominadas metaheurísticas que muestran una mayor capacidad para superar estas limitaciones en problemas de ingeniería (Corne *y otros*, 1999).

3.1.3.3. Técnicas metaheurísticas

Las heurísticas comentadas en el apartado anterior han resultado una estrategia realmente útil para obtener el óptimo global en modelos reales simplificados, como se puede concluir de una revisión de la bibliografía. Las limitaciones de este tipo de aproximaciones numéricas en los problemas sin idealizar han forzado a los investigadores a confiar en metaheurísticas para solucionar la mayoría de los problemas de optimización en ingeniería.

La denominación metaheurística fue acuñada originalmente por Glover (1986) y bajo este nombre se enmarcan todos los métodos que, basándose en las heurísticas anteriores en mayor o menor medida, buscan obtener un mayor rendimiento que el que ofrecen estas aplicándose, fundamentalmente, cuando las heurísticas convencionales no dan buen resultado.

Desde entonces han aparecido muchas técnicas que se enmarcan dentro de esta categoría.

La función que debe ser optimizada se denomina función objetivo y con respecto a la metaheurística funciona como una caja negra, es decir, devuelve el valor de la función objetivo para una combinación de variables determinada, pero su funcionamiento interno no es conocido. La única información que se requiere en muchos casos, aunque depende de la metaheurística utilizada, está relacionada con valores, como por ejemplo, los rangos de las variables o medidas de proximidad para generar soluciones cercanas. Las metaheurísticas proporcionan también el óptimo actual en cada iteración del proceso para permitir hacer un seguimiento del progreso de la búsqueda y detenerla cuando la relación entre el tiempo de procesado y el valor de calidad obtenido sea satisfactoria.

Los profesores Osman y Kelly (Osman y Kelly, 1996) dieron la siguiente definición para los métodos metaheurísticos:

'Los procedimientos Metaheurísticos son una clase de métodos aproximados que están diseñados para resolver problemas difíciles de optimización combinatoria, en los que los heurísticos clásicos no son efectivos. Los Metaheurísticos proporcionan un marco general para crear nuevos algoritmos híbridos combinando diferentes conceptos derivados de la inteligencia artificial, la evolución biológica y los mecanismos estadísticos'

Una metaheurística puede ser considerada como un marco algorítmico general que puede ser aplicado a diferentes problemas de optimización con relativamente pocas modificaciones para adaptarlos a cada problema específico (Blume y Easley, 2002). Por tanto, las metaheurísticas podemos situarlas a un nivel superior a las heurísticas, pues en cierto modo, se sirven de técnicas heurísticas, y de entre estas, principalmente de las heurísticas constructivas y de búsqueda local.

Podemos clasificar los distintos tipos de metaheurística según el tipo de heurística que usan como base. Así distinguimos los siguientes:

- *Metaheurísticas constructivas:* estas van incorporando elementos para ir generando la solución, un ejemplo muy representativo es el

método GRASP (Greedy Randomized Adaptative Search Procedures) (Feo y Resende, 1995).

- *Metaheurísticas evolutivas:* son métodos poblacionales, es decir, la búsqueda se realiza en paralelo mediante una población de heurísticas. El procedimiento de búsqueda puede resumirse en cuatro etapas que se repiten secuencialmente: generar, seleccionar, combinar y reemplazar un conjunto de soluciones. Ejemplos representativos son los algoritmos genéticos (Holland, 1975) y la búsqueda dispersa (Glover, 1994).

- *Metaheurísticas de búsqueda:* se basan en procedimientos de búsqueda local. Su objetivo es encontrar la solución global y evitar los mínimos locales. Para ello, se requiere de la definición de una serie de relaciones de vecindad sobre el espacio de búsqueda. En general, cada solución posee más de una solución vecina y la selección de la solución siguiente se basa únicamente en información local respecto a la solución actual (por eso que se denomina búsqueda local). Con el objeto de evitar mínimos locales, que es el principal inconveniente de estas técnicas, se utilizan tres soluciones:

  - Reiniciar la búsqueda desde otra solución inicial.
  - Modificar la estructura del espacio de búsqueda.
  - Permitir movimientos que empeoren la solución actual para escapar de mínimos locales.

  Ejemplos de estas metaheurísticas son la búsqueda tabú (Glover, 1977) y el recocido simulado (Kirkpatrick *y otros*, 1983). Cabe comentar que la selección de las medidas de vecindad en este tipo de procedimientos es fundamental y por eso deben ser elegidas adecuadamente.

Otra clasificación para las técnicas metaheurísticas es aquella que diferencia las que están basadas en fenómenos naturales y las que no. Según esto, dentro del primer grupo podemos encontrar aquellas inspiradas en procesos evolutivos biológicos (Fogel *y otros*, 1966; De˜Jong, 1975; Koza, 1990), los algoritmos genéticos que se basan en las teorías

Darwinistas de selección natural (Holland, 1975; Goldberg, 1989), aquellas inspirados en comportamiento animal, como los algoritmos de colonias de hormigas (Cole, 1991), o el recocido simulado (Kirkpatrick *y otros*, 1983), inspirado en el proceso termodinámico del recocido de los sólidos. En el segundo grupo estarían aquellas metaheurísticas como la Búsqueda Tabú (Glover, 1977) o la Búsqueda Local Iterada (Stützle, 1998) que no están basadas en procesos naturales.

Desde los años 70, muchas metaheurísticas, combinando reglas y aleatoriedad, se han ido adaptando para mejorar los métodos heurísticos y resolver de manera concreta problemas de optimización en ingeniería. A continuación describiremos brevemente algunas de ellas:

- *Recocido simulado:* es una generalización del método de Montecarlo (Metropolis *y otros*, 1953) y está basado en la manera en la cual los líquidos se congelan o los metales se recristalizan en un proceso de recocido. En estos procesos, el material fundido inicialmente a altas temperaturas y desordenado, se enfría lentamente para que el sistema en un determinado momento pase de nuevo a un estado de equilibrio (lo que correspondería a una solución). A medida que se enfría, el sistema se va volviendo más y más ordenado. Si la temperatura inicial es demasiado baja o si el enfriamiento es demasiado rápido, el sistema puede templarse y pueden formarse defectos o estados metaestables (que sería equivalente a detenerse en mínimos locales). La conexión entre este algoritmo y la minimización matemática fue señalada en primer lugar por Pincus (1970) pero fue Kirkpatrick quien desarrolló las bases de la aplicación de este fenómeno en forma de una técnica de optimización para problemas principalmente combinatorios (Kirkpatrick *y otros*, 1983).

- *Búsqueda tabú:* fue originalmente sugerida por Glover (1977), y es básicamente un descenso de gradiente con memoria. La memoria retiene el número de estados visitados previamente para evitar volver a ellos en un plazo no deseado. Esta información se almacena en una lista tabú y de ahí su nombre. La definición de qué es un estado y la longitud de la lista son parámetros críticos de diseño. Se usan otros dos elementos, aspiración y diversificación, que se utilizan para solucionar los casos en los que todos los vecinos están en la lista tabú y para añadir aleatoriedad a la búsqueda respectivamente.

- *Algoritmo genético:* es un algoritmo de búsqueda basado en la selección natural y los mecanismos de la genética de poblaciones. La teoría fue propuesta por Holland (1975) y más tarde desarrollada entre otros por Goldberg (1989). Un genético simple posee tres operadores: selección, cruce y mutación. La selección es el proceso de supervivencia del elemento de mayor calidad. El cruce es el intercambio de información genética entre los padres para producir descendientes. La principal característica de los algoritmos genéticos es la evaluación simultánea de muchas soluciones, es decir, es una metaheurística poblacional. Esta característica los diferencia de los algoritmos numéricos u otras heurísticas, como el recocido simulado o la búsqueda tabú, que evalúan una sola solución por cada paso. Esto proporciona una ventaja en cuanto a ampliar el alcance de la búsqueda y evitar, en cierta medida, la convergencia prematura a óptimos no globales.

- *Estrategias evolutivas:* son similares a los algoritmos genéticos y fueron desarrolladas para resolver problemas de optimización de parámetros (Schwefel, 1977; Rechenberg, 1973). Las estrategias evolutivas usan la mutación como el operador dominante en lugar de la recombinación como en el caso de los algoritmos genéticos.

- *Programación evolutiva:* desarrollada inicialmente por Fogel *y otros* (1966), es básicamente una variación de las estrategias evolutivas en la que se modifica la representación que se realiza de los individuos. El algoritmo utiliza selección mediante torneo aleatorio y mutación como operadores principales.

- *Programación genética:* es una extensión de los algoritmos genéticos, fue desarrollada por Koza (1990) y busca encontrar un programa que realice las tareas deseadas, y para ello evoluciona una población de programas. Se han utilizado lenguajes como el LISP o el Scheme para representar las soluciones en lugar de las cadenas de números de un genético convencional.

- *Optimización mediante enjambres de partículas:* es un método de optimización inspirado en el comportamiento social de pájaros en bandada o bancos de peces (Kennedy y Eberhart, 1995), pero que

no usa evolución, sino que se basa en una serie de reglas predefinidas que guían al enjambre de partículas hacia la solución.

- *Algoritmo HS:* Lee y Geem (2005) desarrollaron una nueva metaheurística basada en el concepto de armonía musical, y esta ha demostrado ser un algoritmo eficiente para obtener soluciones a problemas de ingeniería. Este se basa en el proceso de búsqueda de un estado perfecto de armonía y utiliza búsqueda estocástica aleatoria en lugar de una búsqueda mediante gradiente.

- *GRASP:* fue introducido por Feo y Resende (1995) y es un método multi-arranque en donde cada paso consiste en una fase de construcción y otra de mejora. En la fase de construcción se aplica una heurística constructiva para obtener una solución inicial y en la segunda fase dicha solución se mejora mediante un algoritmo de búsqueda local. Los algoritmos voraces y el método del vecino más cercano son dos variantes del método GRASP.

Como conclusión a este apartado diremos que las técnicas metaheurísticas han venido proporcionando resultados muy satisfactorios en problemas reales de ingeniería, aunque todavía resta mucho por hacer. Una de las principales limitaciones que hemos encontrado a estas técnicas aparece cuando tratamos de resolver problemas dinámicos o problemas cuyos flujos de información y/o nivel de interacciones se reducen al ámbito local, encontrándose por tanto altamente descentralizados. A continuación trataremos esta limitación de forma detallada.

3.1.4. Problemas de optimización dinámica

Desde la aparición de las técnicas heurísticas, y principalmente de las metaheurísticas que surgieron más tarde, el número y tipo de aplicaciones reales resueltas satisfactoriamente es realmente elevado. En efecto, las metaheurísticas han supuesto un nuevo enfoque más pragmático y generalista y que ha abierto las puertas tanto al tratamiento de problemas que las soluciones analíticas no permitían, como al tratamiento de ciertos problemas de una forma más general que con las soluciones específicas propias de las técnicas analíticas.

Sin embargo, y a pesar de que se han desarrollado técnicas para dominios claramente específicos como la optimización multi-objetivo, optimización sujeta a restricciones, etc., y del amplio uso en que se ha hecho de estas técnicas en ingeniería, matemáticas, inteligencia artificial, etc., podemos fácilmente concluir que todos ellos tienen un punto en común: el uso de una (o varias en optimización multiobjetivo) función objetivo que no cambia durante el tiempo que dura la búsqueda.

Con respecto a esto, consideramos que también son de interés los dominios de optimización en que, o bien la función objetivo o las variables y restricciones del modelo incorporan algún tipo de ruido, o fluctúan a lo largo del tiempo. Estos problemas son denominados No Estacionarios o Dinámicos, y el creciente interés que actualmente suscitan está abriendo las puertas a la investigación en un nuevo dominio: DOP (*Dynamic Optimization Problems*, Branke (2001)). Podríamos definir un problema de optimización dinámica como *aquel en el que se precisa encontrar una solución óptima para una función objetivo determinada cuando o bien esta función o bien los parámetros que afectan al modelo varían con el tiempo*.

Existen un gran número de aplicaciones del mundo real que pueden modelarse como un problema DOP. Este es el caso de la red de semáforos de una ciudad, la selección de rutas en una red de comunicación o los problemas con distribución de demanda variable. Todos estos escenarios son inherentemente dinámicos. Debemos tener en cuenta, por tanto, que las técnicas que traten de optimizar un escenario dinámico deben tener esto en cuenta a la hora de hacer su búsqueda del óptimo pues el espacio de búsqueda cambia a medida que pasa el tiempo.

La definición más básica que podemos realizar para un problema de optimización dinámica es aquella en la que existen dos funciones objetivo entre las que el sistema fluctúa y el cambio se produce en un determinado instante de tiempo, repitiéndose periódicamente esta fluctuación a lo largo de todo el proceso. De forma más general, esto podría extenderse a cambios graduales y no periódicos entre varias funciones objetivo y distintas configuraciones de parámetros. Los problemas que representan conceptualmente los límites de los problemas dinámicos de optimización son, en un extremo, los problemas estacionarios, en los cuales no existen cambios en la función objetivo o las condiciones, y por otro lado, los pro-

blemas definidos con funciones ruidosas, en los cuales en cada paso de tiempo se presentan variaciones no periódicas en la función objetivo o en las condiciones. Por tanto, la estrategia que proporcione una solución satisfactoria para resolver estos problemas debe adaptarse intrínsecamente a los cambios que se produzcan en el sistema.

Un clásico entre los problemas dinámicos es el de la mochila dinámica (Goldberg y Smith, 1987), que consiste básicamente en el problema clásico de la mochila[4] pero que, en este caso, vamos a modificar convirtiéndolo en dinámico variando el valor de capacidad de la mochila cada cierto periodo de tiempo. Este problema resulta un ejemplo muy representativo para el marco de esta tesis por ser NP-Completo, dinámico y por incorporar restricciones. Estas características son muy usuales en problemas reales de ingeniería.

Los problemas DOP se han venido tratando con distintas metaheurísticas diseñadas 'ad-hoc', pero las que mejores resultados han proporcionado en casos generales han sido las basadas en algoritmos evolutivos. Esto es debido a la característica fundamental del uso de una población de soluciones que le permite tolerar el dinamismo de una forma más eficiente al poseer un conjunto de soluciones candidatas susceptibles de adaptarse al cambio. En concreto, su utilización en DOP ha dado lugar a la aparición de varias estrategias, que se pueden englobar en tres generales:

- *Incorporación de Memoria Histórica:* en este caso, se permite al algoritmo evolutivo utilizar información relativa a soluciones encontradas en instantes anteriores. Se incorpora algún tipo de memoria que puede utilizarse a nivel de algoritmo o bien a nivel de genotipo (Ng y Wong, 1995). Esto es, puede utilizarse una memoria explícita, de manera que se mantengan en la población los individuos de mayor calidad, o una memoria implícita utilizando genotipos n-ploides. Otras aproximaciones han utilizado una combinación de ambas, por una lado una memoria implícita en la codificación de los cromosomas (mediante el uso de genes promotores) combinada con el uso

[4] El problema de la mochila consiste en *seleccionar una serie de elementos, teniendo en cuenta que cada elemento tiene un volumen determinado y una utilidad determinada, de manera que no se exceda un valor límite de capacidad y con el objetivo de maximizar la suma de los valores de utilidad.*

de una memoria externa explícita, que ha proporcionado resultados muy satisfactorios (Bellas *y otros*, 2008).

- *Conservación de la Diversidad:* con esta estrategia se busca mantener un alto nivel de diversidad en la población. Se considera como una técnica preventiva para enfrentarse al proceso de cambio, ya que continuamente mantiene una población capaz de realizar una búsqueda con un nivel satisfactorio de exploración. Existen muchas técnicas que buscan esta diversidad por ser una característica útil también en dominios no dinámicos. Algunos ejemplos son la selección disruptiva (Kuo y Hwang, 1997), o el uso de evolutivos con población estructurada (Sarma y Jong, 1999), etc.

- *Reacción Frente al Cambio:* se programan acciones concretas en un evolutivo estándar para adaptar la búsqueda a la nueva situación solamente cuando el cambio sucede. Por ejemplo, elevar la tasa de mutación tras la aparición de un cambio. Estas técnicas requieren del conocimiento previo o de la detección de los cambios, algo que no es posible en multitud de problemas reales.

Estas estrategias buscan adaptar los algoritmos que tradicionalmente se han utilizado para evolucionar sistemas estáticos a las variaciones de un sistema dinámico, aumentando la complejidad de la población de búsqueda mediante el uso de memoria histórica, estudiando cuando se producen los cambios e intentando contrarrestarlos o bien manteniendo a la población preparada para realizar una nueva exploración - con la consecuente pérdida de eficiencia en cada búsqueda-. Esta serie de herramientas ‘ad-hoc’ son dependientes del tipo de problema y del conocimiento que se tenga del dinamismo en el sistema.

3.1.5. Conclusión

De este primer apartado debemos destacar que los problemas que se enmarcan dentro del alcance de la Investigación Operativa poseen una serie de características que los hacen especialmente complejos para ser tratados con técnicas analíticas o heurísticas simples. Solo en el campo de la optimización metaheurística encontramos aproximaciones que han resuelto de forma satisfactoria este tipo de problemas, aunque en dominios acotados. El principal problema es que el enfoque utilizado en la mayor

parte de los casos no es general, ya que no parte de una premisa básica que caracteriza a los problemas reales de la Investigación Operativa: el gran número de interacciones entre las variables y su dinamismo. En este sentido, únicamente dentro de la computación evolutiva se encuentran algoritmos específicos para tratar con problemas de optimización dinámica.

Tras esta revisión, podemos acotar el alcance práctico de esta tesis doctoral, y de todos los problemas que se engloban en el campo de la Investigación Operativa, pasamos a establecer el tipo concreto a los que nos referiremos en el resto de esta memoria. Son aquellos caracterizados por dos propiedades:

- **Dinamismo:** esta característica es típica de los problemas reales que dependen del tiempo, ya que en la mayoría de ellos se producen diversos cambios en tiempo real que pueden afectar a las variables del sistema, a la función objetivo, a las restricciones, etc. Algunos de estos cambios son intrínsecos al problema y otros se derivan de fallos de funcionamiento.
- **Descentralización:** existe un gran número de problemas que, o bien requieren que para su resolución la tarea se lleve a cabo de forma distribuida, o bien este enfoque proporciona una mejora significativa en la utilidad de la solución. Dentro del tratamiento de los problemas que requieren una distribución de la tarea, la resolución de esta puede ser descentralizada o centralizada. En el marco de esta tesis, nos centraremos únicamente en problemas que requieran una resolución descentralizada, característica que engloba a dos típicamente relacionadas:
  - *Conocimiento parcial:* en numerosos ejemplos de problemas de ingeniería se posee únicamente conocimiento parcial de las variables en juego, de forma que la optimización del objetivo no se puede asegurar mediante una resolución centralizada.
  - *Interacciones locales:* por otro lado, también es común que la función objetivo que se pretende optimizar resulte de la combinación no trivial de interacciones locales, lo que implica que, de nuevo, un enfoque centralizado presente grandes dificultades para proporcionar una solución que contemple el estado global del problema en cada instante de tiempo.

Hablaremos pues de la resolución, no de cualquier tipo de problema de optimización en ingeniería, sino de aquellos que sean *dinámicos y descentralizados*. El siguiente apartado entra a revisar los principales desarrollos en este ámbito.

3.2. Resolución de problemas dinámicos descentralizados

La aproximación que planteamos en esta tesis para tratar los problemas dinámicos descentralizados en ingeniería, toma su inspiración de dos campos que tienen como principio intrínseco la distribución del cómputo entre sus elementos componentes y además asumen el dinamismo del sistema como característica básica: por una parte la Inteligencia Artificial Distribuida (o Sistemas Multiagente), y por otra, los Sistemas Complejos.

Las aproximaciones que podemos encontrar en la Inteligencia Artificial Distribuida se encuadran en lo que se conoce como métodos de resolución 'top-down'. En este ámbito, se fijan los resultados o comportamientos de alto nivel que se desean para solucionar el problema de forma distribuida y en consecuencia se diseñan las estrategias de coordinación, comunicación, etc., con objeto de lograr dicho resultado. Este tipo de método de adaptación/aprendizaje 'ad-hoc' de las interacciones las hace, en cierta medida, dependientes del diseñador. Por otro lado, tal y como se menciona en Vlassis (2003), los Sistemas Multiagente que se utilizan dentro de la Inteligencia Artificial Distribuida se caracterizan por trabajar inherentemente en sistemas dinámicos. Esto implica que cada uno de ellos opera en un medio cambiante debido a la presencia de otros agentes, lo cual hace que su comportamiento sea robusto frente a los cambios en comparación con otras aproximaciones. La principal aportación de este campo a la resolución de problemas dinámicos descentralizados proviene de los numerosos estudios que se han llevado a cabo a la hora de establecer funciones objetivo que posibiliten la coordinación entre los agentes del sistema.

Por otro lado, en el campo de los Sistemas Complejos, se han venido afrontando los problemas mediante aproximaciones 'bottom-up', en

las que los investigadores parten de la definición de los modelos básicos de interacción entre los elementos que conforman un sistema complejo y se centran en el estudio macroscópico de las diferentes dinámicas de interacción que se puedan producir. La mayoría de los trabajos buscan la emergencia de un comportamiento estable (Antsaklis *y otros*, 1998) sin fijar un objetivo de diseño concreto. La principal aportación de este campo a la resolución de problemas dinámicos descentralizados proviene del estudio de cómo plantear sistemas con gran número de elementos que interactúan en cuanto a qué parámetros son relevantes, qué reglas de interacción, cómo se controla la evolución, etc.

Nuestra intención en esta tesis es combinar las ventajas de ambas aproximaciones, y partiendo de una especificación clara del objetivo a lograr mediante el establecimiento de *funciones objetivo* (top-down), definir modelos y *reglas de interacción simples* en sistemas multi-componente (bottom-up) que lleven a la resolución de una tarea dinámica descentralizada de forma adecuada, pero permitiendo que dicha solución ideal *emerja* de las interacciones entre los elementos. Este enfoque nos permitirá obtener sistemas que se *coordinan y autoorganizan* en configuraciones estables dentro de escenarios dinámicos, pero con la condición básica de que deben resolver algún tipo de tarea concreta.

La aplicación de las técnicas propias de la Inteligencia Artificial Distribuida a la resolución de problemas de optimización en ingeniería ha sido ampliamente tratada, y su aplicación exitosa en casos reales es cada vez mayor. El análisis de los problemas dinámicos mediante técnicas de Sistemas Complejos también se ha utilizado en diversos ámbitos de la ingeniería, aunque su rango de aplicación real haya sido mucho más discreto. De todas formas, estos dos campos parecen alejados del tipo de problemas de Investigación Operativa que se han puesto como paradigma de aplicación en el apartado anterior por lo que, a continuación, pasaremos a revisar los principales desarrollos llevados a cabo en cada uno de ellos de cara a centrar nuestro enfoque.

3.2.1. Inteligencia Artificial Distribuida

En este apartado se realizará una revisión de las principales aportaciones del campo de la Inteligencia Artificial Distribuida en el ámbito de

resolución de problemas mediante una aproximación multicomponente. Es en este campo donde se encuentran las técnicas con mayor aplicabilidad y donde mayor número de contribuciones a problemas reales han surgido, incluyendo problemas de optimización en ingeniería. El objetivo final de este apartado es mostrar los principales resultados obtenidos en este campo a la hora de diseñar funciones de utilidad que conduzcan a la obtención de soluciones satisfactorias y distribuidas.

3.2.1.1. Inteligencia Artificial vs. Inteligencia Artificial Distribuida

La aproximación moderna a la Inteligencia Artificial (IA) está centrada sobre el concepto de agente racional. La definición más simple y genérica del concepto de agente es la proporcionada por Russell y Norvig (2003): *'un agente es cualquier cosa que pueda percibir su entorno a través de sensores y que pueda actuar sobre ese entorno mediante actuadores'.*

A partir de ella, se establece que un agente cuyo comportamiento está dirigido a optimizar una medida de eficiencia determinada se denomina agente racional[5]. Esta definición es también muy genérica y podría incluir agentes humanos (por ejemplo, con ojos como sensores y manos como actuadores), agentes robóticos (por ejemplo, con cámaras como sensores y ruedas como actuadores) o agentes software que habitan en entornos virtuales (por ejemplo, con un interfaz gráfico como sensor y actuador). En el marco de esta tesis, asumimos que el campo de la Inteligencia Artificial se encarga del *estudio de los principios de diseño de agentes racionales*.

Durante las últimas cuatro décadas, el interés en las técnicas para dotar de inteligencia a agentes, bien para el aprendizaje máquina o para tareas de optimización - aprendizaje máquina no deja de ser una tarea de optimización en la que el objetivo es reducir el valor de discrepancia con un patrón de comportamiento requerido -, se centró fundamentalmente en aplicaciones de un solo agente. Los diferentes avances que se obtuvieron en el campo se basaron en mejorar la eficiencia del aprendizaje o bien

[5]Agente racional: sistema computacional que habita en un entorno dinámico complejo, siente y actúa de forma autónoma en este entorno y de este modo lleva a cabo un conjunto de objetivos o tareas para los que ha sido diseñado (Maes, 1990)

mejorar la cantidad de conocimiento proporcionado, incrementarlo o simplemente tratarlo de otro modo.

En menor medida, se han venido desarrollando sistemas en los que se enfocan los problemas mediante conjuntos de agentes que se coordinan de cara a resolver la tarea (Kraus y Plotkin, 2000; Zhong *y otros*, 2004). Esta coordinación se suele lograr mediante *cooperación* entre los agentes, pero también es posible alcanzarla mediante *competición*, como veremos más adelante. En los SMA, el control y la información están con frecuencia distribuidos entre los agentes, consiguiendo así una serie de ventajas tales como la distribución del cómputo, la sencillez de los elementos, la robustez y la tolerancia a fallos, que veremos en detalle más adelante.

Este tipo de sistemas formados por múltiples agentes se estudian en el campo de la *Inteligencia Artificial Distribuida* (en adelante IAD) que es considerado como un subcampo de la IA dedicado al estudio de las técnicas y el conocimiento necesario para la coordinación y distribución del conocimiento y las acciones en un entorno con múltiples agentes racionales.

Del mismo modo que la IA se encarga del diseño de agentes racionales, podemos ver la IAD como aquella disciplina que se encarga del diseño de *Sistemas Multiagente* racionales, nomenclatura más habitual en la actualidad para este campo, y sobre la cual basaremos el resto del apartado. De hecho, el concepto de IAD ha quedado en desuso y se asocia con frecuencia a los primeros sistemas inteligentes con distribución de la tarea. Así, en lo que resta de apartado nos referiremos de forma genérica a este campo como Sistemas Multiagente.

3.2.1.2. Sistemas Multiagente. Conceptos generales

Dentro del marco de la tesis, un Sistema Multiagente (SMA), además de las características convencionales que citaremos posteriormente, es un sistema que explota la distribución de una tarea en subtareas de un modo centralizado o descentralizado, y cuya investigación asociada se ha focalizado en el estudio de las estrategias de coordinación (Bowles y Gintis, 2004; Wooldridge y Jennings, 1999), comunicación (Kraus *y otros*, 1998; Deloach *y otros*, 2001), división de tareas, etc. Principalmente, en el

campo de los SMA los procesos de adaptación/aprendizaje son desarrollados e introducidos por el diseñador. Se definen, por ejemplo, protocolos de negociación, comunicación o sistemas de subastas (Bernardine˜Dias *y otros*, 2006; Preist *y otros*, 2001) para la asignación de tareas.

Debemos aclarar en este punto que no estamos interesados en este ámbito del campo de los SMA. El diseño a priori de las estructuras de coordinación hace las aplicaciones menos robustas frente a variaciones del entorno, más específicas para cada una de las aplicaciones y limita la libertad para generar dinámicas de interacción que supongan soluciones realmente novedosas. En cambio sí estamos interesados en algunas otras aproximaciones que utilizan técnicas como el aprendizaje por refuerzo o la computación evolutiva para obtener el comportamiento global deseado a partir de los comportamientos individuales de forma no predefinida. Es en estas aproximaciones donde se desarrolla la teoría de *funciones de utilidad*, que será clave en el desarrollo de esta tesis, como mecanismo para fijar el objetivo global a lograr por el SMA a partir del cumplimiento de objetivos individuales.

Las principales características de un Sistema Multiagente son las siguientes (Weiss, 2000; Wooldridge, 2002):

1. *Inteligencia:* entendida como la capacidad de proporcionar soluciones originales ante problemas no contemplados en su diseño. Son, por tanto, sistemas adaptativos que de forma intrínseca pueden ser aplicados en problemas dinámicos. Esta capacidad ha dado lugar al uso de nomenclaturas más específicas tales como agentes racionales o agentes cognitivos, para distinguir estos sistemas de otros 'no inteligentes'. Los agentes inteligentes pueden actuar de forma 'independiente' aún formando parte de un sistema colectivo.

2. *Autonomía:* un SMA posee continuidad temporal, es decir, persiste. Además, posee sensores y actuadores propios que le dotan de capacidad de respuesta individualizada.

3. *Distribución de la tarea:* un SMA está caracterizado por ser multicomponente, y de ahí derivan sus principales ventajas e inconvenientes. La distribución de la tarea a realizar implica otras dos características básicas:

a) *Coordinación:* los agentes deben coordinar su acciones de cara a obtener una ventaja a la hora de realizar la tarea global, bien porque esta no pueda ser resuelta de forma centralizada o bien porque la resolución distribuida sea más eficiente. La coordinación se suele alcanzar mediante cooperación entre los agentes, pero esto no es obligatorio.

b) *Comunicación:* un SMA debe poseer algún tipo de comunicación entre sus elementos para alcanzar la coordinación anterior.

Del análisis de estas características ya podemos extraer una serie de ventajas inherentes a la estructura multiagente:

1. En primer lugar y debido al carácter distribuido, un incremento de la velocidad y eficiencia.
2. En segundo lugar, debido a la computación asíncrona y paralela, *robustez* y *fiabilidad*, de modo que el sistema puede presentar una degradación gradual cuando uno o más agentes desaparecen.
3. *Escalabilidad*, ya que es sencillo añadir nuevos agentes al sistema.
4. Coste de *diseño*, pues el coste de varios elementos sencillos es menor que el de un elemento complejo que integre varias funcionalidades.
5. Su uso facilita el desarrollo y la reusabilidad, ya que es más fácil mantener y desarrollar un sistema modular que uno monolítico.

En cuanto al uso de estructuras SMA para el tratamiento de problemas reales de ingeniería, Baker *y otros* (1997a) sostienen un punto de vista que profundiza en esta ventajas, y afirman que existen una serie de características que han impulsado la aplicación de los SMA a problemas reales de ingeniería. Estas características son: modularidad, descentralización, capacidad de adaptación, sistemas débilmente estructurados y capacidad de representar dinámicas complejas. Si el problema a tratar presenta necesidad de tratar con estas características, una aproximación basada en SMA puede ser adecuada. Describiremos cada una de estas ventajas con mayor detalle:

- *Modularidad:* los agentes son objetos proactivos, y comparten las ventajas de los ya tan extendidos paradigmas basados en objetos, es por tanto que los sistemas de agentes están estructurados en módulos. Si la aplicación puede ser particionada en un conjunto de entidades, puede ser interesante la aproximación multiagente.
- *Descentralización:* muchos procesos industriales se pueden organizar de un modo descentralizado. La tendencia en gestión empresarial ha sido también hacia los modelos descentralizados, dividiendo la empresa en elementos independientes con competencias bien definidas.
- *Adaptabilidad:* entendida aquí como la suma modularidad y descentralización. Permite soportar frecuentes cambios dejando la mayoría del sistema ajena a los cambios locales.
- *Débilmente estructurado:* el diseño de algunos sistemas llega a lo que se denomina 'parálisis por análisis' debido a la necesidad de conocer con exactitud la estructura final, que a veces no es posible conocer hasta que se haya realizado más parte del diseño. La arquitectura de agentes permite al diseñador generar un sistema con las clases que conforman una estructura determinada en lugar de tener que generar la estructura en sí. Esto aumenta la utilidad del sistema resultante y reduce el coste de mantenimiento y reconfiguración además de eliminar el problema que hemos planteado.
- *Complejidad:* un sistema de agentes que interactúan puede crear una gran combinación de comportamientos y soluciones, como requieren las múltiples y complejas soluciones que se necesitan en aplicaciones industriales.

3.2.1.3. Tipos de SMA

De cara a proporcionar una visión actual y general del campo de los SMA, el presente apartado se ha estructurado en base a los distintos tipos de SMA que se han desarrollado en los últimos años. Esto permitirá acotar la gran cantidad de aportaciones en marcos más concretos y así llegar al tipo de sistemas y aproximaciones que más cercanas están a los problemas de ingeniería que nos competen. De esta forma, podemos clasificar los SMA en base a:

- *Tipo de entorno:* el tipo de entorno en el que se desenvuelve un SMA puede ser real o virtual. Los agentes que operan en el mundo real poseen sensores y actuadores cuyo dominio es el entorno real, con toda la dificultad que esto conlleva. Como se establece en Russell y Norvig (2003), el entorno real es: dinámico, no determinista, continuo, inaccesible y no episódico. Estas propiedades tomadas en su definición más formal hacen que en un entorno real se muy complejo abordar la resolución de tareas mediante un SMA. Por este motivo, todas ellas se suelen simplificar o 'relajar' de forma que se suele trabajar con entornos estáticos, deterministas, discretos, accesibles de forma parcial y episódicos. Las soluciones alcanzadas con estas simplificaciones han resultado satisfactorias hasta el momento, como en el caso de aplicaciones a sistemas multi-robot (Mataric, 1997; Thrun *y otros*, 2000; Rekleitis *y otros*, 2004), sobre todo centradas en el estudio de estrategias de coordinación simples.

  Por otro lado, los SMA que habitan en mundos virtuales, formados por los denominados agentes software, se han centrado más en el estudio de estrategias de cooperación, competición, negociación, etc. (Decker y Lesser, 1995; Gmytrasiewicz y Durfee, 2001), ya que la sensorización y actuación se presuponen ideales y no imponen ningún límite. Entre ambos tipos de SMA existen numerosas contribuciones híbridas (Yamaguchi *y otros*, 1996; Gutierrez-Maldonado *y otros*, 2008), donde el SMA se desarrolla en entornos reales simulados (entornos virtuales más realistas) con miras a su posterior aplicación en entornos reales. Este tipo de aproximaciones son necesarias de cara a optimizar la fase de aprendizaje, que de realizarse íntegramente en el entorno real puede resultar infructuosa.

- *Tipo de percepción:* puede ser individual o distribuida. Es decir, el entorno es completa o parcialmente observable para cada uno de los agentes del SMA. En este último caso, cada uno de los agentes no posee conocimiento de todo el entorno sino solo conocimiento parcial. Obviamente, esto ha de ser tenido en cuenta a la hora de tomar decisiones colectivas, de forma directa o indirecta. Pero al tratar el sistema en su globalidad, la información colectiva que llega a los sensores de los agentes individuales en un SMA está distribuida, es decir, los agentes pueden observar datos que proceden de

diferentes posiciones espaciales, temporales o incluso semánticas (diferentes modos de comunicación de cada agente). Esto implica la posibilidad de utilizar estrategias para 'fusionar' estas fuentes de información (percepciones de los agentes) y así mejorar el conocimiento colectivo con respecto al estado actual (Luo y Su, 2003; Molina˜López *y otros*, 2003).

- *Tipo de comunicación:* una de las propiedades fundamentales de un SMA es la coordinación, para la cual es necesario algún tipo de comunicación entre los agentes. El tema de la comunicación es de gran relevancia y las dificultades que surgen para conseguir una comunicación efectiva han sido ya tratadas en diversos trabajos (Ferber, 1999; Steels, 1998). En función del medio utilizado para la comunicación podemos distinguir dos tipos de estrategias comunicativas, las directas, que requieren protocolos e incluso hardware dedicado, y las indirectas, que se realizan mediante modificaciones del entorno, también denominada comunicación tipo *stigmergy* (Theraulaz y Bonabeau, 1993). En general, en aquellos SMA en los que existe comunicación explícita, esta suele ser del tipo de comunicación directa, aunque podemos encontrar también algunas excepciones principalmente dentro del campo de inteligencia de enjambre.

- *Composición del equipo:* distinguimos SMA homogéneos frente a SMA heterogéneos (Anderson *y otros*, 2002). Un grupo de agentes heterogéneos puede serlo debido a diferencias intrínsecas al agente, diferencias hardware, software o de diseño de sensores/actuadores. También se suelen considerar heterogéneos a los SMA que, aunque inicialmente presenten un mismo diseño a nivel hardware, tras el paso del tiempo muestran comportamientos diferentes que pueden afectar desde a la percepción hasta a la toma de decisiones. Los SMA homogéneos suelen estar formados por un número elevado de agentes que presentan estrategias de control simples, normalmente basadas en reglas, y es la interacción con otros agentes la que hace que emerja una 'inteligencia colectiva' (Chades *y otros*, 2003).

- *Tipo de control:* existen SMA con control centralizado, distribuido y aproximaciones mixtas. A diferencia de los sistemas de un solo agente, el control en un SMA, idealmente, debería ser totalmente distribuido (descentralizado). Esto implica que no hay un proce-

so central que recopile información de cada agente y luego decida qué acción se debe realizar, lo que redunda en mayor robustez y tolerancia a fallos del SMA. La toma de decisiones para cada agente recae, en gran medida, en el propio agente. En un grupo de agentes, la toma de decisiones distribuida supone ciertas mejoras en velocidad de cómputo pero también presenta el inconveniente de que se requiere del desarrollo de mecanismos de coordinación. Esta coordinación debe garantizar que las decisiones de los agentes individuales resulten en una acción conjunta beneficiosa. Los SMA con control totalmente distribuido se han utilizado, mayoritariamente, con agentes homogéneos y simples. Las aplicaciones prácticas en SMA han motivado una relajación en este planteamiento totalmente distribuido, ya que en muchas tareas reales con SMA heterogéneos se requiere de un cierto control centralizado que 'gestione' la coherencia del sistema. De hecho, los SMA con mayor aplicación práctica han optado por soluciones intermedias (Wang y de˜Silva, 2008).

- *Tamaño del equipo:* aunque pueda no parecer una característica relevante, en la bibliografía del campo podemos ver multitud de ejemplos que reflejan que los SMA con menor número de elementos suelen poseer estrategias de control centralizadas o mixtas, agentes heterogéneos y comunicación explícita (Ishiwaka *y otros*, 2003; Atanasova *y otros*, 2000). En cambio, los SMA con gran número de agentes (enjambres), suelen tener estrategias de control totalmente distribuidas y simples, además de ser homogéneos y no utilizar comunicación directa (que no es abordable en la práctica). (Zhao *y otros*, 2005).

- *Conocimiento:* dentro de esta misma categoría podemos incluir también otra clasificación que diferencia dos tipos de grupos de agentes. Agentes solipsistas o aislados y agentes sociales. Los primeros son aquellos que no tienen conocimiento de la existencia del resto de los agentes, y por tanto, sus decisiones están guiadas por percepciones del entorno únicamente. Estos agentes suelen presentarse en los SMA bioinspirados en los que cada agente dispone de un comportamiento muy simple y basado en interacciones locales (Kube y Bonabeau, 2000; Chantemargue y Hirsbrunner, 1999). Por otro lado, los agentes sociales son aquellos que sí detectan información

proveniente de otros agentes y pueden por tanto presentar, o no, comportamientos coordinados (Tomlin *y otros*, 1998; Takeda *y otros*, 2002). Un análisis más en profundidad de las capacidades de cada uno de los sistemas fue realizado por Batalin y Sukhatme (2004).

3.2.1.4. SMA y la resolución de problemas de ingeniería

Las principales aplicaciones de los SMA en la actualidad se centran en agentes software, donde podemos encontrar aplicaciones exitosas en subastas en Internet o comercio electrónico (Noriega y Sierra, 1998; Sandholm, 1999). Muchas de estas aplicaciones originales de agentes software, han sido probadas en agentes reales, mayoritariamente robots. Algunos de los trabajos más relevantes dentro de las aplicaciones de SMA a robots han sido el de Langle y Worn (2001) para estudiar la interacción humano-robot en entornos parcialmente conocidos, o el problema de autolocalización de grupos de robots mediante la sensorización de posición entre robots, datos de su propio movimiento y transmisión de datos entre vecinos que fue estudiada por Roumeliotis y Bekey (2002). Zlot *y otros* (2002) presentaron una aproximación mediante multirobot para el mapeo de zonas basada en balances económicos con respecto a la información obtenida durante la exploración. Encontramos también diversas aplicaciones, muy populares en los últimos años, centradas en el futbol con robots, donde los equipos de robots reales o simulados compiten uno contra otro (Kitano *y otros*, 1997; Mota y Reis, 2008; Kuo y Hsieh, 2008). Esta aplicación presenta unas características muy apropiadas para poner a prueba varias de las características básicas de aplicaciones reales: el entorno es continuo y dinámico, el comportamiento de los rivales es difícil de predecir, existe ruido e incertidumbre en las señales de los sensores, etc.

En el ámbito de la ingeniería, las arquitecturas multiagente han mostrado ser de interés en aplicaciones donde la toma de decisiones distribuida es una ventaja. Stephanopoulos (1989) generó un sistema autónomo de control de proceso, Brown (2002) estudió el diseño de plantas modelando una red de operadores, propietarios, ingeniería, y constructores obteniendo soluciones más robustas y flexibles. Julka *y otros* (2002) estudiaron la gestión de cadenas de suministro para el caso de una refinería.

Burmeister *y otros* (1997) se centraron en la aplicación de multiagentes en sistemas de transporte y control de tráfico (Lesser y Erman, 1980).

Algunas otras aplicaciones desarrolladas en SMA dentro del ámbito de la ingeniería se centran en proceso de diseño y manufactura de un producto (Darr *y otros*, 1996), estructura de una unidad almacenamiento y procesado de productos (Baker *y otros*, 1997b), procesos de mantenimiento industrial (Hu y Wellman, 1996; Gao *y otros*, 2009), fiabilidad de procesos con altos coste de paradas para mantenimiento, como el molido de acero o los problemas de redes de transporte (Ihara y Mori, 1984), control de robots articulados para realizar trabajos en cascos de barcos (Overgaard *y otros*, 1994), coordinación de grupos de robots (Farinelli *y otros*, 2004), distribución de potencia en una red eléctrica Jennings (1995); Jennings *y otros* (1992); Wittig (1992); Praça *y otros* (2003), sistemas de planificación de producción de piezas para Daewoo (Chung y Wu, 1997), etc.

Vemos pues, que el número de aplicaciones reales de los SMA es abundante, tal y como habíamos adelantado, y de hecho su popularidad sigue en aumento. Una vez introducida una visión general del campo de los SMA, de las características que definen los distintos tipos que podemos encontrar en la literatura y sus principales aplicaciones en ingeniería, pasamos a comentar las principales estrategias que han sido desarrolladas con el objeto de obtener comportamientos coordinados que persigan un determinado objetivo global.

3.2.1.5. Obtención de comportamientos coordinados

En el campo de los SMA, los problemas se resuelven mediante agentes que se coordinan directa o indirectamente. En estos sistemas, el control y la información están con frecuencia distribuidos entre los agentes, lo que reduce la complejidad de cada uno de los agentes y les permite trabajar en paralelo. En contrapartida, cada agente posee limitaciones de recursos (sensorización, actuación) lo cual hace más complicado resolver individualmente los problemas, y por tanto, se requieren estrategias que permitan resolverlos conjuntamente.

La investigación en SMA se centra básicamente en las interacciones entre agentes y en la adaptación de los agentes a nivel individual (Brads-

haw, 1997; Goldman *y otros*, 1994; Boutilier, 1996a,b; Claus y Boutilier, 1998). Un SMA bien diseñado es aquel que realiza una tarea global a través de las acciones de sus agentes. Según Jennings *y otros* (1998) los pasos de diseño de un SMA son los siguientes:

- Descomponer una tarea global en subtareas distribuidas de manera que sean tratables por cada agente.
- Establecer los canales de comunicación y sensorización que proporcionen la información necesaria a cada agente para llevar a cabo su tarea.
- Coordinar los agentes de modo que cooperen de cara a la tarea global, o al menos que no se generen conflictos para que puedan conseguir llevar a cabo sus tareas individuales.

Como vemos, el concepto común a los tres puntos anteriores es la *realización de la tarea*, que es clave en el diseño de un SMA. Esta es la razón principal por la que pasaremos a revisar las distintas aproximaciones existentes en el campo, de cara a conocer con detalle los resultados obtenidos por los investigadores en este sentido. Estas aproximaciones se pueden englobar en tres grandes grupos: las que se basan en reglas y modelos de coordinación establecidos por el diseñador, las que se basan en aprendizaje por refuerzo y las que se basan en técnicas evolutivas o de diseño automático.

Aproximaciones 'ad-hoc'

La obtención de soluciones a un problema distribuido que requiere coordinación entre los agentes es una tarea compleja. Por este motivo, son muy numerosos los trabajos que se basan en sistemas de reglas o modelos matemáticos fijos, diseñados 'a mano' y donde el énfasis se centra en la creación de protocolos de coordinación eficientes. Así, es común encontrar aproximaciones en el campo de los SMA donde la mayor parte del esfuerzo se concentra en la creación de protocolos de coordinación a priori, negociaciones complejas, formación de coaliciones, sistemas de subastas o estrategias basadas en contratación. Todas estas aproximaciones, al igual que ocurría con los trabajos originales de la IAD, requieren

de un gran conocimiento del problema concreto, son difícilmente generalizables, y además, adolecen de falta de escalabilidad y robustez.

A pesar de estas limitaciones, debemos comentar 3 grandes líneas de desarrollo en este sentido, donde los logros obtenidos a nivel práctico les confieren mucha relevancia a la hora de analizar todas las opciones posibles de obtención de comportamientos coordinados con un objetivo global: los SMA basados en modelos económicos, en teoría de juegos y en modelos bioinspirados.

- *Modelos económicos*: tal y como apunta Clark (1998), gran parte del campo de la economía teórica se ocupa de maximizar un conjunto de utilidades sometidas a ciertas restricciones tanto en los agentes individuales como en sus interacciones. Esta idea es la que ha servido a los investigadores del campo de los SMA para aprovechar los modelos económicos a la hora de obtener comportamientos que maximizan un beneficio global en base a elementos distribuidos.

  Las principales aportaciones en este ámbito se dan en el campo de la *Economía Computacional*, que a su vez se basa en la Ley del Equilibrio General (J. y Debreu, 1954) y en el Diseño de Mecanismos (Dash *y otros*, 2003) para gestionar los componentes de un sistema computacional distribuido. En esta aproximación se trabaja utilizando un *mercado computacional* que guía la interacciones entre los agentes. Este mercado se define como una estructura que permite a los componentes del sistema intercambiar información sobre el valor de los recursos, establecer estados de equilibrio e intercambiar dichos recursos. Este tipo de sistemas han sido usado para investigar economías reales o sistemas biológicos (Kephart *y otros*, 1998; Blume y Easley, 2002; Borkar *y otros*, 1998a,b), y también se han utilizado para el diseño de sistemas computacionales distribuidos (Buyya *y otros*, 2002; Ferguson *y otros*, 1988; Wellman, 1995).

  Un ejemplo de mercado computacional utilizado para un problema real de gestión global de recursos distribuidos, fue desarrollado por Huberman y Clearwater (1995), que creó un sistema de subastas para controlar la temperatura de un edificio. Otras aplicaciones reales

han sido llevadas a cabo en la gestión de la memoria en sistemas operativos (Harty y Cheriton, 1996), ubicación de circuitos virtuales (Ferguson *y otros*, 1989), extracción de ciclos de CPU en una red de ordenadores (Ellison, 1975; Waldspurger *y otros*, 1992) o en la predicción de opciones de futuro en mercados financieros (Potters *y otros*, 1998). Debemos destacar la aplicación llevada a cabo por Baum utilizando un mercado computacional para coordinar un gran número de agentes a resolver una variante del problema de los bloques (Baum, 1996; Baum y Durdanovic, 1999), además de un gran número de aproximaciones para la resolución de problemas de programación y ubicación distribuida de recursos (Wellman, 1993, 1994; Semret y Lazar, 1998; Kurose y Simha, 1989).

En resumen, vemos que los modelos económicos han servido de base a un gran número de aplicaciones distribuidas, pero en todas ellas la coordinación de los agentes presenta una gran dependencia de un sistema de comunicación central y de un mecanismo de control centralizado. Por ejemplo, el sistema investigado por Baum para el problema de los bloques, no solo requiere de un control centralizado del mercado, sino que además requiere del control centralizado de otros parámetros externos al mercado y de gran influencia en el resultado final. En este sentido, Tucker y Berman (1996) expusieron sus dudas respecto al funcionamiento práctico de estos sistemas y pusieron de relieve la gran cantidad de trabajo específico que se requiere aplicarlos a cada caso concreto.

Por otro lado, y ya más centrado en el enfoque que queremos seguir en esta tesis, debemos destacar que el concepto básico que soporta los mercados computacionales de existencia de un valor (dinero) que hay que maximizar de forma global, sí es generalizable e interesante, y como veremos más adelante, ha servido de inspiración para el diseñar el modelo energético de uno de los experimentos que hemos llevado a cabo.

- *Teoría de juegos*: la teoría de juegos es una rama de las matemáticas que está relacionada con las versiones formales de juegos, desde el ajedrez o el póker hasta el dilema del prisionero.

En su versión más sencilla, un juego puede ser descrito de la siguiente manera: tenemos dos o más agentes, también llamados jugadores, y cada uno de los cuales tiene un conjunto de posibles acciones que pueden tomar. En el caso de juegos finitos este conjunto de acciones es finito. Adicionalmente, cada agente posee una función de utilidad, denominada 'matriz de recompensas', que define una relación entre las acciones de cada agente y el valor de utilidad que se le asigna a cada una de ellas.

Los agentes seleccionan sus acciones en secuencia, es decir, una detrás de otra y se denomina el 'conjunto de información' a lo que cada agente sabe en relación a las acciones que fueron seleccionadas por los agentes anteriores. En lo que se denomina un juego multi-etapa, una vez que todos los agentes han actuado, se envía a cada uno de ellos la información correspondiente respecto a las acciones del resto y con esta información ellos deciden su próxima acción. Después de realizar sus acciones, los agentes reciben su recompensa, es decir, ven aumentado su valor de utilidad en función del la matriz de recompensas previamente definida.

Se define la estrategia de un agente como el criterio que sigue para seleccionar una acción determinada en función de la información que posee. Se denomina una estrategia pura si esta selecciona una acción concreta o estrategia mixta si se elige la acción siguiendo una determinada distribución de probabilidad. La solución de un juego, también denominado estado de equilibrio, es aquel conjunto de estrategias en las cuales todos los agentes se comportan de manera que su estrategia optimiza su función de utilidad para unas determinadas condiciones.

Un concepto básico, aunque no siempre se considere como tal, tras la teoría de juegos es que la solución al problema se alcanza mediante una interacción competitiva entre los agentes, es decir, a la hora de plantear una tarea objetivo en un SMA mediante la teoría de juegos, se debe realizar una labor cuidadosa de planteamiento formal del objetivo, de tal forma que la competición entre los

agentes conduzca a una mejora global en la realización de la tarea. Este planteamiento no siempre es posible.

En contrapartida existe la teoría de juegos cooperativa, donde los agentes pueden llegar a acuerdos entre ellos, y por tanto, coordinar sus tareas con un enfoque cooperativo (realmente no es cooperativo, sino que se basa en equilibrios de tipo económico). Esto permite a los agentes evitar quedarse 'atascados' en situaciones de equilibrio de Nash (Nash, 1950), es decir, situaciones en las que todos se beneficiarían si todos cambian su estrategia y ninguno la traiciona. Esto implica una forma de coaliciones entre los agentes participantes.

En resumen, el interés fundamental a la hora de aplicar la teoría de juegos en SMA reside en los estados de equilibrio que se logran, ya que de poder ser alcanzados con generalidad, serían de gran interés en multitud de tareas (Parsons y Wooldridge, 2002; Tuyls y Nowé, 2005; Dash, 2005). Pero el principal inconveniente que presentan es que establecer una matriz de recompensas que contemple todos los casos posibles en problemas reales es muy complejo, o incluso inabordable. Por otro lado, como hemos comentado anteriormente, la mayor parte de los juegos estudiados requieren conocimiento centralizado, normalmente de las matrices de recompensa de los otros jugadores, y con frecuencia, de la estrategia de cada uno de ellos. Por último, el tener que plantear la coordinación de los agentes a partir de juegos suele entrar en conflicto con la filosofía de la mayor parte de los trabajos en el campo de SMA, que buscan dicha coordinación a través de la cooperación, pero no de la competición.

Aún así, este es un campo con gran potencial, y actualmente son numerosos investigadores los que aplican aproximaciones basadas en aprendizaje por refuerzo o computación evolutiva para automatizar el diseño de SMA basados en teoría de juegos.

- *Modelos bioinspirados*: de entre la multitud de aproximaciones existentes para la obtención de comportamientos coordinados en SMA que tienen una inspiración biológica, debemos destacar aquellas

que que utilizan modelos basados en las colonias de insectos sociales, y que se engloban en el campo de la *Swarm Intelligence* o Inteligencia de Enjambre introducido inicialmente por Beni y Wang (1989).

En este tipo de sistemas, el comportamiento coordinado puede llegar a resolver problemas muy complejos, tales como la búsqueda de la ruta óptima, utilizando agentes muy simples. Este comportamiento surge a partir de una serie de reglas de interacción también simples, que son las que se han observado en los insectos reales Perez-Uribe *y otros* (2003), Tarapore *y otros* (2006). La coordinación se alcanza mediante un sistema de comunicación indirecta que modifica el entorno y que, como ya hemos comentado, se denomina *stigmergy*. En el caso más típico, el de las colonias de hormigas, esta comunicación se lleva a cabo mediante un rastro de feromonas que es dejado por las hormigas y que se refuerza con el paso del tiempo (Dorigo *y otros*, 1996).

El comportamiento coordinado que aparece en estos sistemas no es inferible a partir de la combinación simple de las capacidades de los agentes independientes. Cuando esto ocurre se habla de *emergencia* del comportamiento (volveremos a este concepto más adelante), y puede implicar que el ámbito de aplicación de estas aproximaciones sea limitado, ya el diseñador no tiene control ni conocimiento sobre cómo se ha resuelto la tarea objetivo, algo que en muchas aplicaciones reales no es válido.

Como conclusión, diremos que el concepto de estrategia de coordinación compleja a base de interacciones simples que es la base de la inteligencia de enjambre, sí es interesante para la técnica planteada en esta tesis.

Como hemos visto, las aproximaciones a la obtención de comportamientos coordinados basadas en diseños ‘ad-hoc’ más interesantes en el marco de esta tesis, se basan en la utilización de modelos sociales, económicos o biológicos. Aunque han resuelto numerosos problemas distribuidos de forma satisfactoria, para nuestro planteamiento simplemente

proporcionan ideas generales, como el uso de una medida global de beneficio que guíe el proceso de optimización o el uso de reglas de interacción simples a nivel de agente, pero no son válidas como paradigma de desarrollo por su falta de generalidad y escalabilidad.

Aproximaciones basadas en aprendizaje por refuerzo

Tal y como acabamos de ver, uno de los principales problemas asociados al uso de técnicas 'ad-hoc' para la obtención de comportamientos coordinados en SMA es la realización de un modelado específico para cada problema o configuración concreta. El planteamiento contrario para llevar a cabo esta tarea de diseño o configuración, se centra en que este proceso se realice de *forma automática* y de acuerdo a una serie de objetivos predefinidos.

El aprendizaje por refuerzo es un estrategia de adaptación para agentes perteneciente al campo del aprendizaje máquina consistente en proporcionar una recompensa o un castigo asociado a una acción concreta en función de si esa acción está alineada en mayor o menor medida en la dirección del comportamiento objetivo. Con esto se pretende que cada uno de los agentes adapte su comportamiento de modo que maximice la recompensa conseguida a largo plazo.

Debido a que en el aprendizaje por refuerzo no se requiere de un modelo explícito del entorno ni tampoco tener algún tipo de conocimiento que nos proporcione el mapeado de los conjunto entradas-salidas óptimas para cada controlador en oposición a las estrategias de aprendizaje supervisado, este tipo de proceso de adaptación se convierte en una herramienta muy adecuada para la adaptación de un SMA cuando no se disponga de demasiada información sobre el entorno o sobre los otros agentes, o cuando la complejidad del problema impida el conocimiento de dicha información.

Las estrategias centradas en aprendizaje para un agente simple se han centrado en mejorar la eficiencia del aprendizaje o el conocimiento proporcionado, o simplemente tratando la información de otro modo (Samejima y Omori, 1999; Wang y Usher, 2005). Por otro lado, dentro del

aprendizaje por refuerzo en SMA como conjunto, diferenciamos entre dos tipos de estrategias en función del nivel de interrelación entre agentes:

- Aquel que se lleva a cabo en sistemas de agentes solipsistas: en este caso, cada uno de los agentes se considera aislado, y por tanto, no tienen en cuenta en su estrategia ni detectan mediante sus sensores la existencia de otros agentes. Las aproximaciones solipsistas han sido aplicadas a multitud de problemas (Crites y Barto, 1996; Guenther *y otros*, 1997) pero, sin embargo, ninguna de estas permite un escalado eficiente. A medida que el número de agentes aumenta, los efectos de las acciones de otros agentes crean un ruido en las asociaciones de refuerzo que dificulta enormemente el aprendizaje.

- El que se produce dentro de SMA sociales: en este caso los agentes reciben información del colectivo, y por tanto, las asociaciones de recompensa que estos generen estarán influidas en mayor o menor medida por las acciones de otros agentes. Por otro lado, encontramos también un gran número de trabajos con sistemas sociales (Claus y Boutilier, 1998; Fudenberg y Levine, 1993; Hu y Wellman, 1996), en los cuales se utilizan conceptos de teoría de juegos que dan lugar a comportamientos globales estables. En general, esta aproximación requiere de bastante trabajo de diseño a priori para evitar fluctuaciones no deseadas.

Enmarcado en el análisis de aprendizaje por refuerzo para SMA, Wolpert y Tumer (2000) desarrollaron una serie de ideas con respecto al control de sistemas de inteligencia colectiva que son realmente interesantes para la obtención de comportamientos coordinados. Según Wolpert: *'Uno de los descubrimientos de la última década y que abre un gran área de investigación es el percatarse de cómo de útil puede resultar se capaces de controlar sistemas distribuidos descentralizados y hacerlo adaptativamente, con el mínimo conocimiento de el comportamiento dinámico a bajo nivel del sistema'*.

Estos autores desarrollaron el sistema de inteligencia colectiva COIN (COllective INtelligence), que es un SMA descentralizado en cuanto a la coordinación o comunicación y con una *función de utilidad global* que define las dinámicas de interacción objetivo. La principal contribución de este

trabajo es la de estudiar una estrategia para tratar el modo de generar un comportamiento determinado en un SMA descentralizado a través de sus funciones de utilidad.

La base del trabajo de Wolpert y Tumer (2000) es la de haber diseñado un procedimiento para extraer una *función de utilidad individual* para cada agente que posea dos características principales. El incremento en la utilidad individual de cada agente debe implicar un aumento en la utilidad global. El mayor o menor cumplimiento de esta regla se ha denominado el *nivel de factorización de la utilidad individual* y la capacidad de aprendizaje que representa el efecto de las acciones del resto de los agentes en la calidad de un agente. Si este efecto es muy grande el aprendizaje presenta mucho ruido y se hace más complicado, si este efecto es reducido cada agente ve reflejado en su valor de calidad solamente las consecuencias de sus acciones y esto repercute en una aprendizaje más directo.

De las aproximaciones basadas en aprendizaje por refuerzo debemos destacar el enfoque más general que proponen al estar basadas en un aprendizaje no supervisado, de modo que las estrategias de cooperación se adaptan de forma automática en base al problema. Además, las funciones de utilidad planteadas en este ámbito dentro del sistema COIN resultan muy adecuadas al planteamiento que se persigue en esta tesis por y serán especialmente relevantes en la técnica presentada.

Evolución

La opción para coordinar un SMA que analizaremos se basa en la evolución natural. El concepto de evolución presenta diferentes implicaciones en las diferentes disciplinas (Biología, Inteligencia Artificial, Informática, Ciencias Sociales, etc.), pero en todas ellas, el papel central lo juega la existencia de una *población de individuos*, que sufren un proceso de selección y posterior reproducción dando lugar a nuevas generaciones de individuos que son combinación de los anteriores.

En el caso de la computación evolutiva, la evolución representa un modelo simplificado de la evolución natural. En las aproximaciones más tradicionales, -algoritmos genéticos, estrategias evolutivas o programación evolutiva-, los individuos no juegan un papel activo en el proceso. Básí-

camente, funcionan como un almacén pasivo de un conjunto de características y el proceso evolutivo no es distribuido, sino que es controlado globalmente de manera externa a la población. En este tipo de sistemas, a medida que cada generación es evaluada en función de unas medidas de calidad que serán especificadas por el diseñador, se creará una nueva generación .

Sin embargo, en el caso de un SMA, el objetivo no siempre es evolucionar un conjunto de parámetros que codifican una solución a un problema determinado dentro de un contexto concreto, de modo que esa solución optimice algún criterio, sino que lo que se persigue es evolucionar toda una población de manera que esta genere un comportamiento que, ahora sí, optimice algún criterio establecido (Zhong *y otros*, 2004; McPartland *y otros*, 2005; Nolfi, 2005; Marocco y Nolfi, 2007). Esto supone que el paradigma de la evolución tradicional no puede hacer frente a este tipo de problemas a menos que, o bien la población sea homogénea, y por tanto, podamos convertir el problema en un evolutivo tradicional en el cual toda la población del SMA representa a un solo individuo cuyo código genético es el de todos y cada uno de los agentes, o bien la población es heterogénea y cada individuo del evolutivo representa a toda una población, algo que no es computacionalmente viable.

Con el objetivo de estudiar el modo en el que se podría utilizar la evolución para resolver este tipo de problemas, Agogino y Tumer (2008) clasificaron el modo de evolucionar un SMA en función de su grado de heterogeneidad. En esta clasificación se consideran a todos los agentes físicamente iguales y el criterio para establecer que dos agentes son distintos se basa en si su controlador es el mismo o es distinto. Se tienen 3 tipos de evolución posible:

- **Full-System Policy:** representa la primera opción que hemos comentado, donde se utiliza un SMA formado por agentes idénticos y una población de individuos homogéneos que representan diferentes versiones de un programa de control. Para evaluarlos, cada uno de estos programas de control se replica en todos los agentes, es decir, la estrategia de toma de decisiones es la misma para todos. El curso de la evolución se guía por una medida preestablecida de la calidad de la población, no siendo necesaria la creación de una

medida de calidad individual (a nivel de agente) pues se evalúan las poblaciones en conjunto.

- *Ventajas:* el espacio de búsqueda es más reducido en una población homogénea y el entorno es menos ruidoso, pues todos los individuos tienen en el mismo comportamiento durante toda la simulación. Por tanto, el espacio de búsqueda es más simple.
- *Desventajas:* las soluciones potenciales son mucho más simples, no puede existir una descomposición de la tarea en diferentes sub-tareas. El tiempo de evaluación de cada individuo es alto pues hay que simular a toda la población durante un número de pasos suficiente como para obtener una medida significativa de calidad. Si se precisa generar comportamientos diferenciados para llevar a cabo la tarea (empujar y tirar de un objeto, por ejemplo) se necesitan comportamientos más complejos que sean capaces de autoasignarse tareas en función del estado del entorno y del resto de los agentes.

- **Shared Population:** esta es una estrategia mixta en la cual una única población de controladores homogéneos es compartida por todos los agentes del SMA, y al inicio de cada simulación-evaluación cada agente elige un controlador. Los individuos del evolutivo son los distintos controladores, que son evaluados en función de su calidad tras la simulación, y así se genera una nueva población.

  - *Ventajas:* evitamos el problema de tener controladores que tengan que generar todo el espacio de acciones, y esto puede ayudar a generar comportamientos más complejos. La evaluación se realiza para un conjunto de controladores y no de manera global para toda la población, aunque se utilice una medida de calidad global. Esto permite evaluar combinaciones heterogéneas de controladores, y en principio, genera soluciones heterogéneas.
  - *Desventajas:* el uso de una única población de controladores hace que estos tengan que servir a cualquiera de los componentes del sistema, y la población puede tender con facilidad a una falta de diversidad que no serviría para comportamientos

que no fuesen relativamente sencillos. Además, esta aproximación resulta altamente dependiente del número de controladores que se evolucionen con respecto al número de agentes. El proceso de selección es muy dependiente de las elecciones de los agentes, y por tanto, el nivel de exploración es muy bajo.

- **Individual population:** cada agente del SMA tiene su propio controlador o población de controladores. Es decir, ahora tenemos una población heterogénea debido a que cada controlador es distinto al resto. En este sentido, encontramos dos aproximaciones: cada componente se evalúa con la misma función dé calidad global o cada componente se evalúa con una función de calidad individual específica. En la primera aproximación, todos los componentes reciben la misma recompensa, que corresponde al comportamiento del conjunto de agentes, y por tanto, la evolución de cada componente está guiada por la calidad del resto, con lo que no es posible realizar un mecanismo de selección que de preferencias a unos individuos frente a otros. En la segunda aproximación, que es el que ocupa la investigación de los trabajos en sistemas como COIN, y representa el paradigma que vamos a utilizar en esta tesis, cada individuo, y por tanto, cada controlador, poseen una valoración individual que permite realizar una selección natural, y en consecuencia, una mejora continua en la población de agentes. El problema que surge, y que es equivalente al de las aplicaciones en aprendizaje con refuerzo, es el de definir una función de calidad individual adecuada. Esta función de calidad debe conseguir que se produzca una valoración adecuada de cada individuo y guiar la evolución hacia una población con una mejor calidad global.

  - *Ventajas:* La población es intrínsecamente heterogénea, esto permite llevar a cabo tareas que requieran de comportamientos heterogéneos, además de aquellas que requieran de comportamientos homogéneos. La evaluación es individual por lo que es posible una evaluación descentralizada. Mediante la evaluación individual se puede controlar además de la dirección de la evolución la tendencia a crear especies dentro de la población.
  - *Desventajas:* el proceso de generación de las funciones de utilidad individual a partir de la utilidad global puede resultar com-

plicado en algunos casos. El proceso evolutivo es más complejo y por tanto requiere de más tiempo de análisis para extraer información a partir del mismo.

En resumen, debemos destacar que el uso de evolución en SMA continúa la idea presentada con el aprendizaje por refuerzo en cuanto a la automatización en la obtención de los comportamientos coordinados, y que es una idea central en esta tesis. Por otro lado, los principales desarrollos en este campo utilizan las funciones de utilidad privada y global que ya comentamos en el apartado anterior y que, de nuevo, muestran su idoneidad en sistemas automáticos.

3.2.1.6. Funciones de utilidad

Debemos destacar con respecto a las técnicas que hemos comentado para la obtención de comportamientos coordinados, que los trabajos que más importantes serán para su integración en el funcionamiento en el CeAS son aquellos desarrollados por Wolpert y Tumer (2000) en los que se estudia la generación de las funciones de utilidad privada a partir de la utilidad global de un modelo. En ellos se especifican dos características que deben poseer las funciones de utilidad privada, estas son la *factorización* y la *alineación*.

Se define la factorización (F_{g_i}) de una función de utilidad privada (g_i) con respecto a la utilidad global (G) como:

$$F_{g_i} = \frac{\int_z \int_{z'} u\left[\left(g_i(z) - g_i(z')\right)\left(G(z) - G(z')\right)\right] dz' dz}{\int_z \int_{z'} dz' dz} \tag{3.2}$$

siendo z' el estado que sólo difiere de z en el estado de la componente i y $u[x]$ como la función de paso unidad que es igual a 1 cuando $x > 0$.

Esto representa el nivel de alineación que existe entre esa utilidad privada y la utilidad global, es decir, que el aumento o disminución de la utilidad privada aumente o disminuya la utilidad global.

Se define la sensibilidad λ_{i,g_i} de una función de utilidad privada para un estado z como la variación en el valor de la utilidad privada de un agente provocada por las acciones de ese agente con respecto a la variación en

el valor de utilidad privada de ese agente debido a las acciones de otros agentes, esto es:

$$\lambda_{i,g_i} = E_{z'} \left[\frac{|g_i(z) - g_i(z - z_i + z_i')|}{|g_i(z) - g_i(z' - z_i' + z_i)|} \right] \tag{3.3}$$

siendo $E_{z'}$ el valor promedio tomado sobre z' y $(z - z_i + z_i')$ el estado del escenario cuando los efectos provocados por el agente i en el estado z han sido substituidos por los efectos del agente i en el estado z'.

Finalmente, con el objeto de maximizar esas dos medidas, los autores proponen una medida de utilidad privada denominada utilidad diferencial de la forma:

$$D_i = G(z) - G(z_{-i} + c_i) \tag{3.4}$$

siendo D_i el nivel de calidad global actual y $G(z_{-i} + c_i)$ el nivel de calidad global eliminando los efectos de las acciones del agente i.

Esta formulación básica será utilizada en el marco de la técnica que presentamos en esta tesis, como veremos posteriormente en los casos prácticos de aplicación.

3.2.1.7. Conclusiones

Como conclusión a este apartado centrado en la Inteligencia Artificial Distribuída, diremos que son múltiples los avances alcanzados a la hora de diseñar sistemas basados en agentes que se coordinan para resolver aplicaciones dinámicas descentralizadas. Las técnicas utilizadas en problemas reales han sido muchas y muy variadas, consecuencia de que en este tipo de sistemas el diseño es 'top-down', es decir, el énfasis del desarrollo recae precisamente en la creación de estrategias que favorezcan la coordinación, las comunicaciones, el control distribuido, etc., de manera que el sistema se dirija a la consecución del objetivo global.

Sin embargo, debemos destacar que la mayoría de las estrategias desarrolladas en aplicaciones reales de ingeniería para SMA, han sido diseñadas de manera específica para cada problema o tipo de problema. Esto supone una pérdida en generalidad y adaptabilidad, ya que cada

SMA está desarrollado para un funcionamiento dentro de un rango relativamente reducido. Tal y como hemos adelantado, nuestro objetivo fundamental va a ser el planteamiento de una técnica computacional que permita obtener de forma automática las estrategias de interacción en que definen el comportamiento global del sistema de modo que este comportamiento se ajuste a unos objetivos de diseño, con la premisa básica de que la técnica sea independiente del problema.

Del estudio de las aproximaciones utilizadas en el campo de los SMA no extraemos un método general para afrontar la resolución de problemas dinámicos descentralizados, sino una metodología para el planteamiento de las funciones objetivo que lleve al cumplimiento del objetivo global mediante el cumplimiento de los objetivos individuales de cada agente. Esta metodología se basa en la teoría de las funciones de utilidad privada y global, cuya aplicación concreta y resultados serán estudiados en detalle en la parte de desarrollo de esta tesis.

3.2.2. Sistemas Complejos

3.2.2.1. Introducción

Tras revisar las principales contribuciones dentro del campo de la Inteligencia Artificial Distribuida a la hora de obtener soluciones distribuidas guiadas por un objetivo a problemas dinámicos descentralizados, en este apartado pasamos a estudiar con detalle una disciplina que, en muchos ámbitos científicos, ha introducido una serie de nuevos enfoques a la hora de analizar los fenómenos de interacción no lineal en ámbitos como los biológicos, económicos, demográficos, etcétera. Estos estudios se enmarcan dentro de lo que se ha dado en llamar la ciencia de la complejidad o los Sistemas Complejos (SC).

Este campo proviene de dos fuentes, por un lado del estudio de las propiedades de los sistemas dinámicos no lineales, y por otro, del estudio de las implicaciones a gran escala de las interacciones locales a través de la mecánica estadística de las condiciones de criticalidad de un fenómeno. Ambas fuentes confluyen en la idea básica de que la simulación

computacional de estos fenómenos complejos nos permite obtener información sobre los mismos que de otra forma sería inaccesible (Goldspink, 2002).

Los SC, hasta hace pocos años, no constituían una disciplina académica, ni existían los departamentos de ciencias en ninguna universidad. Sin embargo, sí podemos encontrar, desde hace algún tiempo, algunos institutos de investigación y organizaciones profesionales dedicados a este campo. El más relevante es el instituto de Santa Fe (SFI), que fue fundado en 1984 en su mayoría por científicos del laboratorio nacional de Los Alamos y cuyo objetivo ha sido desarrollar la ciencia de los SC como un área de investigación independiente e interdisciplinar. En el SFI[8] se han llevado a cabo, desde entonces, investigaciones en inmunología, anticaos, vida artificial y optimización genética enmarcadas en el campo de los SC.

Otros ejemplos son el centro para SC de Illinois[9], que ha realizado investigaciones principalmente con aplicaciones de autómatas celulares, y la Sociedad para el Teoría del Caos en Psicología - que estudia la mente como un sistema complejo auto-organizado -, que es otra de las organizaciones que centran su investigación en esta disciplina.

A lo largo de este capítulo vamos a dar una visión general del campo de los Sistemas Complejos, una descripción de este tipo de sistemas y un estudio de sus características más relevantes. El objetivo es extraer una serie de conclusiones que nos permitan utilizar los conceptos que fundamentan esta disciplina para la obtención automática de comportamientos coordinados en Sistemas Multiagente que resuelvan problemas dinámicos descentralizados, objetivo básicos que estamos tratando.

3.2.2.2. Descripción de los Sistemas Complejos

Hasta la fecha, no existe una definición exacta y consensuada sobre lo que es un sistema complejo o sobre la complejidad. Pero sí podemos dar algunas características elementales sobre cómo se define comúnmente un sistema complejo (SC).

[8] http://www.santafe.edu/
[9] http://www.ccsr.uiuc.edu/

En líneas generales, los SC son aquellos sistemas formados por un gran número de componentes individuales similares que interactúan entre sí, y que como resultado de estas interacciones locales, modifican sus estados internos y provocan un comportamiento colectivo emergente, es decir, el comportamiento resultante del sistema no puede ser inferido mediante el comportamiento individual de cada componente sino que surge como resultado de las interacciones entre estos, y no está provocado por la existencia de un controlador central. En consecuencia, es muy difícil predecir la evolución futura del sistema, dado que la gran cantidad de interacciones y de elementos hacen muy difícil el estudio preciso de la dinámica del sistema tras un determinado intervalo de tiempo. Estos sistemas pueden ser individual y estructuralmente simples, pero esto no impide que puedan mostrar comportamientos complejos y dinámicos.

Ejemplos reales en los que vemos este tipo de comportamiento complejo son los cardúmenes, las manadas o los enjambres de insectos, que poseen un comportamiento conjunto claramente diferenciado a su comportamiento individual. Del mismo modo, una neurona por sí misma no puede realizar ninguna función realmente útil, y sin embargo, millones de neuronas interconectadas conforman un cerebro. También son ejemplos típicos de SC los sistemas de ecologías sociales, basados en el comportamiento de las sociedades a nivel macroscópico. En cuanto a ejemplos concretos llevados a cabo en simulación computacional, probablemente el sistema más ilustrativo de lo que es un SC, por conocido y simple, y que muestra propiedades emergentes como resultado de reglas de interacción simple, es el Juego de la Vida inventado por John Conway (Gardner, 1970).

Pasamos ahora a describir más en profundidad las principales propiedades que caracterizan este tipo de sistemas y que son de interés para su análisis y estudio en el marco de esta tesis.

Modelos

Tal y como mencionamos anteriormente, para resolver un problema de optimización, en primer lugar necesitaremos un modelo que represente las variables y sus interrelaciones, restricciones y función objetivo del problema real. Del mismo modo, para estudiar el comportamiento de un

sistema complejo, en primer lugar, necesitamos crear el modelo que define los elementos que intervienen en el mismo así como las interrelaciones entre estos elementos. Un modelo es una representación matemática simplificada de un sistema. En el sistema real hay ciertas características que son importantes, pero no todas deben ser incluidas en el modelo, sino sólo aquellas relevantes y que juegan un papel esencial en el fenómeno sobre el que se centra el problema a tratar. Si además queremos utilizar la simulación computacional para representar el modelo y llevar a cabo su optimización, necesitaremos una descripción mucho más exhaustiva: 'mientras que una buena simulación debería incluir tantos detalles como sea posible, un buen modelo incluirá los mínimos detalles que sea posible' (Boccara, 2003).

Emergencia

La aparición de la emergencia o de fenómenos emergentes se detecta tras el estudio de ciertas propiedades espacio-temporales que surgen espontáneamente a partir de interacciones entre los elementos de un sistema y que no son reducibles a las propiedades de sus constituyentes. Estas dinámicas, que aparecen en los SC, producen que, donde antes había un espacio desordenado y homogéneo, ahora se crean ciertas estructuras y orden, y esto se produce de manera espontánea debido a que, inicialmente, el sistema no se encuentra en un punto de equilibrio estable lo que provoca variaciones en su estado. Este tipo de estructuras y orden se representa en los Sistemas Complejos mediante propiedades que aparecen en escalas de tiempo y longitud mayores a las de las interacciones que las generan. Estas propiedades emergentes se han detectado y empezado a estudiar en diversos ámbitos científicos: la física, biología, economía, sociología, etc.

Históricamente, el concepto filosófico de emergencia es muy antiguo. Empezó a estudiarse en profundidad a finales del siglo XIX, pero incluso antes ya se conocen análisis diversos relacionados. En su significado más abstracto, el emergentismo sostenía que los fenómenos que se daban en la materia no podían reducirse a las propiedades de sus componentes fundamentales, lo que estaba en oposición al reduccionismo, que

sostenía que todo proceso podía reducirse a la suma de los efectos de los componentes del mismo.

En el campo de la Vida Artificial, que trataremos con detalle más adelante, se han estudiado numerosos ejemplos para ilustrar fenómenos emergentes, como los algoritmos de hormigas para encontrar rutas en grafos (Chen *y otros*, 2007; Yanovski *y otros*, 2001), las simulaciones de bandadas de pájaros de Reynolds (1987) o los autómatas celulares (Wolfram, 1984), entre otros.

Desde un punto de vista computacional, suele interpretarse el concepto de comportamiento emergente como aquel que no es generado por el comportamiento de las partes constituyentes de un sistema, sin embargo, este enfoque no es totalmente correcto. El comportamiento colectivo está, sin duda, contenido completamente en el comportamiento individual de sus constituyentes, si estos se estudian en el contexto en el que se desenvuelven. Lo que ocurre es que este comportamiento colectivo no es fácilmente entendible a partir del comportamiento de sus partes.

Podemos distinguir entre emergencia local, aquella donde el comportamiento emergente aparece en una pequeña parte del sistema y la emergencia global, donde el comportamiento colectivo se refiere a todo el sistema. Un ejemplo sencillo para entender la diferencia se da en termodinámica: el sistema más sencillo para analizar es un gas. Dos propiedades emergentes del gas son la presión y la temperatura. Las partículas individuales se describen mediante sus valores de posición y velocidad. La presión y la temperatura surgen solo cuando hablamos de grupos de partículas como combinación de la posición y la velocidad individual de estas. Y aunque estas dos propiedades son emergentes, su alcance es muy limitado, serían propiedades emergentes locales. Si tomásemos una muestra de ese gas y los separásemos del resto, aún así podríamos medir los mismos valores de presión y temperatura. En cambio, en una propiedad emergente global, si eliminásemos una parte del sistema, el comportamiento sería distinto. Esto sería un comportamiento emergente global.

En sociología, la emergencia de estructuras ordenadas fue estudiada por Haken (Haken, 1986). Este investigador comprobó que, al finalizar un evento popular, por ejemplo un mitin político, una vez que todos los asis-

tentes dejan de estar concentrados en un punto, comienzan a interactuar de manera local y pueden encontrarse, en un momento dado, grupos grandes de personas junto con grupos pequeños e individuos aislados. Lo que Haken encontró fueron ciertas relaciones entre la frecuencia de aparición de cada uno de los grupos de personas y el tamaño de estos grupos, de acuerdo a lo que se denomina una ley de potencias y que luego veremos que está presente en muchos otros fenómenos, como por ejemplo, en el comportamiento de las termitas (O'Toole *y otros*, 1999).

Dentro de los insectos sociales, otro ejemplo de interés es el comportamiento de las hormigas, cuyos individuos adquieren un comportamiento caótico cuando están aisladas y un comportamiento sincronizado y ordenado cuando están llevando a cabo interacciones sociales (Cole, 1991). Este proceso, llamado facilitación social, produce un orden emergente donde no lo había anteriormente. De hecho, se ha estudiado que las colonias de hormigas poseen un comportamiento que oscila con una frecuencia fija entre un estado de reposo y un estado de gran actividad.

Estabilidad

En la mayoría de los SC nos encontramos con sistemas inestables, es decir, se mantienen en un punto de equilibrio muy sensible a variaciones. Por tanto, cualquier modificación puede afectar al comportamiento de todo el sistema. De hecho, la mayoría de estos sistemas se caracterizan por la intermitencia entre estados de orden y desorden.

Esta es una propiedad muy relevante que se debe tener muy en cuenta cuando se desarrolla una simulación de SC para la obtención de una estructura estable que permita solucionar algún tipo de problema. La ventaja de esta serie de fluctuaciones es que permiten adaptarse con facilidad a nuevas situaciones y no converger a una solución fija, y por otro lado, la desventaja es que debemos extraer las soluciones antes de que el sistema fluctúe y pase al estado inestable o desordenado, y encuentre una solución que puede no ser la esperada.

Complejidad

Otro concepto fundamental en la teoría de los SC es la medida cuantitativa de cómo de complejo es un sistema. Existen tantas definiciones de complejidad como campos en los que esta se estudia. En cierto modo, el sentido común parece indicarnos qué es lo que se entiende por un sistema complejo, pero a la hora de formalizarlo o definirlo surgen diferencias entre unas y otras aproximaciones. En la Teoría de la Información, la complejidad de un sistema es la cantidad de información que se requiere para describirlo y depende también del nivel de detalle requerido para esta descripción. En un sistema físico, donde especificamos las posiciones y el momento lineal de cada partícula, esta complejidad puede tomarse como proporcional a la entropía del sistema. En el ámbito de la algoritmia, la complejidad utilizada es la complejidad de Kolmogorov (Solomonoff, 1960; Kolmogorov, 1965), que dice que el nivel de complejidad de un sistema es equivalente al tamaño del mínimo algoritmo que pueda codificar una secuencia que describa al sistema. En definitiva, no se ha propuesto todavía una medida de complejidad consistente, sino una serie de ellas, adaptada cada una a los diferentes ámbitos en los que se estudia.

Dinamismo

Cualquier sistema complejo es un sistema dinámico en la medida en la que este está cambiando, evolucionando o adaptándose. Un sistema dinámico se define como un sistema que presenta cambios en su estado a lo largo del tiempo. Este comportamiento variable queda determinado por los elementos y sus relaciones y por los límites del sistema. Con esto se pueden elaborar los modelos para intentar representar la estructura de este sistema. Dentro del concepto de sistema dinámico, podemos incluir los siguientes elementos:

- Un espacio de fases, donde cada elemento representa un posible estado del sistema.
- El tiempo, que puede ser discreto o continuo.
- Una ley de evolución, que es la que determina el estado en un instante t, a partir de los estados de los instantes previos.

El nombre, sistema dinámico, proviene por extensión a partir del nombre de las ecuaciones que gobiernan el movimiento de un sistema de partículas. Hoy día se usa como un sinónimo de un sistema de ecuaciones no lineal. En función de si el tiempo variable pudiese ser considerado contínuo o discreto , las ecuaciones constituirán un conjuntos de ecuaciones diferenciales o ecuaciones de diferencias finitas.

Sistemas caóticos

Otra de las características que podemos encontrar en un sistema complejo y que se estudian en profundidad en este campo es el caos. Tradicionalmente, los científicos realizaban la clasificación de sus sistemas según lo predecibles o no que estos pueden ser. Esto es, los sistemas podían ser determinísticos o probabilísticos, en función de si seguían reglas fijas o si seguían algún tipo de reglas estadísticas. Pero esta clasificación se demostró no adecuada con el descubrimiento de determinados sistemas dinámicos que siguen reglas deterministas pero que poseen un comportamiento muy complejo. Es decir, las reglas de interacción local son simples, pero el comportamiento global, es inesperado y no predecible. Se les denominó sistemas caóticos.

Otra de las peculiaridades de un sistema caótico es su alta dependencia sobre las condiciones iniciales. Una pequeña variación en dichas condiciones, puede generar un comportamiento muy diferente aún con condiciones externas muy similares, y es por esto que es muy difícil generar predicciones para estos sistemas. Un ejemplo natural de este tipo de sistemas son las condiciones climáticas de una zona. Aún cuando es posible medir con muy elevada precisión las variables que intervienen en los procesos meteorológicos (temperatura, presión, humedad, etcétera), es muy difícil predecir el estado de estas variables mas allá de un cierto tiempo. Otros ejemplos de sistemas caóticos son los fluidos en régimen turbulento, el mercado de valores, la propagación de enfermedades, las arritmias del corazón, la red neuronal del cerebro, etcétera.

Bioinspiración

La mayoría de los sistemas que nos rodean, son complejos. Y esto ha sido lo que ha inspirado la investigación que trata de descubrir qué tipo

de estructuras subyacen bajo el orden de tan diversos fenómenos en la naturaleza.

En la naturaleza podemos encontrar numerosos SC, desde determinadas reacciones químicas, hasta procesos sociales y culturales. Se ha visto que existe una tendencia natural a mostrar estructuras compuestas de elementos sencillos que interactúan y que se van organizando en distintos niveles de complejidad, por tanto los SC que estudiamos no son casos especiales, o disposiciones concretas de grupos de elementos, sino que predominan en la estructura espacio-temporal de todo lo que nos rodea y es por eso que podemos encontrar manifestaciones de ellos en la mayoría de los fenómenos observables. Por otra parte, y como ya hemos dicho, esta gran cantidad de fenómenos no representa una gran cantidad de comportamientos dinámicos, sino que existen propiedades comunes con independencia de los detalles concretos de cada contexto. Esto hace que podamos estudiar, identificar e incluso simular este tipo de comportamientos colectivos emergentes y extraer conclusiones de los mismos que pueden ser extrapoladas a diferentes ámbitos.

Interacción con el entorno

Cualquier sistema se encuentra situado en un entorno y este puede, en mayor o menor medida, afectar a la dinámica del mismo. En relación con la interacción con el entorno, podemos realizar una clasificación de los diferentes sistemas. Se denominan sistemas cerrados a aquellos que apenas se ven afectados por las condiciones externas, es decir son poco permeables. Por otro lado, los sistemas muy permeables, es decir, que se ven altamente afectados por las condiciones del entorno, se denominan sistemas abiertos. Entre uno y otro extremo se pueden encontrar, por supuesto, casos intermedios.

Los sistemas que nos resultan de más interés en esta tesis son los sistemas abiertos, pues pueden ser alterados o modificados mediante acciones externas, y a su vez, estos producen efectos en el entorno en el que se desenvuelven. Dentro de los sistemas abiertos encontramos los sistemas adaptativos, que son los que reaccionan y se adaptan al entorno, y los no adaptativos, que son aquellos que reaccionan pasivamente al entorno. Dentro de esta nueva clasificación, los sistemas en los que nos

centraremos son los sistemas adaptativos, por su capacidad para cambiar y adaptarse al medio en el que se encuentran, y es esta capacidad de adaptación a la que intentaremos sacar provecho para generar estados estables.

Control

El control de un SC es uno de los temas más tratados y más relevantes dentro de este campo. Tras el estudio de las características y estructura de estos fenómenos, surge de inmediato el interés en ser capaces de producir que las dinámicas de un sistema tiendan hacia uno u otro estado o comportamiento, en función de algún objetivo o interés particular.

Esto posee además un doble interés, por un lado el de modificar las dinámicas de comportamiento de un sistema real. Es fácil hacerse una idea de la utilidad de modificar comportamientos económicos, sociales o incluso fenómenos físicos como los meteorológicos actuando en determinadas propiedades del sistema o mediante la variación de las condiciones del entorno. Y por otro lado, el de poder generar SC adaptativos mediante simulación para estudiar sus propiedades, predecir sus variaciones ante determinadas modificaciones y extrapolarlas a los sistemas reales. También para generar comportamientos que, aunque no estén basados en comportamientos reales concretos, sí exhiban propiedades que nos puedan ser de interés (puntos de equilibrio o soluciones estables a determinados problemas).

Según Zundong *y otros* (2008), para controlar un sistema se requieren tres ingredientes: un modelo del sistema, unas dinámicas objetivo y unos requerimientos de control. El modelo se utiliza para obtener una predicción del estado del sistema experimental. Luego, esta predicción se compara con las dinámicas objetivo, y finalmente, se calcula la acción de control necesaria. Una elección típica para los requerimientos de control es la imposición de que la diferencia entre el estado predicho y las dinámicas objetivo sean nula o lo más baja posible.

Tras haber revisado las principales propiedades de los SC, podemos concluir que en este campo se estudian ciertas características que nos resultan básicas en el tipo de problemas que queremos tratar en esta tesis:

existencia de una gran número de elementos e interacciones entre ellos en el modelo que define un problema, dinamismo intrínseco de todos los sistemas y alta complejidad. Es decir, el campo de los SC considera como propiedades básicas de los problemas que trata, aquellas que hemos detectado como las más características de los problemas reales que queremos tratar en esta tesis, y que típicamente se 'relajan' en otros campos, por ejemplo, en la Inteligencia Artificial Distribuida.

Por este motivo, hemos decidido analizar con detenimiento los desarrollos llevados a cabo en este campo centrados directamente en la resolución de problemas, dejando de lado aquellos de carácter más teórico-analítico. Así, en lo que resta de apartado nos vamos a centrar en un subcampo de los SC, la Vida Artificial.

3.2.2.3. La Vida Artificial como un sistema complejo

Uno de los campos que se enmarcan dentro de las ciencias de la complejidad es la Vida Artificial (VA). En efecto, la vida constituye uno de los ejemplos más evidentes de estructuras muy complejas que se generan como combinación de estructuras más simples. Se trata pues, de una propiedad emergente de la interacción entre un conjunto de elementos y de la dinámica propia del sistema, y cuyo nivel de organización se sitúa al borde del caos (Langton, 1997).

La VA intenta generar este tipo de interacciones entre elementos simulados relativamente simples y con un control local descentralizado y no programado. Simplemente se dota a los comportamientos individuales de un conjunto de reglas sencillas y se dejan evolucionar.

Nuestro interés en la VA reside en que, dentro de este campo, se estudian los SC abiertos adaptativos que ya hemos comentado, mediante los cuales se pueden producir una serie de cambios en las interacciones entre elementos con el objeto de que el sistema se adapte y busque nuevos comportamientos estables capaces de realizar tareas que el comportamiento individual no sería capaz de realizar.

Además, es en la Vida Artificial donde podemos encontrar *SC con evolución natural*, es decir, donde los elementos que constituyen el sistema

complejo interaccionan mediante las reglas básicas de la evolución de las especies. Al ser este enfoque evolutivo una de las bases de la técnica computacional que se plantea en esta tesis, el campo de la Vida Artificial requiere de un tratamiento más detallado.

Introducción

Podríamos situar el origen de la VA a finales de los años sesenta, cuando Conway creó su 'Juego de la Vida' (Gardner, 1970), publicado por primera vez en el número de octubre de 1970 de la revista Scientific American. Este consistía en un autómata celular[7] en el que las celdas podían tener dos estados, vivo o muerto, y que tenía un conjunto sencillo de reglas de interacción local. Este entorno simple consiguió captar la atención de los investigadores de la época, e incluso hoy en día es utilizado para mostrar fenómenos de emergencia y variaciones del nivel de complejidad en un sistema dinámico discreto sencillo. Esto es debido a la sorprendente multitud y variedad de comportamientos combinados que aparecen en el Juego de la vida mediante el uso de unas reglas de interacción simples.

Sin embargo, el nacimiento del término Vida Artificial y el inicio del estudio del mismo se sitúa realmente más tarde, en 1987, cuando Langton reunió a más de 160 expertos de diversas disciplinas y tuvo lugar la primera conferencia sobre VA en el laboratorio de Los Alamos, en nuevo Mexico. Según Langton, la Vida Artificial es: 'el estudio de los sistemas desarrollados por el hombre que muestran comportamientos característicos de sistemas vivos naturales'.

Por otro lado, según la revista Artificial Life: 'la VA investiga los aspectos científicos, ingenieriles, filosóficos y sociales de nuestra habilidad tecnológica para sintetizar, partiendo de cero, comportamientos similares a la vida en ordenadores, máquinas, moléculas u otros medios alternativos. Como aspectos principales de estudio en esta disciplina se encuentran la jerarquía de la organización biológica, incluyendo estudios sobre el origen de la vida, auto-ensamblaje, crecimiento y desarrollo, dinámica ecológica

[7] Un autómata celular es un modelo discreto cuyo estudio pertenece principalmente al campo de las matemáticas y de teoría de la computación, y que consiste en una malla regular de celdas cada una de las cuales posee un numero finito de estados que varían en función de una serie de reglas que implican a un número finito de celdas vecinas.

y evolucionista, comportamiento de robots y animales, organización social y evolución cultural' Artificial Life Journal (2009).

Si buscamos más definiciones formales del campo de la VA, encontraremos varias tendencias, pero de todas ellas se puede extraer un cierto consenso conceptual. A continuación mostraremos las principales, diferenciadas en base a la propiedad central que se considera en cada una de ellas:

Magnenat y Thalmann utilizan un enfoque amplio y computacional: 'La VA hace referencia a todas las técnicas que intentan recrear organismos vivos y criaturas mediante un ordenador' (Thalman y Thalmann, 1994).

Maynard Smith basa la existencia de la vida en la capacidad de evolución: 'La vida debería ser definida por la posesión de esas propiedades que se necesitan para asegurar la evolución por selección natural' (Smith, 1982).

Adami y Brown (1994b) da una definición para la vida basada en propiedades termodinámicas: 'La vida es una propiedad de un conjunto de unidades que comparten información codificada en un sustrato físico y que, en presencia de ruido, se maneja para mantener su entropía significativamente más baja que la entropía máxima del conjunto, en escalas de tiempo que exceden la escala temporal natural de decaimiento del substrato (portador de información) en muchos órdenes de magnitud'.

Según Farmer y Belin (1992), para poder considerar que un proceso entra dentro de lo que se denomina VA debe presentar las siguientes características: sus elementos deben realizar almacenamiento de información o auto-representación, interacciones funcionales con el entorno y/o otros elementos, interdependencia de sus partes (no divisibilidad), estabilidad ante perturbaciones del entorno, capacidad para evolucionar, crecimiento o expansión, auto-reproducción y debe considerarse la VA como un patrón en el espacio-tiempo más que como un objeto material concreto.

Finalmente, aunque es difícil que vayamos a encontrar una definición concisa que acote el alcance de lo que inscribimos dentro de VA, sí podríamos decir con cierta seguridad que la VA representa una *aproximación desde*

el punto de vista de la síntesis al estudio de sistemas dinámicos bioinspirados. Es en el marco de la síntesis donde la VA puede proporcionar una utilidad práctica dentro del ámbito de la ingeniería. Tal y como escribió Langton (1997) en el prólogo a su libro, 'la naturaleza ha descubierto soluciones ingeniosas a muchos problemas complejos de ingeniería, problemas que nosotros no hemos sido capaces de resolver mediante nuestros métodos de ingeniería tradicionales [. . .]. Podemos tomar prestado el primer método de ingeniería que la naturaleza usó para alcanzar esas ingeniosas soluciones: la evolución. Mediante la síntesis de los mecanismos que subyacen en el proceso evolutivo en ordenadores y en otros medios no-biológicos, podemos descubrir soluciones a problemas de ingeniería que se han resistido durante mucho tiempo a nuestras aproximaciones tradicionales'.

Tendencias dentro de la Vida Artificial

La VA distingue dos tendencias principales: la *VA débil* y *la VA fuerte*.

Estas dos posiciones responden en origen a diferentes enfoques filosóficos y están relacionados con las dos interpretaciones que podemos atribuir al término *artificial*. Consideramos artificial todo aquello no real o falso pero también aquello hecho por la mano del hombre.

La VA *débil* propone que la VA sirve para aprender y recrear modelos sintéticos de procesos relacionados con organismos vivos en ordenadores u otros medios artificiales, pero no defiende que estos modelos sean una forma de vida por sí mismos, es decir, es un modelo no real que imita a otro real. En estos trabajos el objetivo es que, mediante la recreación de modelos naturales concretos como la evolución por ejemplo, podemos, por un lado, mejorar nuestro conocimiento sobre estos procesos y su relevancia en el mundo biológico, y adicionalmente, de un modo más práctico, podemos usar estos procesos para inspirar algoritmos de búsqueda u optimización. La mayoría de los trabajos que incluimos en VA en la actualidad se inscriben dentro de esta tendencia, que también se denomina la *tendencia soft*, pues el trabajo principal se basa en simulaciones por ordenador de procesos bioinspirados. Algunos de los trabajos enmarcados en esta tendencia más representativos han sido los llevados a cabo en el estudio de los comportamientos de insectos sociales para generar

algoritmos de búsqueda (Dorigo *y otros*, 1996; Eberhart y Shi, 2001).

La VA *fuerte*, por otro lado, defiende que las recreaciones de VA que poseen una serie de características inherentes a la vida son en sí mismas muestras de comportamientos vivos reales, independientes de la materia en la que se desarrollen. Se denomina también la *tendencia hard*, pues el trabajo se ha materializado tradicionalmente en simulaciones con robots, aunque a mayores ha surgido también una amplia variedad de investigación que podríamos incluir también en este subcampo, incluyendo a los sistemas autoorganizados, evolución open-ended, modelos de autopoiesis y hardware evolutivo entre otros. Sin embargo, es minoritario el trabajo publicado en VA que adopta de forma explícita la tendencia fuerte. Dos de los investigadores más representativos de esta tendencia son Langton y Ray que proclaman la vida como independiente del contexto en el que se desarrolla.

Bedau (2003) sitúa también los trabajos que buscan sintetizar sistemas vivos por medio de substancias bioquímicas dentro de una tercera tendencia denominada VA *húmeda*, pero nosotros no consideraremos este nivel de detalle en esta tesis.

Aunque nuestra investigación se centra en obtener un algoritmo de optimización, su inspiración recae en los trabajos inscritos en la tendencia *fuerte*. Sin lugar a dudas, nuestro objetivo no son las consideraciones respecto al concepto de la vida, pero el tipo de evolución que consideramos de interés para extraer características y estructuras que poseen los sistemas vivos reales de cara a obtener soluciones a problemas reales son aquellas sobre las que han trabajado investigadores como Tom Ray o Christopher G. Langton, las cuales se enmarcan en esta tendencia.

De un modo más concreto, las simulaciones de VA muestran un tipo de evolución particular que difiere de otras aproximaciones bioinspiradas. Estas simulaciones han sido desarrolladas originalmente en un intento de modelar con fidelidad el proceso de la evolución guiada por la selección natural, para estudiar y aprender sobre el proceso real, o simplemente con el objetivo de generar una forma de Vida Artificial tal y como se postula en el marco de esta tendencia. Por el contrario, otras técnicas evolutivas, como los algoritmos genéticos o la evolución diferencial por ejemplo, han

sido desarrolladas con un objetivo práctico claro, el de servir de algoritmos de búsqueda o optimización. Nuestro punto aquí es que consideramos que las evoluciones en VA presentan una serie de ventajas respecto a otros evolutivos que son las que pretendemos explotar.

En el siguiente apartado estudiaremos con detalle el tipo de evolución abierta que se utiliza en los sistemas de Vida Artificial, y que es una de las bases conceptuales de la técnica presentada en esta tesis.

Evoluciones open-ended

A partir de la evolución en entornos de VA y sus características se acuña el término evoluciones open-ended (Ruiz-Mirazo *y otros*, 2004; Bianco y Nolfi, 2004), para referirse a aquellas evoluciones en las cuales sus componentes continúan evolucionando de forma permanente, sin detenerse al alcanzar algún tipo de óptimo o posición estable. Se asocia este proceso evolutivo a un aumento gradual de la complejidad de los individuos dentro de un entorno.

Tal y como hemos visto, la mayoría de los sistemas evolutivos artificiales se han diseñado para ser herramientas de optimización, donde la evolución está guiada por una función de calidad orientada a un criterio determinado. Los individuos van hacia picos de calidad predefinidos y normalmente estáticos, y cuando los alcanzan, la población se acumula en esas zonas realizando una explotación a nivel más fino de las mismas.

Destacamos dos procesos evolutivos que se enmarcan dentro de lo que denominamos evolución open-ended, y que en oposición a otro tipo de evolutivos, presentan un comportamiento menos predeterminado. En primer lugar están los procesos *co-evolutivos*, en los que de una u otra forma el éxito de los miembros de una población depende de otra población co-evolucionada. Dada su relevancia para el sistema que se plantea en esta tesis, en el siguiente subapartado entraremos más en profundidad en lo que denominamos co-evolución.

El segundo tipo de modelos evolutivos (Conrad y Pattee, 1970; Packard, 1989; Ray, 1991; Adami y Brown, 1994b; Holland, 1995), han ido más allá en la idea de funciones de calidad extrínsecas, modelando la

evolución hacia algún tipo de meta de alto nivel. En estos sistemas, los individuos compiten por uno o más recursos limitados que necesitan para sobrevivir y propagarse, y por tanto, incluyen también procesos co-evolutivos. El hecho de que los recursos sean limitados induce una selección natural intrínseca. Este tipo de sistemas representan más a una evolución open-ended, porque ellos no evolucionan hacia ninguna meta predefinida, sino que están siendo seleccionados por su habilidad para conseguir recursos, y esta habilidad es medida en comparación a otros individuos de la población. Por tanto, la calidad de un individuo cambia a medida que nuevos individuos nacen o mueren. Este escenario es equivalente al presentado por Van˜Valen (1973) con su hipótesis de la reina roja para cambios evolutivos indefinidos en sistemas biológicos.

Para promocionar la evolución open-ended en aras de explotar sus capacidades potenciales, algunos miembros de la comunidad de VA han enfatizado la importancia de estudiar sistemas en los que los individuos son parte de un entorno en el que viven otros individuos (Ray, 1991; Arthur, 1994; Bedau, 1998)). Pero, sin embargo, no han aparecido muchas consideraciones teóricas que guíen el diseño de los sistemas comentados, cómo construirlos o cómo permitir que interactúen los elementos (Taylor, 2001).

La co-evolución

Un proceso co-evolutivo es aquel en el incremento en la calidad tiene su origen en que una población cambia sus características para intentar obtener ciertas ventajas con respecto a otra población con la que está compitiendo. Una vez que esto ocurre, la segunda población reacciona buscando una mejora para poder disponer de una cantidad de recursos suficiente como para sobrevivir. Como resultado, un incremento en la calidad de una población lleva a un decremento de la calidad de la otra, y por tanto, ninguna de las poblaciones consigue ser, en relación a los competidores, mejor con el paso del tiempo. Sin embargo, ambas poblaciones han evolucionado con respecto al entorno.

La primera referencia de la que tenemos conocimiento con respecto a efectos de evolucionar poblaciones que compiten entre ellas es el trabajo del biólogo evolucionista Van˜Valen (1973), y se basa en lo que se da a

conocer como el 'Efecto de la reina roja' dentro de la investigación en evolutivos, y que tiene su inspiración en un fragmento de un libro de Dodgson y Lewis (1871).

En el mundo real, esta competencia se encuentra en muchos organismos vivos que compiten entre sí, como depredadores y presas, anfitriones y parásitos y otros organismos que compiten por recursos limitados. El ejemplo más estudiado y más fácilmente entendible es el de la co-evolución entre depredadores y presas, donde el único modo en el que los depredadores pueden compensar una mejora de las presas -por ejemplo, conejos que corren más rápido- es incrementando su eficiencia - por ejemplo, zorros corriendo más rápido-. De este modo, con el paso de las generaciones, cada una de las poblaciones va siendo más eficiente a la hora de huir o perseguir a la otra población, logrando al cabo de las generaciones una mejora absoluta - corren más rápido- pero no una relativa.

Otro ejemplo ilustrativo de co-evolución competitiva entre miembros de la misma especie que nos lleva a un descenso en la calidad absoluta es el siguiente: las agrupaciones de árboles en un bosque tienden a competir por la cantidad de luz solar que reciben. El modo de obtener una mejora relativa al resto de los individuos, y por tanto obtener más cantidad de luz, consiste en crecer un poco más alto que el resto de la población. Sin embargo, esto fuerza al resto de los árboles a crecer también para conseguir la cantidad de luz que necesitaban para subsistir. El proceso se repite y los árboles son cada vez más altos, pero la cantidad de luz que reciben sigue siendo la misma. Como consecuencia, su balance real tras la evolución es negativo. Necesitan más recursos porque son más altos, pero la cantidad de luz que reciben sigue siendo la misma, es decir, la población en sí es más ineficiente.

Consideraciones relativas a este fenómeno son fundamentales en la aplicación de evoluciones open-ended como técnicas de optimización, pues la optimización aplicada a nivel individual puede no llevar a un optimo a nivel global.

En cuanto al uso de la co-evolución en simulaciones de VA debemos decir que a principios de los 90, la co-evolución se presentaba como un campo atractivo para la investigación. Las ventajas que se destacaban

eran que, por un lado, proporcionaba un método para evitar el estancamiento de la búsqueda en máximos locales, y por otro, que conseguía mejores soluciones que evolutivos clásicos debido a las dinámicas y restricciones cambiantes del entorno.

Las primeras investigaciones sobre co-evolución fueron llevadas a cabo por Ray (1991) con el desarrollo del simulador Tierra, Sims (1994) que simuló lo que él denominaba criaturas virtuales y que evolucionaron jugadores del tres en raya.

Hillis (1990) co-evolucionó una población algoritmos que pretendían mejorar la eficiencia de la ordenación, frente a una población de listas para ordenar cuyo objetivo era evitar ser ordenadas. Los algoritmos obtenidos de este modo fueron de hecho mejores y más rápidos que los obtenidos utilizando un conjunto fijo de casos.

En 1999, Floreano *y otros* (1998) desarrollaron una aproximación más enfocada hacia la robótica autónoma. Llevaron a cabo un experimento donde la calidad de cada individuo se probaba en un robot real que competía con otro en un modelo realista de simulación. La evolución del control de generación en generación sí era llevada a cabo en simulación y ellos pudieron comprobar que en su evolución se reproducía el efecto de la Reina Roja.

Más tarde, Urzelai y Floreano (2001) llevaron a cabo un segundo experimento para estudiar cómo evolucionaba la adaptatividad. Los trabajos de Floreano y Nolfi han supuesto, por tanto, un paso importante hacia la explicación de las dinámicas de la co-evolución aunque aún hoy día quedan algunos cabos sueltos.

Aunque nuestra aproximación podría enmarcarse más en la línea de evoluciones open-ended de ecosistemas artificiales, esta comprende a procesos co-evolutivos implícitos, y por tanto, la mayoría de las consideraciones relativas a la co-evolución son relevantes para nuestro trabajo.

3.2.2.4. Evolución asíncrona y situada

Otra característica que encontramos implícita en las evoluciones de Vida Artificial es aquella que implica que el proceso evolutivo se realiza de forma situada, es decir, que las interacciones que conducen al proceso de reproducción están supeditadas a, entre otras muchas posibles condiciones adicionales, coincidencia espacial entre los agentes involucrados. Esto supone que cada agente está asociado a su situación espacial, lo que condiciona sus interrelaciones con otros agentes.

Adicionalmente, los procesos de reproducción van a estar supeditados al cumplimiento de una serie de requisitos tanto de proximidad espacial como de criterios de selección impuestos por cada especie para llevarse a cabo. Esto provoca que la población evolucione de forma asíncrona a diferencia de los algoritmos de evolución tradicionales.

Con respecto a estas dos características, podemos encontrar algunos trabajos que las incorporan en procesos evolutivos. Fundamentalmente encontramos aplicaciones de evoluciones situadas dentro del campo de la robótica evolutiva, en donde esto permite realizar evoluciones en robots reales para evitar el proceso de transferencia de resultados de simulación al entorno real (Usui y Arita, 2003)(Chapelle *y otros*, 2002). También encontramos trabajos que se enmarcan dentro de lo que se ha dado en llamar la *embodied evolution*, en los que los agentes son situados y la evolución es asíncrona e inspirada en la vida artificial, las principales contribuciones han sido realizadas por Watson *y otros* (2002, 1999); Ficici *y otros* (1999). Por su relevancia respecto al tema central de esta tesis, volveremos a comentar estos trabajos más adelante.

3.2.2.5. Conclusiones

En este apartado hemos realizado una revisión del campo de los Sistemas Complejos para llegar, desde un punto de partida más general, al análisis de este tipo de sistemas en el campo de la Vida Artificial. En este ámbito, la mayoría de las aplicaciones que se utilizan para resolver tareas de forma distribuida se pueden incluir dentro de la Vida Artificial débil, es decir, son algoritmos bioinspirados. Dentro de las simulaciones de Vida

Artificial fuerte existen muy pocas aproximaciones con aplicación práctica.

La principal aportación que extraemos de este apartado para el desarrollo de esta tesis se puede resumir en el concepto de sistema complejo con evolución open-ended. Este tipo de sistema está formado por un gran número de elementos simples que interaccionan mediante reglas simples y cuya co-evolución se rige por las reglas naturales de selección, cruce y mutación. Dichos sistemas son perpetuos, y muestran las propiedades típicas de los sistemas dinámicos complejos, tales como comportamientos caóticos, inestabilidades en las soluciones, etc., que siguen ciertos patrones a nivel macroscópico. El enfoque 'bottom-up' que ha seguido en este campo es muy interesante para el planteamiento de una nueva técnica que permita resolver problemas de optimización en ingeniería, ya que posibilita estudiar dinámicas de comportamiento de sistemas muy complejos mediante una definición muy simple de los elementos que lo componen, y con la potencia de búsqueda de un algoritmo evolutivo, en este caso, open-ended.

Como ya comentamos con anterioridad, en este campo de los Sistemas Complejos se asumen de base las características de dinamismo, gran número de interacciones y complejidad intrínseca que en otros campos tienden a evitarse o a ser tratadas por separado. Precisamente este enfoque es el que ha llevado a que la mayor parte de los trabajos desarrollados no contemplen la evolución de los SC desde la perspectiva del objetivo a lograr, sino simplemente desde la analítica. Debemos destacar, por tanto, la notable diferencia entre las teorías y trabajos que fundamentan este apartado y el anterior, dedicado a la Inteligencia Artificial Distribuida.

3.2.3. Integración de conceptos básicos

Las principales conclusiones extraídas del análisis de los campos de la Inteligencia Artificial Distribuida y de los Sistemas Complejos respecto a su utilización en la técnica que se presenta en esta tesis son, fundamentalmente dos:

1. La idoneidad de las *evoluciones de tipo open-ended* que se utilizan dentro de los Sistemas Complejos de Vida Artificial de cara a obtener soluciones distribuidas a problemas dinámicos descentralizados.

En estas, cada individuo de la población representa el sistema de control de cada agente del SMA, de modo que son poblaciones heterogéneas de individuos que co-evolucionan. Este tipo de sistemas de evolución resultan una analogía directa del proceso evolutivo natural.

Las distintas técnicas de análisis que se utilizan dentro del campo de los Sistemas Complejos basadas en las dinámicas de poblaciones son herramientas muy interesantes para estudiar el comportamiento macroscópico del SMA resultante, de modo que los estados de equilibrio que se alcancen puedan ser controlados mediante una modificación paramétrica.

2. La potencialidad de la teoría de *funciones de utilidad privada y global* propia de los SMA que utilizan técnicas automáticas de obtención de comportamientos coordinados, básicamente aprendizaje por refuerzo y técnicas de computación evolutiva. Esta teoría ha proporcionado resultados muy satisfactorios a la hora de obtener SMA que maximicen una función objetivo global mediante la maximización de funciones objetivo privadas de cada agente.

Por tanto, la técnica propuesta será básicamente una *técnica evolutiva de tipo open-ended donde los agentes que conforman la población son evaluados mediante funciones de utilidad privada que deben conducir a la maximización de una utilidad global.*

Este enfoque es totalmente original en los términos planteados, aunque sí existen aproximaciones en la bibliografía que siguen una línea similar y que pasamos a revisar en el siguiente apartado.

3.2.3.1. Trabajos previos

Desde el comienzo de su desarrollo en los años 90, podemos encontrar aplicaciones de la Vida Artificial (VA) en un gran número de campos más allá del campo biológico original. Ya en el año 1991, Taylor (Taylor, 1991) daba algunas ideas de aplicaciones apropiadas para la VA y más tarde Bonabeau y Theraulaz (1991) proponía el diseño de algoritmos como una de las aplicaciones de esta disciplina más allá del simple interés

de simular la *vida tal y como podría haber sido*.

En este sentido, podemos citar algunos resultados prometedores y de interés en los primeros años del nacimiento de la VA. A saber, Hillis (1990) utilizó co-evolución para crear una población algoritmos de ordenación, Colorni *y otros* (1991) propuso un optimizador combinatorio basado en el comportamiento de seguimiento de rutas de hormigas para resolver el problema del viajante, Theraulaz *y otros* (1990) propuso la teoría de swarm intelligence estudiando comportamientos de enjambre para su aplicación en problemas de optimización, entre otros.

Además de esas aplicaciones iniciales, posteriormente hemos visto aplicaciones para campos como: la robótica autónoma, donde se busca obtener formas de comportamiento y organización para los robots basados en comportamientos naturales (Harvey, 1997), los gráficos por ordenador, donde encontramos varias aplicaciones (Terzopoulos, 1999), y también, aunque ya no con un nivel de aplicación no tan inmediato, se encuentran aplicaciones para modelar sistemas naturales (Prusinkiewicz, 2004; Tu y Terzopoulos, 1994).

En cuanto a las aplicaciones para tratar problemas reales de ingeniería tal y como planteamos en esta tesis, hemos encontrado varias aproximaciones inspiradas en algoritmos de VA para la optimización de funciones que luego han sido aplicados a problemas reales de ingeniería. Guo y Kong (2004); Satoh *y otros* (1999); Yang y Lee (2000); fei Yu y wei Wang (2006) proponen varios algoritmos de búsqueda de soluciones, denominados algoritmos de colonización emergente, para la optimización de funciones no convexas y para la optimización de problemas dependientes del tiempo inspirados en las dinámicas que se obtienen en una simulación de VA.

A partir de este tipo de algoritmos, Yang *y otros* (2001) desarrollaron un procedimiento para la optimización paralela y para aplicaciones de diseño. En su trabajo, Yang obtuvo resultados prometedores al aplicar lo que han denominado algoritmos de colonización emergente para el diseño de cojinetes. Más tarde, Ahn *y otros* (2003) utilizaron un híbrido de un algoritmo de VA y una búsqueda tabú en el diseño de alojamientos de motores. También Yang y Lee (2004) combinó la evolución basada en un

algoritmo de colonización de VA y un algoritmo genético para un proceso de optimización.

Si bien estos algoritmos se inspiran en las dinámicas de generación de nuevos individuos, reproducción y colonización que se han estudiado en las simulaciones de VA, no representan en realidad más que un nuevo algoritmo bioinspirado, al igual que el PSO (Kennedy y Eberhart, 1995)o el ACO(Dorigo *y otros*, 1996). En ellos, se generan una serie de interacciones locales que se basan en un comportamiento natural y se utilizan en un algoritmo de búsqueda. Estas aproximaciones requieren de una evaluación centralizada que guíe dicho proceso de búsqueda, y las interacciones entre individuos están prefijadas, no el proceso de búsqueda concreto de cada problema, pero sí las reglas de interacción que guiarán ese proceso.

Ya en una línea que se inspira más en las simulaciones de la tendencia fuerte de la VA encontramos otros trabajos. Una de las líneas que más relevancia poseen dentro del marco de esta tesis es aquella que ha explotado lo que ha dado en llamarse *Embodied Evolution* y ha sido presentada en los trabajos de Ficici *y otros* (1999); Watson *y otros* (1999, 2002).

La idea tras la Embodied Evolution es la de una gran población de robots que se reproducen entre sí y evolucionan en su entorno. Esta aproximación fue primeramente descrita por Husbands *y otros* (1992) y busca una situación ideal en la cual una población de robots evoluciona libre y autónomamente, y por tanto, ofrece muchas posibilidades para la robótica evolutiva. El desarrollo de este objetivo se ha dado en llamar Embodied Evolution (EE) y es un intermedio entre robótica evolutiva y robótica colectiva. Constituye un entorno ideal para estudiar la emergencia de comportamientos de grupo y explorar mecanismos que, adaptativamente, descubren la descomposición de problemas. La reproducción debe ser asíncrona y descentralizada, y para poder ser implementada en los robots Ficici *y otros* (1999); Watson *y otros* (1999, 2002) desarrollaron un nuevo procedimiento de intercambio genético denominado PGTA.

Tal y como explican en su trabajo, la principal inspiración para el desarrollo de la EE son, en gran medida, los experimentos de Vida Artificial. En la evolución natural, los mecanismos adaptativos son completamente

descentralizados y distribuidos, la evaluación es implícita y la reproducción se lleva a cabo autónomamente por los agentes de la población, a diferencia de la evolución artificial en la cual la reproducción la lleva a cabo una autoridad central.

Por otro lado, la aplicación de la robótica distribuida a ciertos problemas es fácil si en estos la descomposición del problema es conocida y las subpartes del mismo son claramente separables e independientes. Sin embargo, las estructuras de la mayoría de los problemas reales no son ni conocidas a priori ni compuestas de subpartes fácilmente separables. Es por esto que la mayor parte del trabajo en robótica colectiva se centra en problemas muy específicos. De este modo, la EE se plantea como un método intrínsecamente basado en poblaciones que se adaptan en un entorno de tareas.

Recientemente, Wischmann *y otros* (2007) realizó un estudio de la EE destacando que para este tipo de evoluciones open-ended, demostrar que el sistema tiende a alcanzar un buen nivel de complejidad es todavía un reto y que muchos investigadores señalan que los experimentos en los que se ha probado esta aproximación son todavía muy simples. En la misma línea que los trabajos de Ficici, encontramos los trabajos de Usui y Arita (2003); Chapelle *y otros* (2002); Konidaris y Hayes (2005) que estudian la EE, y que definen lo que es una evolución situada como aquella en la que los individuos son entidades autónomas que evolucionan dentro de un entorno físico o simulado.

Otra de las aplicaciones que debemos destacar, ya fuera de la EE, fue presentada por Berry y Vamplew (2003) y en ella se introduce una nueva aproximación a la optimización multiobjetivo basada en principios de VA. Esta aproximación no es única y también fue estudiada por Socha y Kisiel-Dorohinicki (2002) y Laumanns *y otros* (1998). El algoritmo se basa en una distribución de un número de recursos distintos equivalente al número de objetivos existente. Se utilizan reglas similares a un sistema de VA típico y una reproducción asociada a una distribución equilibrada de optimalidad en cada recurso. Los resultados son altamente prometedores pues mejoran al método NSGA e igualan al NSGA-II.

Por último, encontramos una aplicación para el diseño emergente en arquitectura (O'Reilly *y otros*, 2000), que si bien no pertenece al campo de la optimización, sí trata con problemas reales y con una serie de restricciones que respetar. En él se señala la idoneidad de muchos de los conceptos de las simulaciones de VA para la resolución de estos problemas que engloban un gran número de elementos fuertemente interrelacionados. Para llevar a cabo esta aplicación, en primer lugar, los autores identifican y cuantifican cada uno de los factores y elementos que caracterizan a cada escenario, así como las interrelaciones entre ellos, y a continuación el sistema se deja evolucionar y se estudian las soluciones que surgen a partir de las interrelaciones definidas.

En conclusión, como hemos visto, existen diversas técnicas que explotan principios basados en las evoluciones de los sistemas complejos de Vida Artificial. Por un lado, más centrados en la tendencia débil, encontramos algoritmos bioinspirados en comportamientos de insectos sociales como el PSO y el ACO, otros inspirados en los procesos evolutivos naturales como los algoritmos genéticos, y otros en las dinámicas de población y en la colonización emergente denominados simplemente algoritmos de VA. Por otro lado, encontramos escasos algoritmos que se inspiran en la tendencia fuerte de la VA. El más destacado en el caso de la Embodied Evolution, perteneciente campo de la robótica, y que simula la evolución asíncrona y situada de un modo natural. Aunque esta aproximación resulta muy prometedora, desde su aparición han sido pocos los trabajos que han explotado este campo y todos han estado limitados al campo de la robótica, nunca a otros campos de la ingeniería.

3.3. Conclusión

El presente capítulo de Antecedentes se ha estructurado en tres partes fundamentales. La primera de ellas se ha dedicado al análisis de los problemas de optimización en ingeniería, y dentro de estos, los que pertenecen al campo de la Investigación Operativa. Este tipo de problemas resultan muy comunes en ingeniería y presentan una serie de características que los hacen especialmente complejos. De entre todas ellas, destacamos el conocimiento parcial de las variables, el gran número de interacciones entre los elementos y el dinamismo del problema. Hemos englo-

bado estas 3 características en 2, que definen el tipo de problemas que se tratarán a lo largo de la tesis: serán problemas *dinámicos descentralizados*. En la actualidad, no existe ninguna técnica de optimización que permita resolver este tipo problemas de ingeniería de forma general, por lo que el planteamiento de una nueva es un tema relevante a nivel investigador, que afronta una necesidad de tipo científico-técnico.

La segunda parte del capítulo se ha centrado en la revisión de los campos de la Inteligencia Artificial Distribuida y de los Sistemas Complejos, de cara a encontrar un enfoque original para la técnica objetivo. Del primero se ha extraído la teoría de *funciones de utilidad privada y global* como fundamento para guiar la técnica computacional hacia un objetivo de diseño. Del segundo se ha obtenido el paradigma básico de planteamiento y resolución de los problemas dinámicos descentralizados, la *evolución open-ended* que se ha venido utilizando en el campo de la Vida Artificial aplicada.

La combinación de evoluciones open-ended con funciones de utilidad es la base de la técnica computacional que se pretende desarrollar en esta tesis. Su diseño, implementación y estudio se llevará a cabo con el objetivo de resolver problemas dinámicos descentralizados en ingeniería. Este planteamiento resulta original y novedoso en el campo de la optimización computacional, y su correcto funcionamiento resultaría muy relevante en distintas áreas de la ingeniería.

La técnica propuesta se ha denominado Co-Evolución Asíncrona Situada (CeAS) y en los siguientes apartados se desarrolla y analiza de forma detallada.

Capítulo 4

Co-evolución Asíncrona Situada. Diseño e implementación

4.1. Especificaciones básicas

En este apartado vamos a describir en detalle la técnica de optimización computacional Co-Evolución Asíncrona Situada (CeAS) propuesta en esta tesis, su esquema de funcionamiento y los parámetros básicos que la definen. Tal y como hemos mencionado anteriormente, la principal aportación de esta técnica es que afronta la resolución de problemas dinámicos descentralizados mediante un sistema de evolución open-ended típico de los Sistemas Complejos de Vida Artificial, que se ha combinado con técnicas de asignación de utilidad propias del campo de los Sistemas Multiagente. Estas técnicas proporcionan la capacidad de guiar la evolución hacia un objetivo concreto además de permitir soluciones que explotan la cooperación de los agentes.

4.1.1. Origen del algoritmo

En el campo de los Sistemas Complejos (SC), se utilizan modelos en los cuales un conjunto de elementos simples generan comportamientos colectivos mediante interacciones locales. Los comportamientos colectivos generados mediante este tipo de estructuras presentan una serie de características inherentes que trataremos de explotar para resolver ciertos

tipos de problemas. Estas son la escalabilidad, la capacidad para encontrar nuevos estados de equilibrio ante cambios en el entorno (adaptatividad), y la distribución y descentralización en la gestión de sus componentes.

El enfoque *bottom-up* que destacamos como característico de los SC hace referencia al nivel en el que se realiza el diseño del escenario, es decir, el trabajo de diseño se realiza en la definición de las interacciones de bajo nivel y el estudio o análisis se realiza en los comportamientos de alto nivel que surgen a posteriori. En concreto, dentro de los SC, e inspirado en la selección y evolución natural, la tendencia fuerte del campo de la Vida Artificial ha estudiado la evolución de modelos de múltiples elementos de interacción local, sus dinámicas colectivas, interrelaciones entre especies, balances energéticos, coevolución, etcétera. Ejemplos de estos son los trabajos de Ray (1991); Adami y Brown (1994b) entre otros.

El algoritmo CeAS se inspira en los estudios de SC en cuanto al uso de evolución open-ended, un tipo de *coevolución descentralizada y asíncrona* de poblaciones que está presente en las simulaciones de Vida Artificial. A diferencia de muchas de las aproximaciones tradicionales bioinspiradas, como es el caso de los algoritmos genéticos, estrategias evolutivas y otras, en los que la selección de los individuos para la reproducción se lleva a cabo mediante una evaluación y comparación centralizada en base a una función objetivo y tras un número fijo de pasos de simulación, *en el CeAS la evolución es situada y asíncrona*. Esto implica que todas las interacciones entre los individuos de la población son locales y están supeditadas a una serie de relaciones de proximidad o coincidencia espacio-temporal, lo cual conlleva una descentralización intrínseca. El CeAS pertenece a la categoría de algoritmos de coevolución debido a que se establece una competencia y comparación local entre individuos de una población, y no en base a una función global preestablecida. El CeAS toma también del campo de los SC la inspiración para el uso de la energía, o una variable equivalente, para guiar y controlar las dinámicas de población y la dirección de la evolución mediante la asociación entre la realización de ciertas tareas y la recompensa energética. Este concepto energético es similar al que aparece en los Sistemas Multiagente basados en modelos económicos, aunque en estos se habla de ‘dinero’.

A pesar del interés de este tipo de coevolución asíncrona situada y su potencial para generar comportamientos complejos, adaptables y escalables que resultan idóneos para tratar problemas reales, nos encontramos con el inconveniente de que en los desarrollos teóricos del campo de los SC no se persigue la optimización de ningún tipo de objetivo o función de utilidad. Como hemos venido destacando a lo largo de los capítulos anteriores, la aproximación *bottom-up* implica un gran trabajo de diseño a nivel de interacciones locales y no un diseño de algún tipo de comportamiento global deseable. Es por esto que vamos a combinar estas técnicas de coevolución asíncrona situada con los desarrollos obtenidos en el campo de los Sistemas Multiagente (SMA), en los que se han venido estudiando múltiples herramientas para obtener el comportamiento deseado en un grupo de agentes. Así, en el campo de los SMA, los protocolos para negociación, comunicación, establecimiento de funciones de utilidad individuales, etcétera, que se han desarrollado y configurado para un sinfín de aplicaciones resueltas mediante una estructura multiagente, nos servirán de inspiración a la hora de plantear los problemas en el CeAS. Las asociaciones entre los elementos de control, supervivencia y reproducción se definirán de manera que la evolución automática, que es la base del CeAS, esté dirigida hacia el comportamiento deseado.

El flujo de ejecución del CeAS, a diferencia de otras técnicas de computación evolutiva, viene definido por la evolución de las interacciones entre elementos del entorno. La generación de eventos de reproducción, aparición de nuevos individuos y eliminación de otros está supeditada a cierto tipo de interacciones entre los agentes que se producen durante la simulación del sistema. En la estructura del CeAS encontramos dos partes que corresponden a las dos aproximaciones que estudiamos como base: por un lado, el motor de la evolución que está basado en las interacciones entre elementos, y por otro lado, los procedimientos que guían la evolución hacia un objetivo, actuando en una serie de puntos concretos que representan la equivalencia entre energía y utilidad y los criterios de evaluación y selección reproductiva.

4.1.2. Elementos y funcionamiento básico del CeAS

A continuación vamos a definir los elementos que forman parte del CeAS y el modo de funcionamiento del mismo. En primer lugar debemos

definir formalmente qué significa que el CeAS sea una técnica que se enmarca dentro de la categoría de evolución*open-ended*. Primeramente, no existe un valor de calidad prefijado o un estado objetivo que definan un criterio de parada para la evolución, sino que se produce una coevolución de forma indefinida realizando mejoras relativas dentro de la población. Adicionalmente, los procesos propios del evolutivo, creación y eliminación de individuos, se realizarán en paralelo con la evolución del escenario creado para representar el problema planteado y estarán supeditados al cumplimiento de ciertas condiciones en este escenario. A diferencia de un evolutivo tradicional, en el cual se fijan estos procesos (evaluación, selección, cruce, etcétera), se realizan secuencialmente y se aplican por igual a toda la población, CeAS requiere de un proceso de evolución del escenario para llevar en paralelo el proceso de evolución de la población de individuos. Es por eso que, dentro de la definición de los elementos y procesos del evolutivo, es necesario incluir procesos de evolución del escenario asociados inevitablemente a la evolución de la población que en el caso de un evolutivo tradicional pertenecerían a la evaluación del individuo y serían independientes de la definición general del algoritmo. Finalmente, esto provocará que la definición y formalización de este algoritmo sea más compleja.

La figura 4.1 muestra una representación de la estructura y el funcionamiento básico del CeAS. Para poder describirlo con detalle, debemos previamente definir una terminología formal para los elementos básicos del mismo. En base a esta terminología, se realizará también la descripción del CeAS en la implementación de los ejemplos propuestos en los siguientes capítulos:

- **Elementos:** serán cada uno de los constituyentes del escenario individuales e independientes.
  - **Individuos:** serán todos aquellos elementos que posean parámetros evolucionables.
  - **Entorno:** denominamos entorno a todos aquellos elementos que no posean elementos evolucionables.
- **Escenario:** es el conjunto de elementos y reglas de interacción que define un modelo determinado.

- **Eventos de interacción:** serán todas aquellas reglas que rijan las interacciones entre los elementos y el entorno.
- **Población:** está representada por cada uno de los grupos de individuos del mismo tipo presentes en el escenario y cuyos miembros se combinan según ciertas normas para generar descendientes del mismo tipo. Dentro de un evolutivo tradicional solo existe una población de individuos, sin embargo en CeAS puede haber más de una población en evolución.

Los principales procesos involucrados en la ejecución del CeAS son los siguientes:

- **Evolución del escenario:** se define como el proceso mediante el cual los estados de los elementos del escenario van modificándose de acuerdo a las reglas de interacción. Este proceso es independiente de la evolución de la población, y por tanto, se produce la evolución del escenario aunque no exista evolución de la población. Dentro de esta evolución, definimos una serie de situaciones que requieren ser tratadas independientemente y que enlazan con el reemplazo de la población:
  - **Flujo de energía:** representa las diferentes reglas que rigen las variaciones de energía entre los individuos y el entorno, y entre el entorno y el exterior.
  - **Selección reproductiva:** representa el proceso mediante el cual se realiza la selección de determinados individuos para participar en el proceso reproductivo. Tal y como veremos, para cada ejemplo debemos definir este proceso de selección, aunque en algunos casos concretos, como veremos en los ejemplos desarrollados, es posible la definición de un procedimiento general para realizar la selección cuando se cumplen ciertas condiciones.
- **Evolución de la población:** se define como el proceso mediante el cual una población cambia el número y/o las características de sus componentes. Requiere de la evolución del escenario para llevar cabo varios procesos que lo componen. Este proceso está constituido por los siguientes elementos:

- **Codificación de la población:** es el proceso de definición de los parámetros evolutivos que poseerán los individuos de manera que estos puedan representar un conjunto de comportamientos que comprendan el comportamiento objetivo que se desea generar.
- **Función objetivo:** es la encargada de guiar la selección reproductiva a través de dos mecanismos. Por un lado, define las condiciones de entrada en la selección reproductiva, y por otro lado, de manera más indirecta, define la asociación energía-utilidad para premiar energéticamente comportamientos que van a favor de la función objetivo. Posteriormente, la determinación de los individuos base de entre aquellos incluidos en la selección reproductiva, se hará en función de los valores de energía de cada individuo y de algún otro criterio de utilidad no incluido en la asociación con la energía.
- **Proceso reproductivo:** es el proceso en el cual a partir de unos individuos base se realiza un proceso de *combinación genética* en el cual se genera otro individuo y se introduce en el escenario.
    - **Individuos base:** dos individuos elegidos de entre aquellos incluidos en la selección reproductiva que serán los que intervengan en el proceso reproductivo.
    - **Cruce bipolar:** consiste en un proceso de combinación de los parámetros evolutivos de los individuos base, desarrollado específicamente para el CeAS, y que se describe más adelante en el apartado de diseño. Su característica principal es que permite mantener grupos diferenciados de elementos con distintos parámetros y distintas capacidades, en oposición a los métodos de cruce habituales que tienden a homogeneizar la población.
- **Eliminación de individuos:** representa las condiciones que se deben producir para la eliminación de un individuo del escenario.
- **Criterio de parada:** como comentamos en la introducción a este capítulo, el criterio de parada que puede utilizarse aquí es el detener la evolución tras un número fijo de pasos de tiempo,

pues implícitamente el uso de coevolución implica que no pueden existir otros criterios de parada.

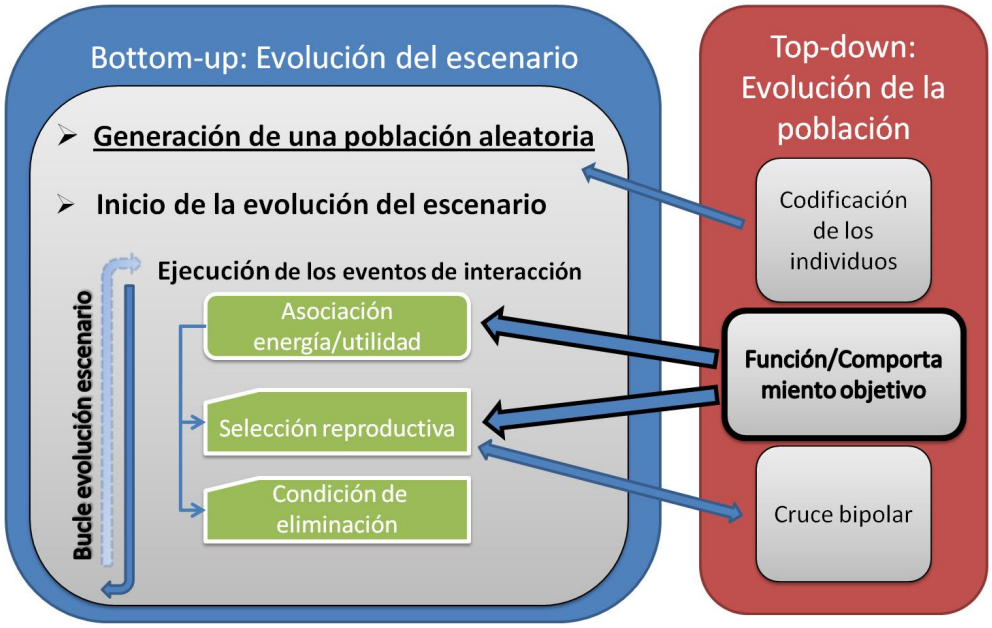


Figura 4.1: Esquema básico de funcionamiento del CeAS.

En la figura 4.1 se muestra una representación del proceso de evolución que propone el CeAS. Tal y como se muestra en la figura, el proceso posee dos partes diferenciadas y están representadas por los bloques de la izquierda y de la derecha. Estas partes corresponden a las dos aproximaciones que confluyen en el diseño del algoritmo, por un lado la evolución de un sistema complejo mediante la definición de una serie de reglas de interacción de bajo nivel (bloque de la izquierda), y por otro lado, la evolución de la población de individuos de acuerdo a una serie de objetivos definidos por el diseñador e introducidos en el evolutivo a través del proceso de selección y evaluación reproductiva (bloque de la derecha). El proceso que se lleva a cabo en el bloque de la izquierda para evolucionar el sistema dinámico generado por los elementos y sus interacciones se ejecuta en los siguientes pasos:

1. En primer lugar, se genera una población de individuos cuyos parámetros se crean aleatoriamente dentro de la definición establecida en la codificación de los individuos.

2. A continuación, se inicializan el resto de elementos del escenario y se inicia el bucle de evolución del mismo. En este bucle se llevan a cabo las acciones definidas en los eventos de interacción. Existen también ciertas interacciones y estados del sistema que estarán controlados por el bloque de evolución de la población, y mediante esta conexión entre los dos bloques se conseguirá guiar a la población hacia el objetivo predefinido. Podemos resumir esta interacción entre bloques en los siguientes puntos:

   a) En primer lugar, determinadas acciones definidas por la función o comportamiento objetivo llevan asociada una recompensa energética y otras una penalización. La definición de esta asociación cuantitativa y cualitativamente la lleva a cabo el bloque de gestión del reemplazo de la población, y se concreta en la variación del valor de energía asociado a cada individuo durante la evolución.

   b) En segundo lugar, el proceso de selección reproductiva seguirá criterios basados en los establecidos a partir de la función objetivo en el bloque de reemplazo de la población. Según estos criterios, un individuo entrará dentro de la selección reproductiva cuando sus parámetros cumplan ciertas condiciones. Una vez que se ha realizado esta selección, los individuos entran un torneo dentro del cual se seleccionarán dos individuos base. El valor de utilidad por el que se rige el torneo será, por norma general, el valor de energía, aunque también puede incluir algún otro tipo de restricción o criterio de evaluación que no haya sido incluido en la asignación de energía.

   c) Finalmente, en tercer lugar, existen una serie de criterios, definidos también por el bloque de evolución de la población con respecto a las condiciones para la eliminación de un individuo. Primeramente, siempre tenemos la condición de eliminación asociada a que un individuo agota sus reservas de energía que nos asegura que el balance de energía sea consistente. A mayores de esta, estarán otras condiciones como el tiempo de vida

máximo y algunas otras definidas con el objeto de guiar a la población de un modo u otro.

El bloque de la derecha en la figura 4.1 corresponde al establecimiento de los criterios que regularán los cambios en las poblaciones y están inspirados en las técnicas desarrolladas dentro de la Inteligencia Artificial Distribuida para guiar a un Sistema Multiagente hacia un comportamiento con un objetivo. Posee tres elementos principales que se interconectan con la evolución del escenario:

1. La codificación de la población: como hemos comentado, definirá los comportamientos que pueden generar los individuos, y por tanto, el espacio de búsqueda sobre el que trabajará el evolutivo que guía el reemplazo de la población. Sobre esta codificación se generará aleatoriamente el valor de cada uno de los parámetros de cada individuo y esto constituirá la población inicial.

2. A partir de la función o comportamiento objetivo definido para cada escenario, se definen los criterios de selección reproductiva y de asignación de energía dentro de la evolución del escenario. La definición de estos criterios es fundamental dentro de la aplicación del CeAS a un problema dado, pues estos guiarán, en primer lugar, la evolución en el proceso de reemplazo de la población, y como consecuencia, la dinámica de comportamientos que se generarán en el escenario. Estos criterios definirán también el modo en el que se alinea la utilidad global del escenario con las utilidades privadas de cada individuo. Este estudio debe llevarse a cabo particularmente para cada escenario, aunque hemos encontrado en la bibliografía ciertas consideraciones generales como se comenta en el apartado de comportamientos coordinados.

3. Una vez que se han seleccionado los individuos base, se utiliza un tipo particular de cruce para la combinación genética que se denomina cruce bipolar y cuyo objetivo es el de mantener poblaciones estables en las que existan varios tipos de individuos diferenciados. Esto provoca que se reduzca la tendencia de los cruces tradicionales a converger hacia poblaciones homogéneas.

4.1.3. Consideraciones de diseño para el planteamiento de un problema en el CeAS

A continuación expondremos una serie de puntos que hemos considerado indispensables para llevar a cabo la transformación entre la definición real de un problema (tarea requerida, definición del entorno, restricciones aplicables, función objetivo) y la definición del escenario de simulación y de los individuos evolucionables, teniendo en cuenta que, tanto la tarea como el proceso de evolución, deben realizarse de manera distribuida y descentralizada para garantizar la adaptabilidad, escalabilidad y robustez de las soluciones generadas.

- **Definición de los individuos**

  El primer paso que debemos llevar a cabo en la tarea de diseño, es el de definir cuáles serán los individuos que estarán sujetos a evolución dentro del escenario. Estos individuos deben ser capaces de llevar a cabo la tarea que se desea ejecutar de forma distribuida y deben comprender con sus potenciales comportamientos el abanico de soluciones que queremos abarcar.

  En este apartado se definirán también todas aquellas acciones que se realicen de forma individual y cuyos efectos no afecten de forma directa a otros individuos o al entorno, como por ejemplo, las sensorizaciones o actualizaciones de estados internos. Los eventos que definan las interacciones colectivas y las interacciones con el entorno se definirán en el siguiente apartado.

  Se definirá también el sistema de control utilizado, su funcionamiento, sus parámetros de configuración, el proceso de entrenamiento o adaptación de este sistema de control y las acciones asociadas a él.

  Finalmente, se definirán cuáles serán los parámetros de los agentes que constituirán el código genético de los individuos a los que representan. Es decir, cuáles serán los parámetros evolucionables de entre todos los parámetros que poseen los individuos.

- **Modelado distribuido y descentralizado**

  La primera tarea dentro del proceso de generación de un escenario susceptible de ser evolucionado mediante el CeAS consiste en transformar el problema que queremos resolver en un problema distribuido, es decir, que el comportamiento, tarea, búsqueda, etcétera que se esté intentando generar de acuerdo a unos objetivos, sea divisible en elementos más sencillos, que permitan afrontar dicho problema de forma descentralizada. Este es de forma que las interacciones, alcance de actuación o entrada de información es local y parcial. El CeAS podría, al igual que cualquier otro algoritmo evolutivo, utilizarse para un problema centralizado, pero sin embargo, y a diferencia de la mayoría de los evolutivos, el CeAS debido a su carácter local (situado) puede ser también aplicado a problema descentralizados, y de hecho, es en este tipo de problemas donde cobra mayor relevancia y en los que nos interesa su aplicación. Si buscamos características como la robustez o escalabilidad, un modelo descentralizado es un requisito indispensable. Para garantizar la resolución distribuida y descentralizada debemos garantizar que estos dos requisitos se cumplen en los siguientes 4 niveles:

  - **Información de entorno**

    Es preciso definir los procedimientos de entrada de la información proveniente del entorno y que consideremos relevante para resolver el problema de forma descentralizada: *sistema de sensorización exclusivamente local e información completamente distribuida*.

  - **Información de grupo**

    Procedimientos de obtención de información de otros individuos de la población: *protocolos de comunicación local implícita o explícita*.

  - **Actuación sobre el entorno**

    Definición de las tareas que pueden llevar a cabo los elementos de la población para que estas puedan generar el comportamiento conjunto deseado, y paralelamente, las actuaciones que el entorno puede tener sobre los elementos de cada población:

definición de eventos de interacción entre individuos y entorno con un rango de alcance limitado.

- **Actuación en el grupo**
  Definición de las tareas que se pueden llevar a cabo de manera que modifiquen el estado de otros miembros del grupo y que puedan ser relevantes para la resolución de la tarea total: *interacciones entre individuos de alcance limitado para tareas de colaboración, competición y reproducción.*

- **Estudio de la utilidad privada y utilidad global**

  En este apartado se estudiará la asignación de recompensas de manera que el comportamiento individual de los vigilantes conduzca hacia el objetivo global propuesto por el diseñador. Esto, como ya hemos visto en el estudio de los antecedentes, para algunas aplicaciones no es una tarea trivial, y tal y como estudiaron ciertos autores (Agogino y Tumer, 2008; Wolpert y Tumer, 2000), la utilidad individual debe cumplir una serie de condiciones para que guíe al sistema hacia una mejora de la utilidad global. Por un lado, la mejora en la utilidad privada de un agente debe conllevar una mejora en la utilidad global, y por otro lado, la variación en la utilidad privada de un agente debe estar afectada en la medida de lo posible únicamente por acciones propias de ese agente para permitir la creación correcta de asociaciones entre acciones y valores de utilidad por parte de cada agente. Adicionalmente, la asociación de estos valores de utilidad con ventajas evolutivas se realizará más adelante mediante los mecanismos de gestión energética, selección reproductiva y evaluación. Deberemos tener en cuenta, a la hora de definir estos valores de utilidad, que esas asociaciones deben realizarse de forma descentralizada, mediante las interacciones locales que se producen en la evolución del escenario, y que por tanto, alguna de las medidas de utilidad podrían no ser válidas.

- **Gestión energética**

  Del estudio de las simulaciones de Vida Artificial, dinámicas de población y los Sistemas Complejos extraemos la idoneidad del uso de

una variable energética para establecer control sobre las dinámicas que se generan en el sistema y para guiar la evolución, de modo que esta energía represente una medida implícita de calidad. Taylor (2001); Bianco y Nolfi (2004) en sus estudios sobre evolución open-ended establecen también que en este tipo de evoluciones se hace necesario el uso de la energía en lugar de una función de calidad de alto nivel.

A mayores, CeAS propone una gestión energética no inspirada en algún fenómeno natural o decidida arbitrariamente, sino diseñada de manera que se premie el comportamiento de la población que optimice una variable de calidad de alto nivel.

Estos flujos de energía representan una herramienta indispensable para controlar las dinámicas de población, asociar la eficiencia con la reproducción, con la supervivencia, etcétera. En el CeAS, el flujo de energía será también la vía que utilicemos para dirigir la evolución del escenario de la manera deseada. Es así un apartado fundamental en la definición de las interrelaciones entre elementos de la población y el entorno que, aunque podríamos incluirlos en los dos últimos apartados del modelado, requieren de un tratamiento diferenciado. Distinguimos los siguientes niveles en los que se debe concretar esta gestión energética:

- *Entrada de energía:* la entrada de energía al entorno va a definir, por un lado, una dinámica poblacional. En el extremo inferior de la cadena energética, el entorno como generador de energía será la base que rija las variaciones en las densidades de población de cada especie. En función de si esta entrada de energía depende de parámetros del entorno, de variables externas definidas por el diseñador o de una combinación de ambos, definirá las fluctuaciones y el punto -o los puntos- de equilibrio factibles en el modelo. Por otro lado, estas mismas dinámicas regirán la presión selectiva, y por tanto, el curso de la evolución de cada una de las poblaciones.
- *Salida de energía:* esta salida de energía, que es por supuesto necesaria para poder alcanzar una estabilidad, puede realizar-

se, en parte, por el propio entorno (*caducidad de la energía o estados de disipación de energía*) o también a través de los propios miembros de cada población, que poseerán un ritmo de consumo energético.

- *Intercambio energético:* dentro de los flujos de energía está el intercambio de energía entre los agentes, que en muchos casos puede ser interesante en fenómenos de cooperación, en la reproducción, etcétera y debe ser también definido en la caracterización del escenario
- *Asociación energía-utilidad:* por último, y realmente de gran interés, es la asociación que vayamos a realizar entre la energía y el nivel de utilidad. De cara a generar todos los flujos de entrada de energía a cada uno de los individuos, debemos tener en cuenta que esta debe ir asociada de algún modo a la eficiencia en la realización de alguna tarea o en la consecución de algún objetivo. Es aquí donde entran las consideraciones *top-down* de los Sistemas Multiagente de cara a establecer protocolos de transmisión de energía, consumo energético, coeficientes de obtención de energía/eficiencia, estudio de la utilidad individual, etcétera y que van a servir, como veremos en los siguientes apartados, para establecer los criterios de evaluación y selección para la reproducción.

- **Selección reproductiva**

  Tras haber definido el método de evaluación, es necesario definir el criterio de selección para la generación de nuevos individuos. Esto, al igual que todos los elementos anteriores, debe hacerse de manera descentralizada, y es por eso que utilizaremos un tipo de selección equivalente a un torneo tradicional, pues es la única estrategia de selección que permite ser realizada de forma descentralizada.

  - *Proceso de selección:* un torneo estándar en un evolutivo tradicional consiste en seleccionar aleatoriamente un número de individuos de entre toda la población existente. Es esta parte centralizada la que debemos rediseñar para que sea sustituida mediante una selección asíncrona y descentralizada. Para

ello, en CeAS el torneo siempre se realizará de forma local y supeditado a determinadas condiciones que irán asociadas a la eficiencia en la tarea individual y que se generarán de forma asíncrona.

- *Asociación selección/nivel de utilidad:* permitir a un individuo acceder a la selección reproductiva es otro modo, junto con la evaluación, de conducir a la población hacia uno u otro tipo de comportamiento. Por tanto, las condiciones de entrada pueden asociarse, o bien al mismo criterio de evaluación, o bien a algún otro criterio que añada otro nivel de selección. De todos modos, sea cual sea el criterio de selección, al igual que en el resto de eventos, este debe estar basado en interacciones locales entre los individuos.
- *Parametrización de la selección reproductiva:* una vez definidos los criterios de selección, se definirá el criterio para la realización del torneo en función del criterio tradicional de tamaño de ventana, es decir, cuando se alcance un tamaño de ventana requerido se realiza el torneo. También será posible la utilización de un criterio basado en otro parámetro como el tiempo transcurrido desde el inicio del torneo, o algún otro que en el caso de este torneo asíncrono sí pueda ser de aplicación.

- **Evaluación**

Con respecto a la evolución de la población, debemos estudiar cómo se realiza la evaluación de cada uno de los individuos de cara a elegir aquellos que participarán en la selección reproductiva y la generación de nuevos individuos.

- *Estudio de los criterios de evaluación/asignación de utilidad:* en una primera aproximación, el criterio para la evaluación de un individuo con respecto a los criterios de selección para ser escogido como individuo base, estará ligado directamente a su nivel de energía. Sin embargo, esta relación no tiene porque ser totalmente directa, e incluso puede depender de más parámetros. Si bien es cierto que podríamos variar la asignación de energía para incluir los valores de estos parámetros también

relevantes, puede, en algunos casos, ser de interés establecer una diferenciación entre la energía y otros parámetros que afecten también a la evaluación. Por ejemplo, aunque el tiempo de vida de un individuo podría ser de interés para la evaluación, y por tanto, podríamos incluirlo de algún modo en el nivel de energía, es posible que nos interese diferenciar el tiempo de vida de un individuo y su nivel de utilidad para procesos que diferencien estos dos indicadores. Adicionalmente, podemos utilizar no sólo el nivel de energía actual sino el nivel de energía histórico, el máximo o mínimo de un período, etc.

- *Estabilidad de la utilidad*: previo al análisis de la estabilidad de la evaluación de un individuo, debemos definir si nos interesa o no dicha estabilidad, o si nos interesa en unas situaciones y no en otras. Si bien el nivel de estabilidad de la utilidad nos genera una evaluación más precisa, también un cierto nivel de inestabilidad puede mejorar el comportamiento exploratorio del algoritmo de una forma que no se podría llevar a cabo con algunas otras estrategias convencionales utilizadas para mejorar la exploración. Un nivel de utilidad inestable, generará valores de utilidad instantáneos, y en muchos casos diferentes al valor de utilidad promedio, debido a fluctuaciones en el entorno de simulación, a modelos con entrada de energía puntual y periódica, etcétera. Sin embargo, esos valores son valores reales, se han generado en unas condiciones determinadas que se han producido en un momento dado, y por lo tanto, esa variabilidad no es arbitraria o completamente aleatoria como ocurre al introducir una probabilidad de cruce o una ventana de torneo más reducida, lo cual puede ser de interés y tenido en cuenta por el diseñador.

- **Combinación genética**

Con respecto al cruce, una vez seleccionados los individuos participantes, este no difiere del cruce tradicional en cuando a estructura, sin embargo, sí debemos considerar que, a diferencia de otros evolutivos en los que el objetivo es obtener una población tal que su mejor individuo tenga el mayor nivel de utilidad posible, en este

caso se busca una configuración de población lo más eficiente posible, es decir, el evolutivo busca una población idónea y no un individuo óptimo. Los cruces habituales tienden a generar una población que, a medida que avanza la evolución, es más homogénea. Si bien muchas de las soluciones a modelos de problemas descentralizados podrían estar formadas por poblaciones homogéneas, debemos contemplar la posibilidad de que una población heterogénea pueda generar un resultado más eficiente que una homogénea. Para ello, se ha diseñado un operador de cruce diferente denominado *cruce bipolar* que permite la existencia de varias especies diferenciadas.

- *Nivel de explotación:* de igual modo, el nivel de explotación indicará la capacidad para afinar una solución basándonos en la solución óptima actual. El cruce que planteamos en el siguiente sub-apartado permite un ajuste de la relación entre estos dos niveles modificando la función de probabilidad utilizada para generar nuevos genes. Adicionalmente, la disminución de variabilidad genética hace que, a medida que la población va convergiendo, la explotación va imponiéndose con respecto al de exploración.
- *Nivel de exploración:* como en cualquier otro evolutivo, diferenciamos un nivel de exploración que nos va a indicar la tendencia a buscar nuevas soluciones independientemente de las zonas con un mayor nivel de utilidad asociado actuales, y que en el caso del cruce que vamos a utilizar, vendrá dado por la probabilidad de cruce y la desviación estándar de la función de probabilidad aplicada para generar un nuevo gen a partir de los genes de los padres.
- *Creación de especies:* Como hemos dicho, se permitirá la existencia de especies diferenciadas mediante un nuevo tipo de cruce que pasamos a explicar a continuación y que hemos denominado **cruce bipolar**.

El cruce es un cruce de números reales, que se realiza gen a gen.
Sea $F_p = [F_1, F_2, \ldots F_n]$ el cromosoma del primer padre.
Sea $S_p = [S_1, S_2, \ldots S_n]$ el cromosoma del segundo padre.

Sea $N_c = [N_1, N_2, \ldots N_n]$ el nuevo cromosoma.

La idea que inspira la creación de este nuevo cruce es la de permitir obtener poblaciones no homogéneas tras el proceso de evolución. Debido a que los operadores de cruce utilizados en los algoritmos evolutivos tradicionales, una vez que se pierde la diversidad inicial, tienden hacia una población homogénea, vamos a crear un cruce que permite que las especies no tiendan a un código genético intermedio.

Para ello, la idea fundamental consiste sencillamente en que, en primer lugar, se selecciona un cromosoma base, F_p o S_p, con la misma probabilidad para ambos y el nuevo individuo será, con una alta probabilidad, una variación de este. Si no seleccionásemos un cromosoma base, la probabilidad de obtener un nuevo individuo similar a alguno de los padres sería muy baja y se reduce exponencialmente con el número de genes.

A continuación, para cada uno de los genes obtenemos una desviación $D_i =\mid F_i - S_i \mid$. El valor para cada nuevo gen se genera utilizando una función de densidad de probabilidad de tipo gaussiano centrada en cada uno de los genes del cromosoma base y con una anchura basada en la desviación (D_i) (ver figura 4.2).

La función presenta una forma similar a una campana gaussiana (y_{sg}) y en efecto las diferencias con una función gaussiana (y_g) son menores a 0.01 ($\sum \mid y_g - y_{sg} \mid$) si ajustamos las dos curvas pero sin embargo esta expresión sí es integrable a diferencia de una gaussiana estándar y por eso nos resulta más eficiente en términos computacionales.

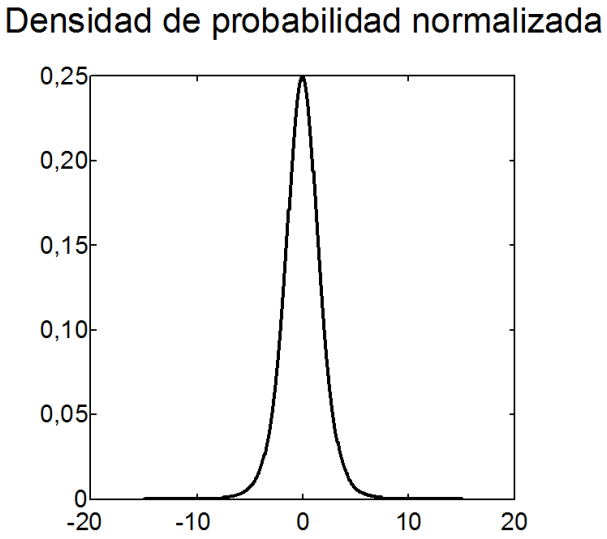


Figura 4.2: Función de tipo gaussiano que representa la densidad de probabilidad normalizada para el cálculo de los genes del nuevo cromosoma a partir del cromosoma base.

La densidad de probabilidad normalizada es igual a:

$$Densidad\_probabilidad = \frac{e^{-X}}{(1+e^{-X})^2} \tag{4.1}$$

siendo para cada gen $X = \frac{N_i - F_i}{Dev}$

Para generar los nuevos valores usamos la función de distribución de la función de densidad que hemos definido (figura 4.3).

$$Distribucion_Normalizada = \frac{1}{1+e^{-X}} \tag{4.2}$$

En esta distribución de probabilidad, el 90 % de los valores se sitúan dentro del intervalo $[F_i - 3 \cdot Dev, F_i + 3 \cdot Dev]$ Usando $D_i/2 = 3 \cdot Dev$ estoy situando el 90 % de los valores de los nuevos genes dentro del intervalo $[F_i - D_i/2, F_i + D_i/2]$.

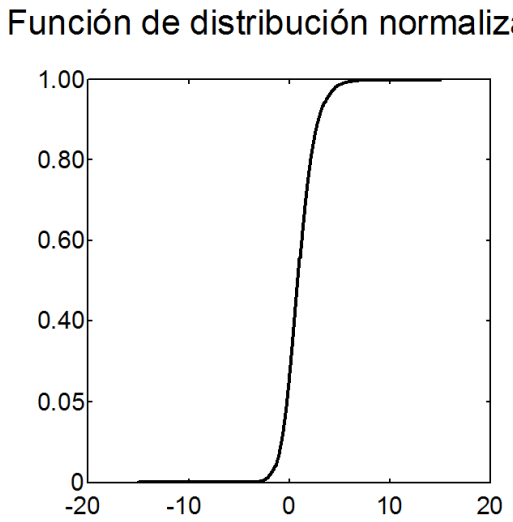


Figura 4.3: Función de distribución obtenida mediante la integración de la función de densidad de probabilidad para la generación aleatoria de los nuevos genes.

Este valor de Dev se elige de un modo arbitrario. La idea es mantener la gran mayoría de los valores nuevos dentro del intervalo $[F_i - D_i/2, F_i + D_i/2]$, y después de hacer varias pruebas con diferentes valores en las simulaciones iniciales, tomamos esa amplitud $(Dev = D_i/6)$ para la campana tipo Gaussiano.

Con la finalidad de obtener la distribución de probabilidad deseada a partir de una distribución aleatoria constante tenemos:

$$N_{cd} = \frac{1}{1+e^{-x}} \Rightarrow X = log\left(\frac{N_{cd}}{1-N_{cd}}\right) \Rightarrow \frac{N_i - F_i}{Dev} = log\frac{N_{cd}}{1-N_{cd}} \tag{4.3}$$

Introduciendo un valor aleatorio para N_{cd} dentro del intervalo $[0, 1]$ y substituyéndolo en la última ecuación, obtendremos el valor deseado de X, que corresponde a N_i.

En resumen:

$$N_i = log\frac{N_{cd}}{1 - N_{cd}} \cdot \frac{D_i}{6} + F_i \tag{4.4}$$

con $N_{cd} = Random[0, 1]$.

Por tanto, la función de densidad de probabilidad resultante, teniendo en cuenta ambos padres, presenta la forma que se representa en la figura 4.4.

Debemos tener en cuenta también que esta representación es para cada uno de los genes, y que una vez que el elemento base está seleccionado, todos los puntos se generarán en torno a los genes de dicho elemento base.

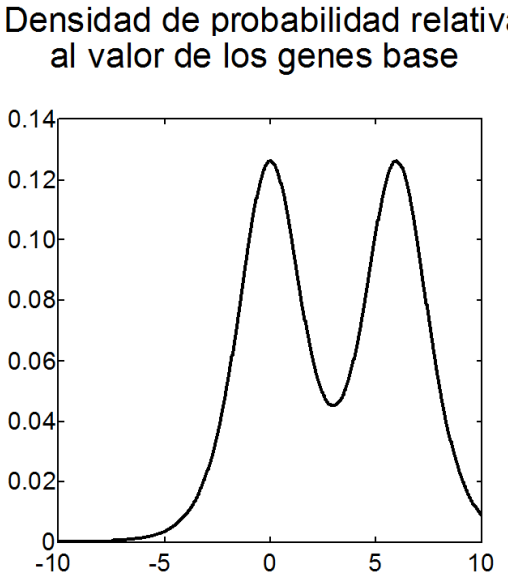


Figura 4.4: Densidad de probabilidad relativa al valor de los genes base. En este caso $F_1 = 0$, $S_1 = 6$ pero la forma es siempre la misma.

La mutación sigue también una función gaussiana pero no es dependiente de D_i y es lo que aporta variabilidad a la evolución cuando la población ha convergido. La probabilidad de mutar cada gen es 0.05

y la desviación es siempre 0.05.

`If (`$rand[0,1]$ `<`$Pmutacion$`)`
`(`$Xmutated$ `-` F_i`)/`$DesviacionMutacion$ `= log(`N_{cd}`/1-`N_{cd}`);`

Cuando la población ha convergido a una población homogénea, o a varias poblaciones homogéneas, D_i tiende a ser cercano a cero y prácticamente no hay cambios en el código genético en las nuevas generaciones. Solo esta mutación podría permitir ligeras variaciones que conducirían a un 'ajuste fino' en la solución.

- **Reemplazo**

Finalmente, dentro de los parámetros de diseño de la evolución, tenemos aquellos que nos van a dar una idea del tiempo de reemplazo medio de una generación de individuos y que nos permitirán controlar, principalmente, la estabilidad y adaptabilidad de una población. Un tiempo de reemplazo grande permitirá que una población explote a fondo una configuración determinada de elementos y afine su comportamiento para optimizarlo. Por otro lado, una variación en las condiciones del entorno será más difícil de superar, pues la población tardará más en renovarse y en generar una nueva configuración de elementos que se adapten a la nueva situación. Por el contrario, una población con un tiempo de reemplazo pequeño será más inestable y convergerá con mayor dificultad hacia una configuración de población fija. Sin embargo, se adaptará con más facilidad a cambios en el entorno que exijan cambios en la configuración.

A nivel de diseño distinguimos dos parámetros que afectan al tiempo de reemplazo:

- *Periodo de permanencia:* es el tiempo medio de supervivencia de un individuo. Está relacionado con el tiempo de vida máximo, con la cantidad de energía disponible por agente y con la variabilidad de energía disponible, es decir, con el carácter fluctuante de la energía disponible. Un modelo en que se generen fluctua-

ciones locales grandes en relación al nivel medio de energía por agente, provocará la eliminación frecuente de individuos.

- *Elitismo:* representa la tendencia del operador de cruce a mantener a individuos con un nivel de eficiencia alto. Esto hace que un mismo individuo, aún a pesar de haber sido eliminado, transmita una copia exacta de su código genético, y por tanto, siga presente en el entorno. Es decir, alarga el tiempo de vida de aquellos individuos con gran nivel de eficiencia, y por tanto, aumenta el tiempo de reemplazo.

4.1.4. Consideraciones sobre el análisis de un problema en el CeAS

Finalmente, y para concluir la descripción del CeAS, a continuación se presenta una metodología básica de análisis del sistema, cuyo objetivo es simplificar la extracción y el estudio de resultados durante la ejecución del evolutivo. Tal y como podemos imaginar, y más adelante comprobaremos, teniendo en cuenta los parámetros definidos para todos los elementos que constituyen el escenario, los procesos que rigen los flujos de energía, los parámetros propios de la evolución de la población, selección reproductiva, cruce bipolar, etcétera, cualquier implementación de un problema, ha de tratar con una serie de valores y rangos para un gran número de parámetros que afectan de una forma importante al proceso evolutivo tanto del escenario como de la población.

Por tanto, este análisis se va a basar principalmente en el establecimiento de un conjunto de parámetros de alto nivel que aglutinen a todo el conjunto de parámetros que son relevantes para el transcurso de la evolución, representen una serie de medidores significativos y suficientes para la caracterización de dicha evolución y sean generalizables para cualquier escenario. Este análisis no solo servirá para estudiar el comportamiento de una evolución determinada, sino que también servirá para configurar los parámetros de diseño del escenario para generar dinámicas que nos sean de interés para guiar la ejecución: regular la convergencia de la población hacia una situación estable, regular la diversidad de la población, regular el tamaño de la población, etcétera. Adicionalmente, el uso de un

conjunto fijo de parámetros de análisis nos permitirá extrapolar resultados y técnicas de control a otros casos de estudio.

Hemos denominado a los parámetros de bajo nivel *parámetros de configuración*, y a los parámetros de alto nivel, *parámetros de análisis*. Como se podrá observar, estos parámetros de análisis se identifican con muchos de los parámetros expuestos en el apartado de especificaciones del CeAS, y como hemos mencionado, a diferencia de los parámetros de configuración, serán los mismos para cualquiera de los ejemplos generados. Por último, asociado a cada uno de los parámetros de análisis, debemos definir forma de medir cuantitativamente el valor de cada uno de ellos.

En la figura 4.5, se muestra un esquema de la transformación paramétrica. Este esquema representa el estudio paramétrico que deberemos hacer para cada uno de los escenarios propuestos de cara a estudiar y ordenar las interrelaciones entre los parámetros de configuración de cada escenario y los parámetros de análisis genéricos. Son los siguientes:

- *Tamaño de la selección reproductiva:* define el número de individuos que son seleccionados para participar en la evaluación y elegir a los individuos base. Es equivalente al tamaño de torneo en un cruce tradicional, y por tanto, un número de elementos alto incrementa la explotación y disminuye la exploración en el proceso de búsqueda asociado a la evolución de la población, y viceversa para tamaño de selección bajo. En el caso de que este tamaño sea fijo, su medición es directa. Si el tamaño está supeditado a un tiempo máximo de selección o algún otro tipo de condición se generará empíricamente un valor promedio.

- *Tiempo de evaluación:* mide el tiempo medio de evolución de escenario que cada individuo ha estado interactuando con el entorno cuando este es evaluado. Representa el nivel de precisión a la hora de asignar un nivel de utilidad a un individuo: cuanto mayor sea el tiempo de evaluación, el valor promedio de utilidad es más representativo del comportamiento del individuo. Debido a que la selección es asíncrona, este parámetro tendrá un medidor obtenido siempre de forma empírica.

- *Variabilidad del nivel de utilidad:* indica la variabilidad a la hora de asignar un nivel de utilidad a un individuo, es decir, la sensibilidad de la medida escogida para evaluar el nivel de utilidad frente a variaciones del entorno. Para medir el nivel de variabilidad, se obtendrá alguna medida de desviación asociada a los valores de utilidad de todos los individuos de la población a lo largo del tiempo de ejecución.
- *Presión de explotación y exploración en la evolución de la población:* indica la tendencia que tiene el operador de combinación genética a converger hacia la solución óptima actual o a explorar zonas subóptimas. Debido a que la combinación genética se realizará en el CeAS siempre mediante el cruce bipolar, estos dos parámetros se generarán siempre del mismo modo y a partir de la probabilidad de cruce, probabilidad de mutación y las desviaciones estándar de las funciones de probabilidad del cruce y mutación.
- *Tiempo de reemplazo medio:* finalmente, y como ya expusimos en el apartado de diseño, representa el tiempo medio para reemplazar una población de individuos. Este nivel de reemplazo se obtiene también de forma empírica durante la evolución del escenario, contabilizando al número de eliminaciones y generaciones de nuevos individuos.

Una vez que hayamos definido nuestro escenario, si vamos a llevar a cabo este análisis, lo primero que debemos hacer es la creación de las asociaciones entre los parámetros anteriores y definir el efecto de la modificación de los parámetros de configuración en los parámetros de análisis. En la mayoría de los escenarios no vamos a poder definir de forma exacta el efecto de una variación en un parámetro de análisis en un parámetro de configuración, pero sí si siguen una relación directa o inversa. Esto será suficiente para modificar en la dirección adecuada hasta alcanzar el valor deseado. El resultado final, tal y como veremos en los tres ejemplos propuestos en esta tesis, será una tabla que refleje todas estas relaciones, de modo que cuando queramos influir sobre algún parámetro de análisis, podamos extraer de la tabla qué parámetros debemos modificar y de qué manera.

El planteamiento y análisis de los problemas que se resuelven mediante el CeAS requiere del desarrollo de una herramienta de simulación

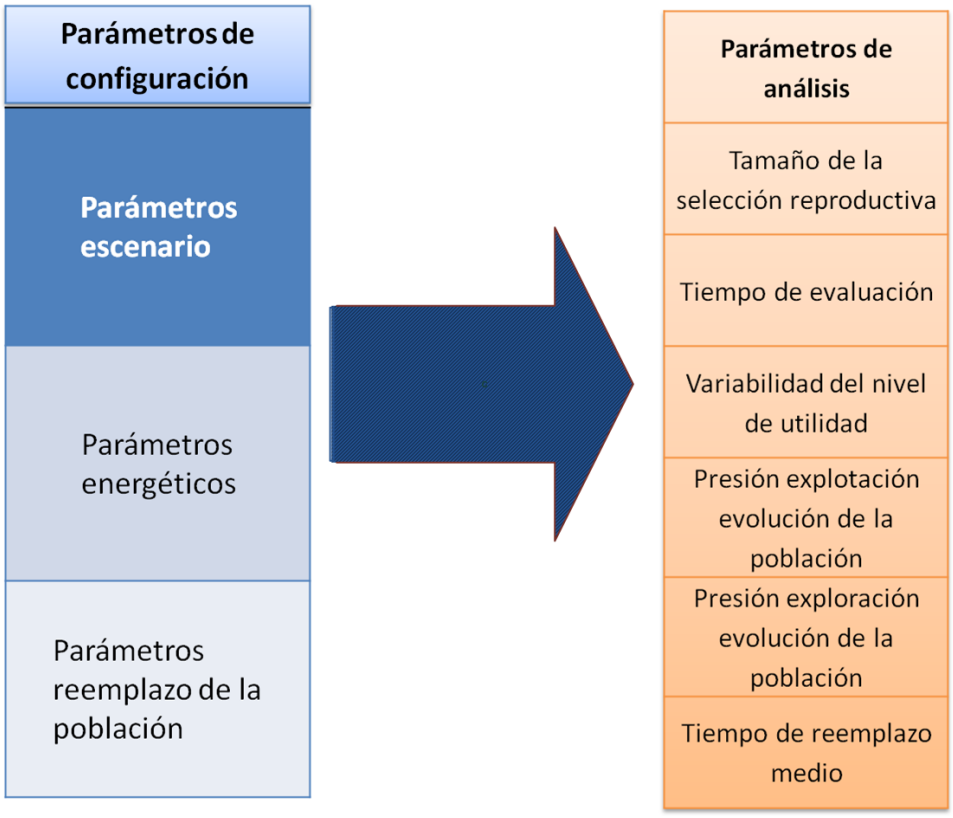


Figura 4.5: Esquema de la transformación paramétrica.

que le de soporte. Esta herramienta se ha denominada WaspBed (World-Agent Simulation Platform for BEhavior Design) y será presentada en detalle en el apartado siguiente.

4.2. WaspBed

4.2.1. Introducción

Para llevar a cabo el estudio y análisis de la evolución e interacciones de los componentes del modelo que define un problema real dinámico y descentralizado, es imprescindible el uso de simulaciones computacionales. Estas nos permiten realizar previsiones de la evolución de los mode-

los reales y comprobar el efecto de las variaciones que realicemos sobre distintos elementos o parámetros relevantes en el modelo en un tiempo razonable.

En nuestro caso, el uso de la simulación del escenario que represente el sistema dinámico en el que transcurre nuestra evolución es imprescindible dada la estructura que plantea el CeAS. La evolución de la población está ligada y es controlada mediante la evolución del escenario, es decir la simulación de las distintas interacciones previamente definidas para obtener los diferentes estados que atraviesa el sistema.

Para la realización de una simulación por ordenador, por un lado, se requiere un modelo, y por otro lado, debemos buscar unas herramientas que nos permitan introducir los parámetros de todos los componentes del mismo y modelar sus interacciones. En cuanto a esto, contemplamos dos opciones, o bien utilizar un lenguaje de bajo nivel para llevar a cabo esta codificación (programación directa en algún tipo de lenguaje, java, c++, etcétera) o bien utilizar alguna herramienta orientada hacia la simulación que nos permita utilizar un lenguaje de más alto nivel, y por tanto, nos facilite la implementación.

En el primer caso, se requiere generar una nueva aplicación para cada modelo que vayamos a estudiar. Se deduce con facilidad que esto presenta varios inconvenientes. En primer lugar, cada aplicación parte desde cero y eso aumenta considerablemente el esfuerzo que requiere estudiar un modelo, y por tanto, el tiempo necesario para generarlo. Por otro lado, en la mayoría de los modelados, tal y como hemos venido comprobando a lo largo del desarrollo de esta tesis, es frecuente que se planteen variaciones o adaptaciones al modelo a medida que se va analizando el mismo o realizando pruebas parciales de los distintos módulos. Muchas de estas variaciones no contempladas requieren de modificaciones y subterfugios costosos y poco eficientes sobre la aplicación inicial. Adicionalmente, es frecuente que tras haber realizado la aplicación y los análisis sobre la simulación, surja el interés en probar ciertas variaciones, introducir nuevos elementos u otro tipo de interacciones, por lo tanto es deseable que haya cierta flexibilidad y facilidad para realizar variaciones en la aplicación. Debemos destacar, sin embargo, que en el caso de que el modelo sea fijo y completamente definido desde el inicio, una aplicación desarrollada

únicamente para un modelo, será siempre más rápida y eficiente, es decir, requerirá menos coste computacional.

En el segundo caso, la implementación del modelo utiliza alguna herramienta de desarrollo de simulaciones ya existente, o bien se implementa una. La ventaja de utilizar un entorno de simulación es básicamente la facilidad y flexibilidad para generar un modelo nuevo o para realizar modificaciones del mismo. Adicionalmente, muchas de las herramientas de análisis, representaciones de datos, extracción de estadísticas, etcétera, son utilizadas en la mayoría de las simulaciones, y realmente resulta rentable implementarlas una sola vez.

En cuanto al uso de un entorno de simulación propio o uno ya existente, esta elección va a depender de si tenemos necesidades específicas que no encontramos en los simuladores disponibles y del nivel de control que requerimos de nuestras simulaciones. En el caso de un simulador propio, realizaremos un diseño del mismo siempre orientado a nuestras necesidades. En el caso de utilizar un simulador ya desarrollado, adaptaremos nuestras necesidades al simulador disponible.

Tras haber estudiado los simuladores existentes, en nuestro caso hemos decidido desarrollar uno propio, pues no hemos encontrado uno que se adapte completamente a nuestras necesidades ni que nos permita la suficiente libertad para crear modelos y herramientas. Hay que tener en cuenta que el algoritmo CeAS tiene unas características muy particulares que engloban requerimientos de diversas áreas, y además, el tipo de problemas a los que será aplicado está muy acotado. Esto hace que los simuladores generales que existen en la actualidad se alejen de nuestros requerimientos.

Por otro lado, el estudio de un algoritmo como CeAS requiere el conocimiento detallado de todos los parámetros que entran en juego así como su manipulación efectiva, resultando la implementación de un entorno de simulación propio una parte importante de dicho estudio.

Por ello, se ha desarrollado un interfaz de simulación denominado WaspBed (World-Agent Simulation Platform for Behavior Design) cuyos objetivos principales han sido:

- Permitir un proceso de definición de la simulación sencillo y flexible.
- Facilitar la realización de modificaciones en el entorno, en los elementos que constituyen el entorno, sus interacciones, y en las restricciones del mismo.
- Permitir un control completo y libertad para añadir nuevos componentes, reglas, interacciones y sistemas de control.

Basado en la arquitectura de gestión de la simulación de estos entornos se busca también establecer un formalismo a la hora de implementar los modelos, tanto para acelerar el proceso de configuración como para poder estudiar formalizar el análisis de las distintas simulaciones.

Adicionalmente, se han incorporado a la aplicación ciertas herramientas de análisis genéricas y configurables que facilitan la supervisión de los distintos parámetros que caracterizan la situación de cada entorno.

4.2.2. Algunos antecedentes

A continuación vamos a llevar a cabo una revisión de los principales simuladores para entornos dinámicos que hemos encontrado de cara a determinar la idoneidad de implementar nuestro propio simulador o utilizar uno de los ya existentes.

Pocos años después de la acuñación del término Vida Artificial y del comienzo de los estudios con respecto a este campo con la conferencia organizada por Langton y celebrada en los Alamos en 1987 (Langton, 1997), surgió uno de los simuladores más conocidos y que fue denominado *Tierra*. Este fue inicialmente desarrollado por el biólogo Ray (1992) y consiste en una simulación de un sistema que imitaba en gran medida los procesos evolutivos. En él, cada organismo es un programa con capacidad de autorreproducirse, encontrándose todos ellos en un recinto de memoria limitada. Cuando un programa encuentra una zona de memoria libre, copia su propio código e intenta así expandir los recursos de los que dispone para ejecutarse. Se obtuvieron resultados muy interesantes en cuanto a las estrategias de los códigos para perpetuarse.

Tras la creación de Tierra surgieron varios simuladores que explotaban la misma idea de agentes que tenían asignada una zona de memoria y que competían entre ellos para expandir y perpetuar su código genético. El simulador denominado *Avida* fue desarrollado por Ofría, Adami y Brown en 1993 en Caltech (Adami y Brown, 1994a). Estaba inspirado en el precedente de Tierra, aunque se diferenciaba en que, en Tierra, los programas compiten por recursos en las zonas de memoria, sin embargo, en Avida cada programa posee su zona de memoria asignada a la que no pueden acceder otros. Más adelante, Pargellis y Yurke (2000) crea el entorno de simulación *Amoeba* basado en la evolución de las moléculas de la sopa primigenia que dio lugar a los seres vivos actuales. Cada uno de los organismos artificiales ocupa una zona de memoria, posee una serie de instrucciones que ejecuta e intenta reproducirse. El simulador está inspirado también en su predecesor Tierra, pero la diferencia con este es que, en Tierra, el diseñador genera los agentes ad-hoc y en Amoeba estos emergen a partir de una serie de instrucciones aleatorias en una representación de esa combinación de moléculas inicial del referente biológico.

Cosmos (Taylor, 1997) es un entorno diseñado para estudiar la evolución entre entidades auto-replicantes que compiten por los recursos, que en este caso también están representados por espacio de memoria en la CPU, pero utilizan un enfoque distinto en cuanto al uso compartido de memoria y tiempo de proceso que el utilizado por el simulador Tierra.

Polyworld es un programa desarrollado por Yaeger (1993) para evolucionar Vida Artificial a través de selección natural y de algoritmos evolutivos. Los elementos que intervienen en la simulación son entidades que poseen forma trapezoidal y buscan comida, reproducirse y cazar. Trabajan con poblaciones de varios cientos de agentes que poseen un sistema de control basado en una red neuronal y en un proceso de aprendizaje Hebbiano. El código genético de cada uno de ellos determina, además de los pesos de la red neuronal, el tamaño, velocidad, color y algunos otros parámetros funcionales y morfológicos. En la misma línea encontramos otros muchos simuladores. como por ejemplo *Darwinbots*, que es un software de código abierto desarrollado por Comis (2003) y que genera un entorno virtual en el cual los agentes interactúan, pelean y finalmente se reproducen y evolucionan. También *Biotope* (Symeonidis *y otros*, 2005) permite simular un entorno formado por una malla bidimensional que de-

limita el escenario y en la que se pueden definir una serie de agentes con unas acciones determinadas para desplazarse e interactuar.

A pesar de la existencia de estos simuladores, que han dado muy buenos resultados, y de otros muchos que no entraremos a detallar, el tipo de entorno que necesitamos para la implementación del CeAS y su aplicación a diversos problemas reales difieren en gran medida del planteamiento de estos simuladores. Tanto los simuladores basados en la idea original de Tierra como los que permiten generar un entorno de Vida Artificial, están orientados a lo que hemos denominado la aproximación 'bottom-up' y están limitados a un ámbito muy concreto de simulación de un ecosistema virtual. El uso de estos simuladores para implementar los diferentes escenarios que representan problemas reales presentaría grandes incompatibilidades a la hora de adaptar las reglas de interacción de los elementos que define un escenario real a un escenario de simulación concreto y predefinido como el que utilizan.

Más en la línea del tipo de simulador que requiere la implementación del CeAS podemos encontrar, ya en el año 1989, un entorno desarrollado por Durfee y que se enmarca dentro de la disciplina de resolución de problemas distribuida, y que surge en su momento para estudiar comportamientos coordinados. El entorno se denomina *MICE* (Durfee y Montgomery, 1989) y permite la simulación de agentes sobre una malla bidimensional en la estos interactúan. Surge para acelerar el proceso de implementación de los distintos 'testbeds' que se realizaban en aquel tiempo para las tareas de coordinación, y permite cambiar las reglas e interacciones que afectan al grupo de agentes. El lenguaje en el que se programa es el LISP, y aunque poseía un diseño muy flexible y rápido para generar nuevos escenarios, no soporta una estructura de agentes, y por tanto, cada agente esta representado por una función del programa. Pronto esto creo una limitación para que se extendiese su uso.

Más recientemente encontramos varios simuladores que presentan características de gran interés para nuestras necesidades. Destacamos dos de ellos, en primer lugar, el entorno *Breve* fue presentado en 2002 por Jon Klein (Klein, 2002) como un entorno 3D para la simulación de sistemas descentralizados y vida artificial, y continúa en desarrollo. Permite la implementación de agentes autónomos, y para su programación se utiliza

un lenguaje que ellos han desarrollado específicamente denominado 'Steve' o bien el lenguaje Python. El entorno de simulación está enfocado a la simulación física de agentes en un entorno 3D. Es, en cierto modo, una combinación entre un simulador dinámico no completamente preciso pero sí lo suficiente para generar condiciones equivalentes a las reales, y un entorno multiagente que permite definir comportamientos a los elementos del entorno.

Netlogo (Sallez *y otros*, 2004) es otro simulador con la misma filosofía de Breve, basado en Java y que permite la generación de Applets. La primera limitación que encontramos es el lenguaje interno de NetLogo, el cual hay que utilizar para para desarrollar modelos basados en agentes. Aunque está basado en Java, el lenguaje de modelado es Logo, que está orientado a procedimientos, lo cual es totalmente diferente a paradigma actual de orientación a objetos que nos interesa. Por otro lado, no posee código abierto. Dispone de una gran cantidad de comandos y elementos para la creación de interfaces gráficas que no hacen sino complicar la generación de modelos que en muchos casos son muy simples.

Finalmente, dentro de la literatura de simuladores para sistemas multiagente, encontramos conjuntos de librerías de Java como las de Mason (Luke *y otros*, 2005) que implementan un conjunto de herramientas de gran utilidad para la generación de entornos de simulación como los que presentan Breve o NetLogo.

En resumen, diremos que a lo largo de los últimos 20 años han surgido un gran número de aplicaciones y simuladores para estudiar la evolución de sistemas dinámicos. En los primeros años, estos estaban principalmente orientados a estudios de síntesis de organismos vivientes, y por tanto, eran simuladores concebidos específicamente para determinados tipos de evolución. Más adelante, encontramos entornos que presentan una mayor libertad de actuación al diseñador para generar una gran variedad de simulaciones, como son Breve o NetLogo. Sin embargo, encontramos ciertos problemas como solución definitiva para implementar el CeAS: Breve proporciona gran libertad a la hora de generar interacciones y tipos de elementos, sin embargo, está fuertemente orientado a la simulación dinámica de agentes físicos y a la representación 3D, lo cual podría suponer una merma en la velocidad a la hora de generar simula-

ciones que no requieran un entorno tridimensional o interacciones físicas (simulaciones de colas de espera, redes eléctricas, etcétera). Por otro lado, el lenguaje y sintaxis propias para definir elementos e interacciones suponen otra desventaja. NetLogo posee también un lenguaje propio no orientado a objetos y aunque está entrando con fuerza en el campo de los Sistemas Multiagente aún no es de código abierto.

Por lo tanto, como ya se ha indicado, en este trabajo hemos decidido desarrollar una plataforma particular que hemos denominado WaspBed y que pasamos a presentar en detalle.

4.2.3. Estudio de requisitos

Basándonos en los objetivos que hemos comentado anteriormente, definimos las siguientes características que vamos a perseguir en la creación del WaspBed.

- *Sencillez en la implementación de modelos:* Waspbed ha sido diseñado originalmente para estudiar procesos de adaptación y aprendizaje, análisis de dinámicas de población, coevolución competitiva y comportamientos emergentes en entornos dinámicos. Tal y como hemos visto, las definiciones básicas, en la mayoría de los entornos que vamos a estudiar, están basadas o pueden ser adaptadas a una misma estructura. Esto nos va a permitir la creación de una plantilla válida para las diferentes configuraciones de cualquier entorno. De este modo guiaremos y simplificaremos el proceso de definición de las simulaciones basándonos en la estructura que está representada con esta plantilla.

- *Versatilidad:* uno de los objetivos de desarrollar nuestro entorno es el de comprobar los efectos producidos tras realizar determinadas modificaciones en el mismo: cambios en parámetros de configuración, variación del número de elementos de cada tipo, eliminar o añadir un grupo de elementos con nuevas funcionalidades durante el proceso de simulación, variaciones en las interacciones entre componentes o en las restricciones del entorno. Por otro lado, los mismos algoritmos de aprendizaje o estrategias de evolución, pueden ser probadas en problemas que van desde los autómatas celulares a problemas

reales de ingeniería. Debido a esto, minimizar el coste de implementar cualquier cambio es una prioridad en el diseño de la aplicación.

- *Alto nivel de control sobre las simulaciones:* la implementación de un entorno de simulación propio, permite que podamos enfocar la arquitectura del mismo a las necesidades que consideremos más relevantes. A pesar de la existencia de otras aproximaciones en el campo de la simulación de entornos dinámicos, estos estarán siempre o bien adaptados en mayor o menor medida a un modelo específico con una tipología de interacciones, o bien restringido al modo en que los autores definen, ejecutan y gestionan la simulación. Para obtener la libertad que necesitamos en cuanto a introducir cualquier componente, reglas, interacciones o sistemas de control es inevitable el desarrollo de nuestro propio interfaz de simulación. Por tanto, vamos a necesitar de un tipo de arquitectura que nos permita realizar modificaciones a bajo nivel cuando sea necesario, aunque hayamos implementado un lenguaje de más alto nivel, con acciones compuestas pre-implementadas.

- *Reutilización de herramientas:* el objetivo último de cualquier proceso que queramos simular será sin lugar a duda la parte que corresponde al análisis de las simulaciones llevadas a cabo. Con mucha frecuencia, en este tipo de Sistemas Complejos con multitud de interacciones, no es obvio el modo en el que extraer cierta información sobre lo que está sucediendo. Es por ello que otro de los objetivos es poder visualizar de forma sencilla diversos parámetros del modelo así como el desarrollo de un conjunto de herramientas para llevar a cabo la extracción de determinada información relevante de un modo fácilmente interpretable. La mayoría de los entornos dinámicos simulados requerirán de las mismas o similares herramientas de análisis, y es por eso que es interesante desarrollar un conjunto de herramientas de fácil uso y no dependientes del modelo para su posterior reutilización, y por tanto, mejora de la eficiencia en el análisis de modelos. Para permitir esto, debemos realizar un conjunto de herramientas que sean completamente configurables y permitan utilizar como parámetros de entrada los de cualquier tipo de componente de un modelo.

- *Reducción del coste computacional de la simulación:* muchos de los problemas que vamos a tratar, requieren de tiempos de cálculo elevados para obtener la siguiente iteración del modelo. Adicionalmente, en el caso de modelos con gran cantidad de componentes, el número de interacciones entre ellos resulta muy elevado. Sin ningún tipo de método de reducción, para un evento sencillo como el consistente en localizar al elemento más cercano, el número de operaciones crece con el cuadrado del número de elementos. Localizar un grupo de M elementos que minimicen una medida determinada (distancia entre ellos, diferencia de energías, etcétera) requeriría de un número de operaciones proporcional a NxM. Nuestra aplicación debe permitir de algún modo la paralelización de los distintos procesos de cálculo para, si fuese necesario, posibilitar la simulación distribuida en varios procesadores.

4.2.4. Diseño

El diseño que se presenta a continuación está basado en una idea principal: 'cualquier comportamiento complejo estará asociado a un entorno, a un conjunto de elementos que interactúan y a una serie de reglas que rigen estas interacciones'. Esto implica que, para caracterizar cualquier modelo basamos nuestra representación en tres elementos principales:

- El **entorno**, es el medio en el que se llevan a cabo las interacciones entre elementos y entre los elementos y el propio entorno. En un primer nivel, el entorno representa el subespacio de dimensión N, en el que las variables que intervienen en el modelo toman valores y que es común para todos los elementos. Esto se define implícitamente cuando modelamos los elementos del entorno, sus parámetros y los distintos rangos o restricciones en las que pueden tomar valores. En un segundo nivel, el entorno son todos aquellos elementos que no poseen ninguna variable que puedan modificar mediante acciones propias. Es decir, aquellos componentes que aunque no realizan acciones, sí intervienen en las acciones individuales o interacciones del resto de los elementos.

- Los **agentes**, son los elementos susceptibles de realizar modificaciones en sus parámetros por ellos mismos, bien mediante el uso de

un sistema de control que realiza ciertas acciones o bien mediante una o un conjunto de reglas de interacción fijas asociadas de las cuales son responsables. Es decir,los agentes son aquellos elementos que son capaces de actualizar sus propios parámetros. Una vez definidos los agentes y el entorno, se han caracterizado de un modo más formal los tipos de parámetros que intervienen en su definición como veremos más adelante.

- Las **interacciones**, representan cualquier modificación en las variables de los agentes o del entorno. El cambio de cualquier elemento del entorno se representa necesariamente mediante la variación de alguna de las variables del modelo. Estas variaciones las vamos a agrupar en lo que se denominarán interacciones, o a nivel de la arquitectura del Waspbed, *eventos*. Dentro de los eventos diferenciamos, con respecto al origen de la interacción, tres tipos.

  - **Acciones:** son aquellas variaciones que se producen en las variables de uno o más elementos del entorno debido a una decisión del sistema de control de algún elemento. Ejemplos de esto podrían ser los desplazamientos de un agente en el entorno, la liberación de feromonas, etcétera.
  - **Reacciones:** son aquellas variaciones que se producen como consecuencia de una regla del entorno, es decir, no son resultado de una decisión particular, sino de una característica del modelo. Ejemplos podrían ser el aumento de la edad de un agente con el paso del tiempo, el descenso del nivel de energía de un elemento cuando se sitúa en una zona determinada, el cambio de la posición de un elemento cuando cae bajo la fuerza de la gravedad, etcétera.
  - **Percepciones:** son aquellas acciones que modifican variables que denominamos sensores y cuya aplicación está limitada a servir de parámetros de entrada al sistema de control de un agente. Percibir implica lanzar todas las acciones de percepción existentes y llenar la memoria sensorial. Estas variables que denominamos sensores son siempre una combinación de otras variables, variables internas de un elemento (propiocepción) o variables externas (exterocepción). Podríamos incluirlas dentro de las reacciones, pero debido a que modifican unos

parámetros específicos y sólo afectan al sistema de control de los elementos consideramos adecuado clasificarlas separadamente.

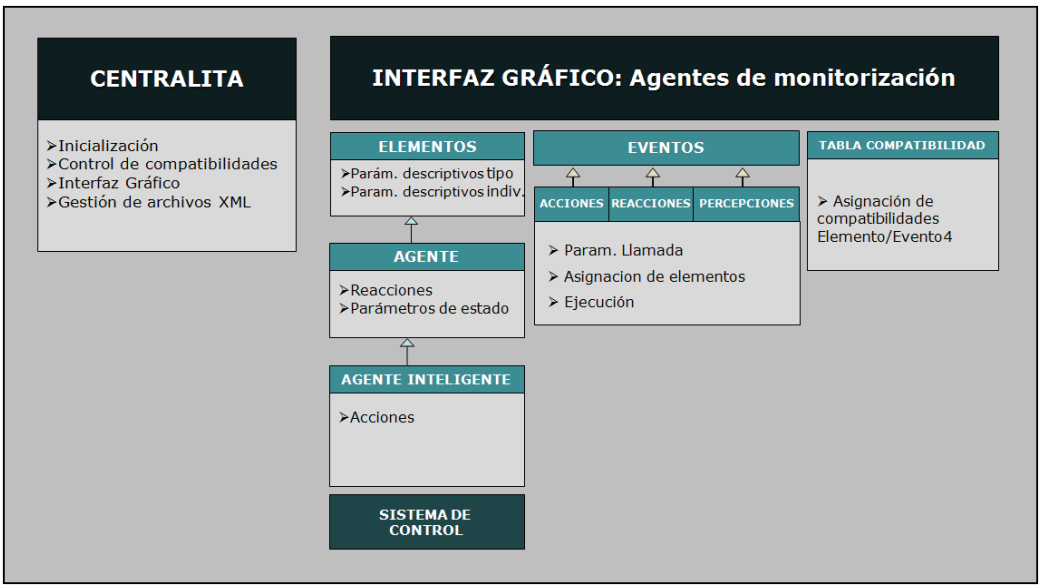


Figura 4.6: Esquema de diseño de la aplicación Waspbed.

Basándonos en la estructura que acabamos de presentar, la aplicación estará compuesta en líneas generales por los siguientes módulos, que se muestran en la figura 4.6.

1. *Centralita:* este módulo es el responsable de gestionar toda la aplicación. El acceso a los archivos de configuración, la inicialización y la conexión de todos los módulos, control de entradas externas, conexión con el interfaz gráfico, creación y eliminación de elementos, llamadas a los eventos y generación de las estructura de hilos en la que corren los procesos de la simulación.

2. *Elementos:* representan cualquier componente del entorno que contenga un conjunto de parámetros de caracterización. Cada uno de estos elementos pertenece a una categoría denominada *Tipo de*

Elemento. Las variables que contiene cada elemento son de tres clases diferentes. Variables que son comunes a todos los *Elementos* del mismo *Tipo de Elemento*, son los *Parámetros descriptivos de tipo*. Aquellas variables asociadas a una instancia concreta de un *Elemento* son los *Parámetros Descriptivos de Elemento*, y finalmente aquellas variables que pueden ser modificadas durante la simulación mediante eventos se denominan *Parámetros de Estado*. Dentro del grupo de elementos, llamaremos *elementos agentes* a aquellos elementos que poseen parámetros de estado, y llamaremos *elementos agentes inteligentes* a aquellos que además son capaces de realizar acciones. Es decir, poseen un sistema de control. Adicionalmente, los Elementos llevan asociados un grupo de parámetros no funcionales, estos no están definidos en el modelo ni tienen ninguna influencia en las interacciones entre ellos, son los parámetros gráficos y son, al igual que los sensores, combinaciones de otros parámetros. Sirven para definir la representación que se va a hacer de los elementos en el entorno y están formados por una etiqueta que indica el tipo de representación y un vector de valores que indican los parámetros que configuran esa representación.

3. *Eventos:* estos están asociados a un *Tipo de Elemento* y producen variaciones en los parámetros de estado de los elementos a los que afectan. La centralita se encarga de ejecutar los eventos asociados a cada *Elemento* cuando sea su turno. Dentro de cada Evento distinguimos tres componentes principales. Los parámetros de llamada, son aplicables solamente a las acciones y son aquellos parámetros de entrada que recibe el evento del sistema de control para modificar sus efectos. La asignación de elementos, es la parte del evento encargada de localizar o asignar los elementos que intervendrán en el evento y, por tanto, cuyos parámetros vamos a leer o modificar. Por último, la ejecución es la parte en la que se realizan las operaciones necesarias y finalmente se definen y ejecutan los cambios en los parámetros correspondientes.

4. *Plantillas de configuración:* estas son un conjunto de archivos xml que definen el entorno. Poseen una estructura prefijada para definir, para cada elemento, cada uno de los parámetros que los constituyen. Existen tres tipos de plantillas:

- *Los creadores*, que contienen la información de cuántos elementos de cada tipo se van a generar y los valores particulares de cada uno de los parámetros descriptivos de elemento y de los parámetros de estado necesarios en la inicialización.
- *Las definiciones de tipo*, en las cuales se definen los nombres y rangos de cada uno de los parámetros (parámetros de tipo, de elemento y de estado) y se definen los valores de inicialización por defecto.
- *La tabla de compatibilidades*, en la que se definen las asociaciones entre cada elemento y los eventos.

5. *Interfaz gráfico:* este se encarga de leer los parámetros gráficos de todos los agentes de la simulación y representarlos en el panel de simulación.

6. *Estructura de hilos de proceso:* esta estructura de hilos va a estar gestionada por la centralita y vamos a distinguir tres tipos de hilos principales:

   - El hilo de la centralita, que es hilo del programa principal y ejecuta las tareas asignadas a la centralita que ya hemos comentado: inicialización, comprobación de la existencia de elementos o eventos nuevos, gestión de entradas externas (entradas a través de menús, botones del panel, barras de deslizamiento o cuadros de texto del panel de simulación), etc.
   - Los hilos de los elementos, cada uno de estos hilos tiene asignados uno o más elementos, su función consiste en ejecutar secuencialmente para cada uno de sus elementos asignados todos los eventos compatibles con ese elemento.
   - Por último, el hilo del interfaz gráfico se encarga de ejecutar la actualización del panel de simulación de acuerdo a los valores actuales de cada uno de los parámetros gráficos de cada uno de los elementos.

A continuación, entraremos más en detalle para justificar cómo vamos a satisfacer cada uno de los requisitos de la aplicación que expusimos en el apartado anterior con el diseño presentado.

Procedimiento detallado

De cara a facilitar el proceso de implementación, se ha diseñado la aplicación de tal manera que la definición del modelo simulado se corresponde a un conjunto de objetos de dos tipos dentro de la estructura del Waspbed: elementos o eventos. Esto simplifica las consideraciones a la hora de modelar un problema o entorno. Adicionalmente, los elementos poseen una plantilla que guía al diseñador en el proceso de configuración de cada uno de ellos. El diseñador debe decidir en primer lugar qué parámetros de cada tipo va a poseer cada tipo de elemento. Los parámetros en Waspbed son siempre un par (nombre, vector de valores) y cada uno de los valores posee un rango de funcionamiento en el que este parámetro puede tomar valores y un rango de inicialización en el que se generan aleatoriamente los valores iniciales.

El diseñador debe decidir, a continuación, los rangos de cada parámetro. En el caso de los parámetros de estado, el rango de cada parámetro funciona como una restricción, cuando un evento intenta asignar a un parámetro un valor fuera de su rango, este parámetro mantiene su valor anterior. En el caso de un parámetro descriptivo de elemento, la variación se produce cuando se genera un nuevo elemento durante una simulación y los nuevos parámetros se definen en la propia simulación (procesos de reproducción, etcétera), en el caso de los parámetros descriptivos del tipo de elemento estos no requieren de un rango. Por último el diseñador puede establecer que los parámetros tomen valores continuos, o discretos, establecimiento el paso de cada parámetro de modo que los parámetros sólo pueden tomar valores que sean múltiplos del *paso* ($nuevo\_valor = k * paso$).

Posteriormente, el diseñador debe generar los eventos, para ello Waspbed dispone de una clase Evento de la que heredan todos los nuevos eventos que se vayan a crear. Esta clase evento representa también una plantilla a la que deben adaptarse los diseñadores para crear estas interacciones. Los parámetros que reciben todos los Eventos por defecto son: una referencia al Elemento que los ha llamado, el valor del tiempo en el instante en el que fueron llamados y un vector de parámetros de entrada. Adicionalmente, existe una clase estática denominada *Gestor de Eventos* que es la que se encarga de devolver una lista con todos los ele-

mentos de un tipo determinado que estén activos en ese momento. Con estos parámetros de entrada y el acceso al Gestor de Eventos, podemos configurar cualquier interacción entre elementos. En la primera parte del evento, el diseñador debe implementar la fase de asignación de elementos en la que se seleccionan los elementos que cumplan unas condiciones específicas para formar parte de la interacción. En la segunda fase se realizan las operaciones necesarias para generar el conjunto de variaciones que se van a llevar a cabo en ciertos elementos de los seleccionados y se ejecutan esas modificaciones.

Para definir los agentes inteligentes, es decir, los agentes con su sistema de control, el diseñador debe programar el funcionamiento y actualización de este sistema de control. Los sistemas de control se implementarán en un evento de una clase especial denominados EvTool (Eventos Herramienta) y se llamarán desde una reacción denominada generalmente *Decidir* (ejemplos de sistemas de control son una red neuronal, un sistema de reglas, etcétera). *Decidir* implica, por lo tanto, ejecutar el algoritmo del decisor para realizar una acción con unos parámetros de entrada determinados. El sistema de control modifica también un tipo especial de parámetros de estado que se incluyen en lo que se denomina *memoria interna* y a los que sólo el sistema de control accede para modificar o para leer. Estos parámetros representan información histórica que puede ser relevante para decisiones futuras. A nivel de implementación no difieren de otros parámetros de estado, pero a nivel conceptual sí representan una categoría distinta, y por tanto, a la hora de diseñar es de utilidad su diferenciación para guiar el proceso de definición de cada agente inteligente. En el caso de que el sistema de control sea muy sencillo (p. ej. una función de ponderación con una serie de coeficientes para elegir entre varias opciones), el sistema de control podrá ir incluido en la acción de Decidir para rebajar el coste de la llamada a otro evento.

En cuanto a flexibilizar las modificaciones en las simulaciones implementadas, el Waspbed permite: realizar cambios en los agentes modificando rangos de parámetros, añadiendo nuevos parámetros o eliminando otros cambiando los archivos de configuración *XML*, añadir nuevas asociaciones entre eventos y elementos o eliminar algunas de la existentes modificando el archivo *XML* de la tabla de compatibilidad. Es posible también añadir elementos nuevos durante la simulación sin tener que dete-

nerla, introduciendo un nuevo archivo creador si son elementos de un tipo ya especificado, o un *archivo de definición de tipo* junto a un archivo creador en las carpetas correspondientes. Del mismo modo, si generamos a partir de una nueva clase de Evento un *archivo precompilado ċlass* y lo situamos en la carpeta de eventos, este pasará a estar disponible para los elementos de la simulación. Por lo tanto, Waspbed permite tanto una modificación flexible de cualquier simulación ya diseñada como el añadir elementos, eventos o asociaciones entre ellos en tiempo de ejecución.

Con respecto a tener control total y libertad para generar y representar cualquier tipo de modelo, esto es una consecuencia casi directa de haber implementado el entorno de simulación, puesto que cualquier necesidad que implique algún cambio estructural podría realizarse, con mayor o menor facilidad, pero más directamente que usando una aplicación externa. A mayores, tal y como se ha definido esta arquitectura, es posible combinar la definición del entorno a bajo nivel, en cuanto a definir o modificar los eventos generados, paso a paso, con un lenguaje de más alto nivel haciendo uso de herramientas ya desarrolladas.

Las herramientas son un caso particular de los eventos, estas heredan de la clase *EvTool* y se llaman desde los Eventos. El sistema de control es, a nivel de arquitectura del Waspbed, una *EvTool* y se llama desde una reacción denominada *Decidir*. El evento de eliminar un elemento, el evento para generar un nuevo elemento, el evento de reproducción, el evento de localizar un elemento cercano (para una distancia definida como la norma de una variable configurable), el evento que calcula la distancia en un entorno toroidal. . . son ejemplos de algunos de los *EvTool* a los que se puede acceder desde cualquier evento. Y que por tanto son reutilizables entre distintas simulaciones.

En cuanto a las herramientas de análisis, estas están integradas en Waspbed mediante lo que se llaman ‘agentes de monitorización’. Estos agentes sólo realizan percepciones, es decir que no afectan a ningún parámetro que no sea suyo, se limitan a leer los parámetros de otros elementos y generar valores nuevos para sus parámetros sensoriales como combinación de aquellos. A partir de los parámetros gráficos de cada agente de monitorización, se representan en el panel de simulación estas variables de monitorización que pueden incluir desde medias, desviacio-

nes, medidas locales, hasta operadores más complejos como medidores de la variabilidad de una población, etcétera.

Para conseguir la paralelización, Waspbed ejecuta una estructura de hilos que permite que la carga de cómputo de toda la simulación esté distribuida entre estos hilos. La centralita que gestiona toda la aplicación, incluso la estructura de hilos de los elementos y del interfaz gráfico, se ejecuta en un hilo independiente. El hilo de la centralita no posee mucha carga de trabajo pero debe ser independiente para no estar supeditado al transcurso de los eventos de la simulación, y por tanto, no ser bloqueado por estos. El hilo del interfaz gráfico se encarga de leer el estado actual del entorno y representarlo. Waspbed permite modificar la velocidad de los hilos del interfaz gráfico y de los hilos de ejecución para así conseguir una visualización con mayor o menor velocidad de refresco, y por tanto mayor o menor prioridad al cálculo de la simulación, gracias a esta capacidad no necesitamos ejecutar la simulación sin interfaz gráfico para acelerarla, basta configurar un tiempo de refresco de la visualización lo suficientemente bajo como para que el tiempo de cálculo perdido en el proceso sea despreciable frente al tiempo de cálculo de los hilos de ejecución.

Los elementos se distribuyen en hilos de ejecución y debido a que cada elemento tiene asociados una serie de eventos toda la carga de trabajo que implica resolver la física/reglas del mundo se reparte entre los agentes. Es importante para el diseñador distribuir de forma equilibrada las asignaciones de las reacciones entre todos los elementos si se busca una buena paralelización. Cada uno de los hilos de ejecución de los elementos se encarga de ejecutar de forma sucesiva las Percepciones, Acciones y Reacciones de cada uno de los elementos que tiene asociados. Estos hilos se pueden distribuir de forma sencilla en varias CPU para disminuir el tiempo de simulación.

4.2.5. Implementación

La implementación del sistema Waspbed se ha realizado en JAVA. La decisión de utilizar este lenguaje se ha basado en principalmente en los siguientes motivos: en primer lugar, el lenguaje JAVA permite un uso multiplataforma de nuestra aplicación, lo cual es realmente interesante de cara a lanzar las simulaciones en clusters de procesadores independientemen-

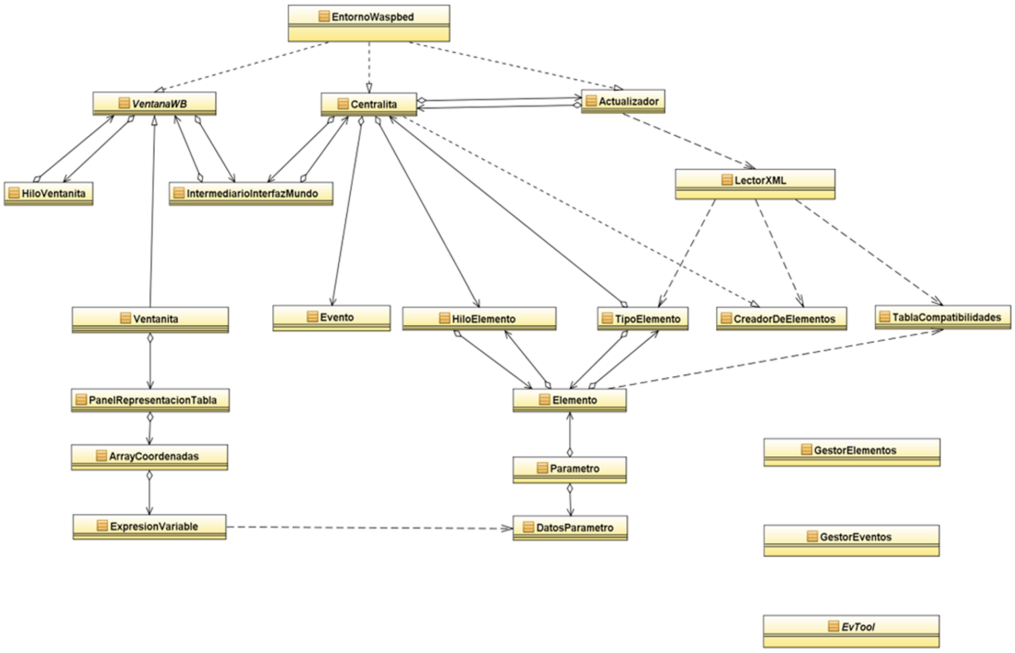


Figura 4.7: Diagrama de clases de la aplicación Waspbed.

te del sistema operativo sin tener que recompilar la aplicación. También nos permite la integración del simulador con otros interfaces, como por ejemplo un simulador dinámico comercial, dispositivos externos (cámarás, sensores de luz, etcétera) independientemente de la plataforma utilizada. Otro de los motivos es que JAVA es un lenguaje basado en objetos, lo cual permite traducir, de forma casi directa, el modelo de mundo que queremos simular al modelo de mundo utilizado en la aplicación. Por último, de cara a un uso en un navegador web de la aplicación, JAVA permite la generación de applets los cuales están incorporados en los principales navegadores actuales.

El diagrama de clases de la aplicación Waspbed se muestra en la figura 4.7. La descripción detallada del funcionamiento de cada una de las clases puede consultarse en el apéndice I.

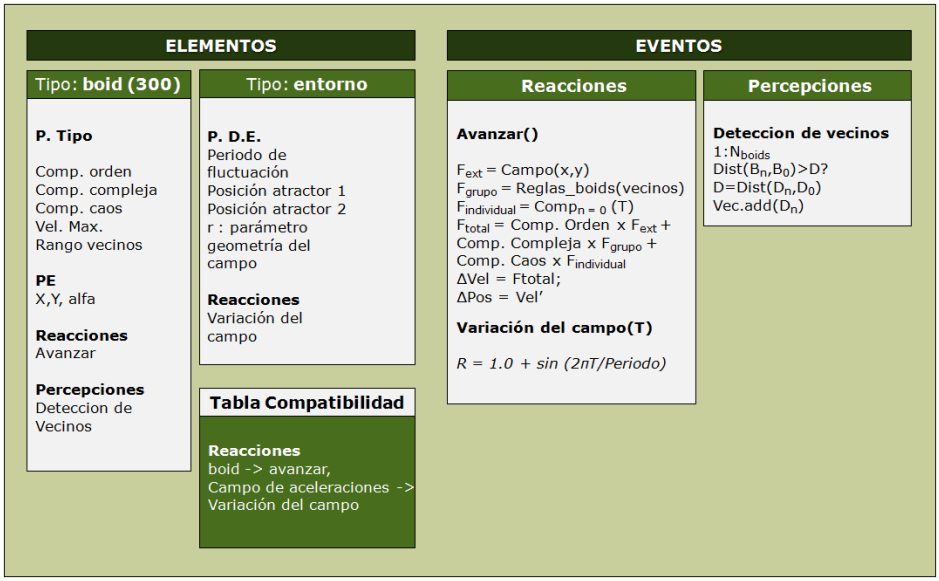


Figura 4.8: Plantilla de creación de una simulación.

4.2.6. Comprobación de funcionamiento básico

Para llevar a cabo la comprobación de que el WaspBed funciona correctamente según el diseño e implementación anteriores, se ha creado un escenario simple compuesto por dos tipos de elementos. Un primer elemento que denominaremos *boid*, por estar inspirados en los famosos boids de Reynolds (Reynolds, 1987), y que van a desplazarse por el escenario, y un segundo elemento que será el *entorno* y que generará un campo de aceleraciones al que estarán sometidos los boids. La plantilla de definición para este escenario puede verse en la figura 4.8.

El objetivo de esta prueba es comprobar el funcionamiento correcto de los distintos módulos del WaspBed. Para ello este escenario, aunque simple, comprenderá todas las funcionalidades del mismo en cuanto a la inicialización de varios tipos de elementos, generación de acciones que involucren a elementos de distintos tipos, estudio de la representación gráfica y sus capacidades de configuración, etcétera.

Los elementos que denominamos boids, estarán sujetos a la acción de tres componentes que guiarán su movimiento. Esto se hará de cara a representar las tres componentes que intervienen en un sistema complejo (Cotsaftis, 2006) y que son la componente individual, la externa y la colectiva. Estas tres componentes están representadas por la implementación en la acción *avanzar* de tres fuerzas que afectan a los boids. Una primera fuerza es la debida a un campo de aceleraciones externo que es generado por el elemento entorno y que varía con el tiempo. Este campo tiene la forma que podemos ver en las figuras 4.9 y 4.10.

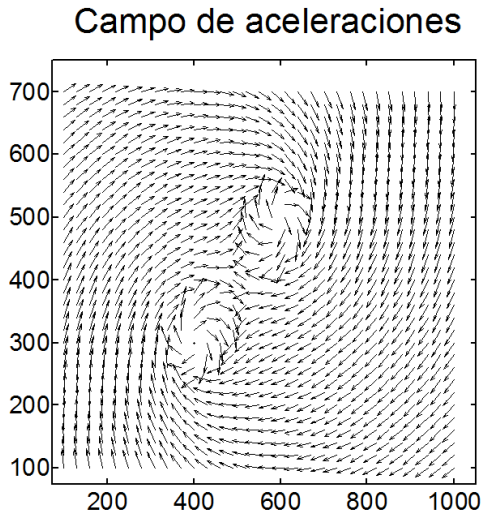


Figura 4.9: Campo de aceleraciones generado por la componente externa en el estado 1 ($p = 1$).

La segunda componente es debida a un conjunto de reglas de interacción local entre los distintos boids, y sigue la misma estructura que las reglas de los boids de Reynolds (1987). Estas reglas se basan en la velocidad y posición de los vecinos de cada uno de los boids y generan un comportamiento de movimiento en bandada.

Concretamente, las reglas son las siguientes y están representadas en la figura 4.11:

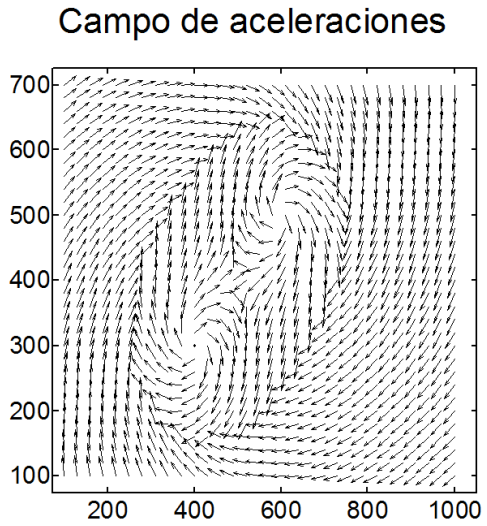


Figura 4.10: Campo de aceleraciones generado por la componente externa en el estado 2 ($p = 2{,}8$).

- Una fuerza de atracción que dirige al boid hacia el centroide de la posición de los vecinos haciendo que la bandada se agrupe.
- Una fuerza que tiende a igualar la velocidad a la de los vecinos de manera que la bandada se mueva uniformemente.
- Una fuerza de separación que impide que los boids se acerquen demasiado.

Y por último, cada uno de los boids posee un comportamiento individual que hace que siga una trayectoria variable con el tiempo y que es diferente para cada boid. Esta sería la componente individual del movimiento. La velocidad en cada instante viene definida por la siguiente ecuación:

$$\overrightarrow{F_i} = \left(A_i \sin\left(\frac{2\pi T}{P_i}\right), B_i \sin\left(\frac{2\pi T}{Q_i}\right)\right) \tag{4.5}$$

donde los coeficientes A_i, B_i, P_i y Q_i se generan aleatoriamente para cada uno de los individuos cuando se crean en la inicialización.

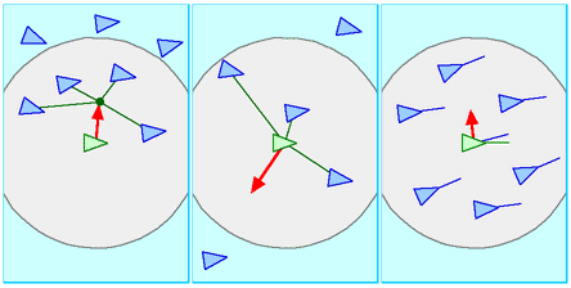


Figura 4.11: Representación de las reglas de atracción, velocidad y separación de los boids.

Estas tres componentes, fuerza externa, fuerza colectiva y fuerza individual, representan respectivamente comportamientos ordenados, complejos y caóticos. El comportamiento resultante que poseerán los boids será una combinación de estos tres y estará dado por los coeficientes de orden, caos y complejidad que posee cada uno de ellos.

El otro elemento denominado entorno es el encargado de generar el campo de aceleraciones y producir variaciones en él a lo largo del tiempo. Básicamente, este campo de aceleraciones posee dos puntos atractores que definen una orbita estable. Esta órbita (P) la constituye el lugar geométrico de los puntos que cumplen que el producto de las distancias a estos dos puntos es constante.

$$x \in P \Leftrightarrow d(x, P_1) \cdot d(x, P_2) = K \tag{4.6}$$

donde el valor de la constante K es el parámetro que el entorno hace variar con el tiempo mediante la reacción *Variación del campo*.

Para llevar a cabo la representación, se generaron unos polígonos cuya posición y orientación es la posición y orientación de cada boid. Adicionalmente, se probó la representación del módulo de la velocidad de cada boid mediante el color con el que se representan. Obteniendo así una visualización de este parámetro. Se probaron diversas configuraciones inicializando una población de 300 boids con diferentes valores de los coeficientes de las componentes del movimiento.

En la figura 4.12 vemos algunas capturas de la simulación obtenida, donde se representan distintos estados de los boids. La imagen a muestra la situación inicial con una dispersión aleatoria de posiciones y velocidades para los boids, la b muestra el estado resultante de combinar las componentes externa e individual, es decir, los boids tienen tendencia a seguir la dirección marcada por el campo de fuerzas externo pero poseen otra componente de comportamiento individual independiente que genera en el comportamiento colectivo cierto desorden, en la captura c y d podemos apreciar el comportamiento determinado solo por la componente externa en dos instantes que corresponden a los valores del parámetro $p = 1{,}0$ y $p = 2{,}8$, como se comprueba todos los elementos siguen la tendencia marcada por las lineas de campo. La captura e muestra una combinación de componente externa, y colectiva lo cual se traduce en un comportamiento de agrupación generado por las reglas de los boids mientras se sigue la trayectoria de la fuerza externa y por último en f vemos el estado resultante con sólo componente colectiva que es equivalente al movimiento en bandada definido por las reglas de lo boids. Este ejemplo nos sirve para comprobar mediante el uso de estos comportamientos ya conocidos que la reglas de interacción en estos tres niveles predefinidos están siendo interpretadas correctamente.

Comprobamos también, que todos los módulos funcionan correctamente, la representación gráfica, la creación de elementos y eventos, los archivos de inicialización, configuración y definición se leen e interpretan de forma correcta, y por tanto, este ejemplo nos sirve para probar el funcionamiento básico de la aplicación.

En los siguientes capítulos se llevará a cabo un estudio detallado del CeAS en 3 problemas dinámicos descentralizados de ingeniería. En estos escenarios, creados para estudiar el CeAS, se realizará también el testeo de algunas otras funcionalidades del WaspBed que puedan surgir al implementar un sistema de control en los agentes y crear escenarios más complejos y con mayor número de interrelaciones.

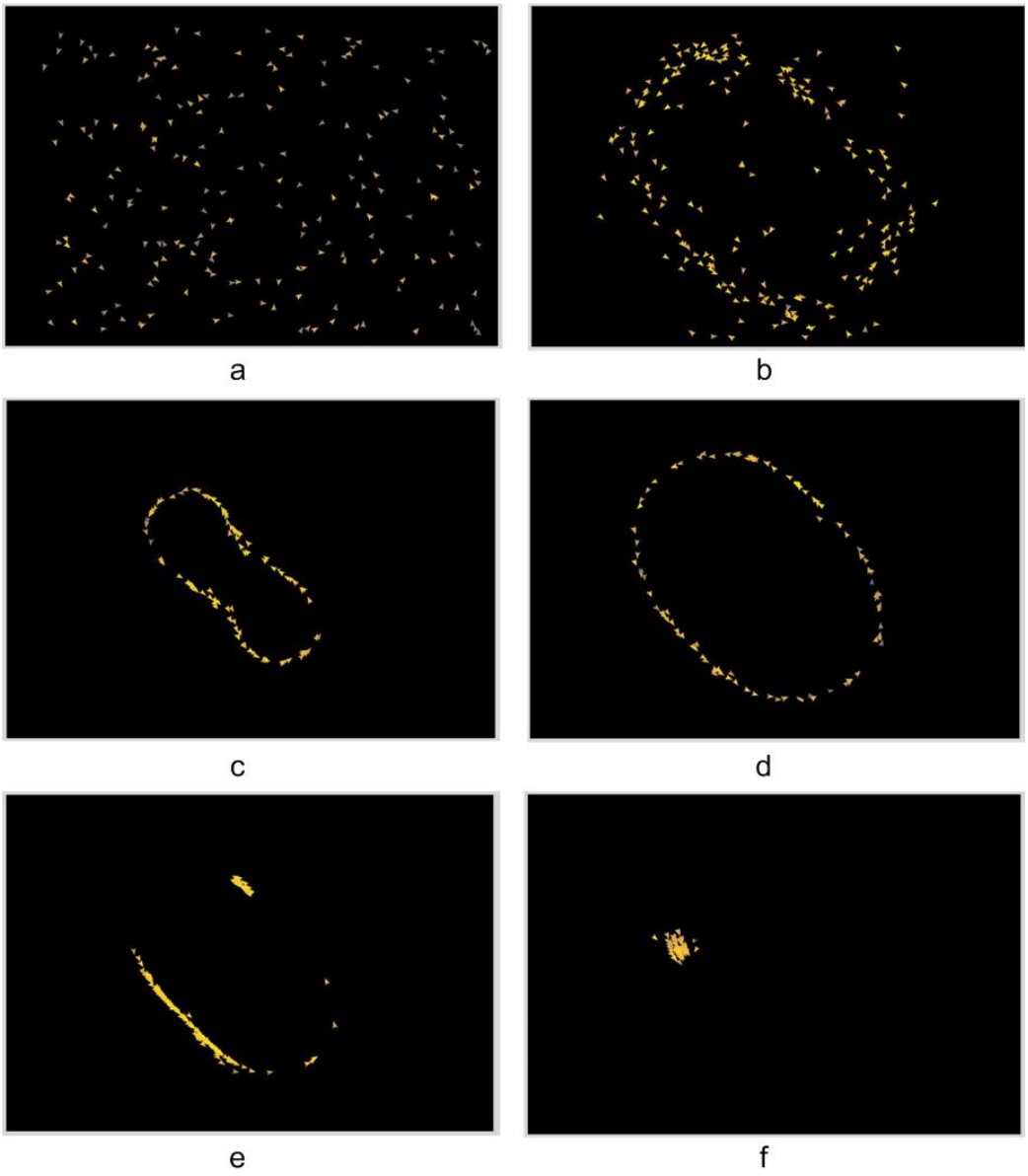


Figura 4.12: Representación de boids en distintos estados durante la simulación. a: Situación inicial; b: Combinación de componente externa e individual; c: Sólo componente de fuerza externa $p = 1{,}0$; d: Sólo componente de fuerza externa $p = 2{,}8$; e: Combinación de componente externa y colectiva; f: Sólo componente colectiva, comportamiento de bandada obtenido mediante las reglas de los boids de Reynolds.

Capítulo 5

Estudio del CeAS en un problema de ruta óptima

5.1. Introducción y Objetivos

El primer problema dinámico descentralizado que vamos a utilizar para estudiar las capacidades del CeAS, y que nos servirá también para profundizar en el uso de la herramienta de simulación WaspBed, será el problema de encontrar una ruta óptima, uno de los problemas prototipo en el campo de la Investigación Operativa. El objetivo es encontrar la mejor ruta posible en un una red de rutas interconectadas. Dependiendo del caso concreto, se establecen distintos valores para las variables que se deseen maximizar. El caso más típico de problema de ruta óptima es encontrar encontrar el camino más corto entre dos puntos de grafo. En estos casos, la magnitud a minimizar es la distancia entre esos dos puntos, pero se podrían utilizar otras como el tiempo, la longitud de los tramos, etc.

Utilizaremos el CeAS con el objetivo de encontrar un *algoritmo genérico para la búsqueda de rutas óptimas*, es decir, la tarea objetivo final será la de desarrollar un algoritmo capaz de llevar a cabo el proceso de búsqueda no para un grafo concreto, sino para cualquier grafo con una

serie de características dimensionales y de complejidad dentro de un rango que hemos utilizado para acotar el alcance del problema.

Este problema que vamos a estudiar en primer lugar cubre varios objetivos dentro del marco de esta tesis. Principalmente, con respecto al algoritmo CeAS, estudiaremos los problemas y consideraciones necesarias para la parametrización y diseño de un escenario de simulación real a partir de un problema modelo. Se validará la metodología de diseño presentada en el apartado 4 para la adaptación de un problema dinámico descentralizado al dominio del CeAS, y de este proceso extraeremos las conclusiones que encontremos pertinentes para mejorar la transformación, el proceso de diseño y la configuración del algoritmo.

Por otro lado, se comprobará la capacidad del CeAS para evolucionar una población de agentes de modo que su comportamiento constituya una solución para la realización de una tarea o la optimización de una variable objetivo, en este caso, la resolución descentralizada de un problema de búsqueda en un espacio definido por un grafo. Es decir, debemos comprobar que realmente el algoritmo permite que una evolución abierta como la que se plantea esté guiada hacia un objetivo de diseño.

Finalmente, servirá para testear el funcionamiento del entorno de simulación WaspBed en cuanto a la implementación del procedimiento de evolución de una población planteado en CeAS. La generación y eliminación de nuevos elementos en tiempo real en la simulación, los agentes de monitorización y la extracción de datos de análisis a partir de la evolución de la población y del escenario, son los principales puntos a analizar.

El problema de encontrar una ruta óptima se puede definir como: dado un grafo formado por *un conjunto de nodos y por un conjunto de aristas que interconectan algunos de esos nodos y que tienen asignado un valor (peso, longitud, etcétera), y fijados un nodo origen y un nodo destino, encontrar el camino entre el origen y el destino de tal manera que la suma de los pesos de cada una de las aristas que forman dicho camino sea la mínima de todas las combinaciones de aristas que unen el origen y el destino*.

Formalmente, dado un conjunto de vértices V y un conjunto de aristas A sea:

- $g : A \rightarrow \mathbb{R}$ una función que relaciona a cada una de las aristas con un valor real.

- M el conjunto de todas las combinaciones de aristas que unen dos nodos fijos del conjunto de aristas A (el nodo origen y el nodo destino).

Encontrar la combinación c tal que

$$c = min \sum_{c \in M} g(a_c) \tag{5.1}$$

El problema que hemos descrito se plantea como un problema con una única tarea objetivo no distribuida. Esta tarea de optimización puede ser llevada a cabo por un único agente con algún tipo de heurística constructiva o algoritmo de exploración. Por otro lado, de acuerdo a lo que hemos expuesto en el apartado 1, el tipo de problemas que entran dentro del alcance de la técnica que exponemos en esta tesis es aquel en el que las interacciones entre elementos constituyentes y los flujos de información son parciales y locales, y por tanto, descentralizados. En este sentido, podemos encontrar en la literatura numerosas aproximaciones para la resolución distribuida de este problema mediante el uso de poblaciones de agentes de exploración con capacidades limitadas de sensorización y de comunicación inspiradas en comportamientos de colonias de hormigas (Stützle y Hoos, 2000; Schoonderwoerd *y otros*, 1996; Costa y Hertz, 1997). Una de las más relevantes la constituye el algoritmo ACO (Ant Colony Optimization).

El algoritmo ACO fue definido por primera vez por Dorigo *y otros* (1996) y en él se indica que la tendencia de una hormiga para seleccionar un camino se realiza en función a la siguiente expresión:

$$p_{ij}^k(t) = \begin{cases} \dfrac{\left(\tau_{ij}(t)\right)^{\alpha} \cdot \left(\eta_{ij}\right)^{\beta}}{\sum_{k \in allowed_k} \left(\tau_{ij}(t)\right)^{\alpha} \cdot \left(\eta_{ij}\right)^{\beta}} & \text{si } j \in \text{ allowed}_k \\ 0 & \text{en otro caso} \end{cases} \tag{5.2}$$

En ella intervienen los 3 parámetros siguientes: τ que indica la concentración de feromonas, η que representa la inversa de la longitud de una arista y $allowed_k$ que representa la lista de todas las aristas exceptuando aquellas consideradas tabú por haber sido visitadas en un plazo menor a los k pasos anteriores.

Basándonos en el modelo que plantean estos trabajos hemos decidido generar un escenario para evolucionar una población de agentes de manera que ésta constituya un algoritmo de búsqueda de ruta óptima en grafos consiguiendo, de esta manera, transformar el problema inicial en un problema distribuido para el que generaremos una solución basada en un algoritmo no centralizado.

En cuanto al proceso que vamos a llevar a cabo para la definición e implementación de este problema en CeAS, y de acuerdo a la metodología establecida en el capitulo anterior, este se compondrá de:

1. La definición del entorno, los agentes y sus interacciones para que los comportamientos que se puedan generar en el escenario supongan soluciones al problema buscado.

2. La definición de la función de utilidad individual que utilizaremos para evaluar a cada agente teniendo en cuenta la utilidad global que define el problema para todo el conjunto.

3. La definición del modo en que la evolución de la población se dirige hacia comportamientos que optimizan nuestra función objetivo, basándose en la utilidad individual propuesta y mediante la definición de las condiciones que regulan:

   a) El proceso de selección y evaluación para la generación de nuevos individuos basados en individuos existentes.

   b) La asociación entre la tarea objetivo y tanto la utilidad asociada a ese individuo como su supervivencia en el entorno (tiempo de vida).

4. Asegurar que aunque el sistema posea un punto de equilibrio sobre la solución deseada, no existan zonas (o que sean poco frecuentes) que hagan divergir la dinámica del sistema (extinciones), y que si el

sistema se dirige hacia la solución, lo haga de forma estable, y por tanto, no fluctúe en torno a esa solución (control de la evolución y dinámicas de población).

En los siguientes apartados explicaremos, primero de un modo más exhaustivo, cómo ha sido el proceso de diseño del escenario para este ejemplo siguiendo la estructura propuesta por la metodología del CeAS. A continuación se expondrá cómo se ha realizado la implementación en el WaspBed mediante el uso de las plantillas de parametrización y de definición de los elementos del escenario y de sus interacciones. Posteriormente se mostrarán los resultados obtenidos para distintas configuraciones del escenario, y finalmente, se expondrán una serie de consideraciones con respecto al análisis energético que han surgido tras el estudio en profundidad de esta aplicación concreta pero que serán extrapolables a muchas otras, y que por tanto, pasarán a formar parte de la metodología del CeAS.

5.2. Configuración experimental

El escenario que nos planteamos consiste en un mundo de grafos con poblaciones de agentes que los recorren y que se ven beneficiados energéticamente por el hecho de alcanzar un punto objetivo y volver al origen (que son los dos puntos que definen el principio y el final de la ruta cuya longitud queremos minimizar). Este mundo de grafos va modificándose con el tiempo ofreciendo distintos escenarios de grafos a la población, con el objeto de que no se especialicen en un tipo de grafo concreto y que el sistema de control distribuido resultante pueda tomarse como un algoritmo distribuido para la resolución del problema. De hecho, postulamos la hipótesis de que los sistemas de control de los agentes individuales deberían tender a ser iguales, constituyendo un conjunto de agentes homogéneos, dado que no parece haber, en principio, ninguna razón que lleve a diferenciar especies, esto es, a hacer unos individuos diferentes a otros.

En este apartado se describen todos los elementos que van a constituir el escenario de esta primera aplicación y cuáles son sus interacciones. En este caso, estos elementos se reducen a los agentes de búsqueda y a los

grafos, y sus interacciones serán aquellas que permitan el desplazamiento de los agentes a través de los grafos, la comunicación entre agentes y la sensorización.

Definición de los individuos

Con respecto a la definición de los individuos vamos a describir fundamentalmente los sensores que poseen, las acciones que pueden ejecutar, las características del sistema de control y los parámetros evolucionables que forman su genotipo.

De la descomposición que hemos hecho de la tarea de exploración de un grafo en una tarea distribuida en agentes independientes y heterogéneos, extraemos que la transformación más directa es la de asignar un individuo a cada uno de los agentes de exploración. Funcionalmente, estos agentes serán homogéneos, pero la estrategia que los guía a través de los grafos y que está representada por su sistema de control, será propia de cada uno de los agentes, e inicialmente generada de forma aleatoria.

Cada uno de los agentes debe poseer sensores que proporcionen información relativa al entorno, información relativa a los otros agentes, información interna y algún tipo de sistema de comunicación para compartir información con otros agentes. Esto nos permitirá que los comportamientos puedan poseer las tres componentes que generan la interacción dinámica presente en un sistema complejo (Cotsaftis, 2006): el *comportamiento individual*, que es generado por parámetros propios de cada agente bien intrínsecos o bien adquiridos durante la simulación, el *comportamiento colectivo* generado por la influencia mutua entre agentes y el *comportamiento externo* generado por parámetros propios del entorno en el que se encuentran los agentes en cada momento.

Concretando, los sensores que se proporcionan a cada uno de los agentes son los siguientes:

1. **Sensor de agentes**, que proporciona el número de agentes que están viajando cada una de las posibles futuras aristas.

2. **Sensor del rastro de feromonas**, que obtiene el valor del parámetro del nivel de feromonas para cada arista.

3. **Sensor de longitud**, que nos indica el coeficiente de longitud o costo asociado a cada arista.

4. **Sensor de proximidad temporal**, que indica cuanto tiempo hace que un agente pasó por última vez por cada una de las posibles futuras aristas.

Para este último sensor, los agentes poseen una *memoria de itinerario* asociada que almacena su itinerario mediante una reacción. Esta reacción tiene como objeto actualizar la memoria del itinerario, y cada vez que un agente abandona una arista, esta arista se añade a la memoria.

Debido a que la solución perseguida se representa como una ruta a través de las aristas de los grafos, los movimientos de los agentes se restringen a desplazarse de vértice a vértice a través de las aristas. Cada vez que alcanzan un vértice, deciden cual será la próxima arista a través de la que van a desplazarse. En el intervalo de tiempo durante el cual están recorriendo una arista no se les permite llevar a cabo ninguna acción que no sea la de avanzar en dirección a su vértice destino, y sus reacciones (acciones no generadas por el sistema de control) son las asociadas al consumo energético por unidad de tiempo y el incremento del marcador de tiempo de vida que más adelante explicaremos.

Para desplazarse por el entorno se implementa el evento que se denomina *avanzar*: cada agente avanza una distancia a lo largo de la arista en la que se encuentra situado. Esta distancia es el resultado de multiplicar un valor fijo y común para todos los agentes denominado $paso\_avance$ por la *inversa del coeficiente de tamaño aparente* del grafo en el que se encuentra. El paso avance lo define el diseñador y es una medida de la velocidad de referencia que poseen los agentes. El coeficiente de tamaño aparente es parámetro que se introduce para modificar la densidad de recursos de los grafos y así realizar un control de la entrada de recursos en el escenario, se explicará adecuadamente en el apartado 5.5.

$$Avance = \frac{Paso\_avance}{k_{aparente}} \tag{5.3}$$

Para tomar la decisión de cuál será el próximo destino una vez que un agente alcanza un vértice, el sistema de control se va a basar en una serie de valores que obtendrá de sus sensores. La sensorización de los agentes debe permitir una riqueza de comportamientos que comprenda estrategias como las de las colonias de hormigas reales en las que se basan los algoritmos tipo ACO, para poder utilizarlas como referencia y/o comparación, y también debe permitir generar otras estrategias alternativas igualmente eficientes o necesarias en caso de que se realicen cambios en la configuración del entorno que supongan el uso de diferentes comportamientos.

Los agentes poseen un sistema de control sencillo que incorpora dos funciones: se encarga de generar una función de evaluación para las aristas basada en los valores que se obtienen de los sensores, y devuelve el valor de la tasa de feromonas que se depositan al cruzar una arista. Los parámetros que configuran el sistema de control son un conjunto de pesos que se asignan a cada uno de los valores de los sensores y que indican el valor y el sentido de la contribución de los valores de los sensores a la evaluación de cada arista. Se ha elegido esta codificación para el comportamiento por dos motivos: en primer lugar está basado en los algoritmos de optimización por colonias de hormigas en los que se inspira esta aplicación, y en segundo lugar, permite una extracción sencilla del comportamiento de la población con un simple análisis de los valores de estos parámetros, lo cual será importante a la hora de estudiar los resultados.

Para la ejecución del sistema del control se ha implementado la siguiente acción:

Decidir arista: esta acción se activa cuando un agente alcanza un nodo, este ha de decidir la próxima arista. Para ello el sistema de control evalúa cada una de las posibles opciones y selecciona la más valorada. El sistema de control es sencillo y por eso está incorporado en la acción de decidir y no en una EvTool (ver el apéndice anexo I) separada. La evaluación se realiza mediante la siguiente expresión:

$$V_i = C_i^f F_i + C_i^d D_i + C_i^l L_i + C_i^m M_i \tag{5.4}$$

Siendo los valores correspondientes a los sensores de nivel de feromonas (F_i), sensor del número de agentes en cada arista(símbolo), sensor de la longitud de cada arista (L_i), y por último, el sensor de memoria (M_i) que indica el tiempo que ha pasado desde que ese agente cruzó esa arista. Todos los valores obtenidos por los sensores están normalizados en el intervalo $[0, 1]$.

C_i^f, C_i^d, C_i^l y C_i^m son los coeficientes propios de cada agente que ponderan la influencia de cada uno de los términos de los sensores en la valoración global y toman valores en el intervalo $[-5, 5]$. La arista que obtenga una valoración más alta será la que seleccione el agente para continuar su ruta. Estos cuatro coeficientes son heredables, y por tanto, intervienen en el proceso de cruce. La arista que obtenga una valoración más alta será la que seleccione el agente para continuar su ruta. Cada combinación de estos coeficientes determina un comportamiento nuevo, y por tanto, un agente distinto. Los parámetros evolucionables, es decir, el genotipo de los individuos, está formado por estos 4 coeficientes junto a 1 coeficiente de producción de feromonas.

En resumen, la tarea básica que deben realizar los individuos será la de *explorar los grafos para alcanzar los puntos de destino y dejar un rastro de feromonas cuando hayan encontrado una ruta que lleva dicho destino*. La forma concreta de realizar esta tarea estará definida por los 5 parámetros evolucionables de los agentes. Una vez que la evolución llevada a cabo mediante el CeAS ha alcanzado una situación estable, a partir del número de agentes y de las estructuras de control obtenidas para cada uno de ellos, será posible extraer el algoritmo resultante de la misma forma que el caso de los algoritmos basados en colonias de hormigas, en los cuales, tras estudiar el comportamiento de estos insectos sociales, se extraen una serie de reglas y parámetros para configurar el algoritmo bioinsipirado.

Este algoritmo obtenido por el CeAS va a representar una solución *distribuida*, pues está repartido entre un grupo de agentes, *descentralizada*, pues no existe ningún elemento de central para el control, comunicación o acceso a información, y *estable* frente a ligeras variaciones en las condiciones del entorno o en el número de agentes, pues durante el proceso

de evolución estas condiciones han ido variando y la solución representa un punto de estabilidad frente a estos cambios.

Modelado distribuido y descentralizado

En cuanto a la definición del entorno, teniendo en cuenta que el objetivo es generar un grupo de agentes que recorran diferentes grafos y que busquen la ruta óptima para cualquiera de ellos, éste estará formado por un conjunto de grafos generados aleatoriamente y que irán cambiando cada cierto tiempo. Con esto aportamos generalidad a la solución encontrada, es decir, evitamos que surjan un grupo de agentes que sean capaces de resolver un grupo concreto de grafos debido a alguna particularidad o alguna característica común a todos ellos, pero que estos mismos en otro nuevo grafo no sean eficientes.

Los grafos están compuestos por dos componentes básicos claramente diferenciados: los nodos y las aristas que unen ciertos nodos. De entre estos últimos, existen dos especiales que son el nodo origen y el nodo destino, que deben estar claramente identificados. En los nodos se guarda una lista de las aristas con las que se conecta.

Con respecto a las aristas, cada una de ellas conecta dos nodos y posee tres valores asociados:

1. Un valor fijo de *longitud* o coste asociado (tiempo, energía, etcétera) a recorrer esa arista.
2. Un parámetro de estado que representa el *número de agentes* que están situados en un momento dado sobre una arista.
3. Un parámetro de estado que puede ser modificado por los agentes que atraviesan una arista y que permite cierto nivel de comunicación asíncrona a través del entorno. Este parámetro se inspira en el modo de comunicación de los insectos sociales, denominado *stigmergy*, mediante el uso de rastros de feromonas. En nuestro planteamiento lo denominaremos *nivel de feromonas*.

Cada uno de los grafos tiene asociado un valor de energía y un tiempo de funcionamiento. Cuando el nivel de energía llega a cero, éste es substituido por un nuevo grafo generado aleatoriamente. Esto ocurre también

cuando el tiempo de funcionamiento alcanza un valor límite, aunque el grafo no haya agotado toda su energía.

Dependiente del nivel de energía, cada uno de los grafos posee un valor de tamaño aparente. Este valor es un coeficiente que modifica los valores de longitud de cada arista haciendo que el tiempo empleado en recorrerlos aumente o disminuya. Comentaremos más en profundidad este aspecto en el apartado de gestión energética.

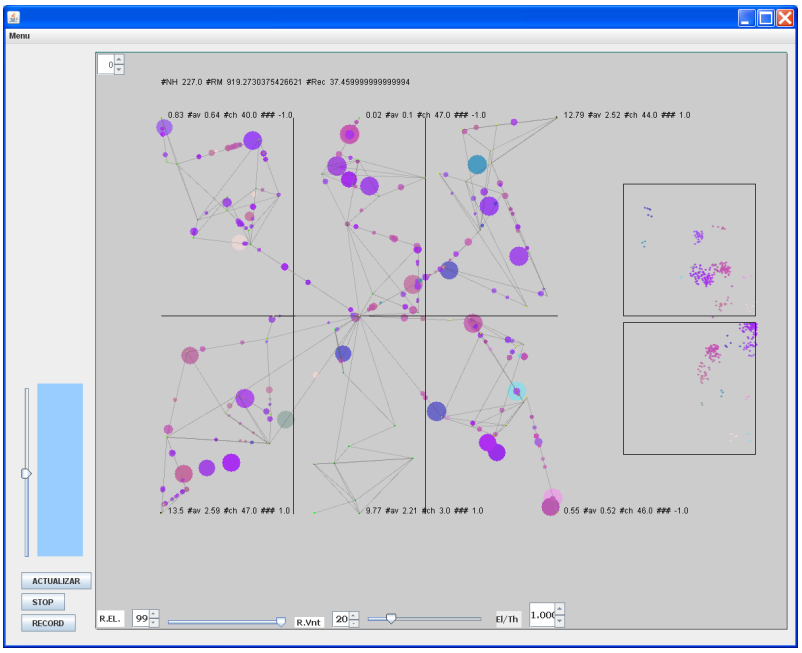


Figura 5.1: Escenario para la búsqueda del camino más corto en un grafo. Los agentes están representandos por círculos cuyo color y tamaño indica su código genético y edad respectivamente. Los 6 grafos se dibujan como un conjunto de líneas que en la posicióllas aristas y por un círculo negro que representa el nodo destino y cuyo tamaño es proporcional al nivel de energía de cada grafo.

En la simulación en el WaspBed se generan seis grafos interconectados compuestos por 12 nodos y 25 aristas. Estos grafos se generan de forma aleatoria y todos comparten el mismo nodo origen. El entorno en cada instante, por tanto, está compuesto por seis grafos. En la figura 5.1 se

muestra una captura del entorno. Cada uno de los agentes se representa mediante un círculo cuyo color representa sus parámetros evolucionables y su área el tiempo de vida, así los individuos más exitosos, además de ser muy numerosos, son aquellos que alcanzan la edad máxima, lo que hace que la superficie ocupada por un color sea una representación de lo exitoso de ese código genético.

Los grafos no poseen sistema de control, y por tanto, solo poseen reacciones. Éstas corresponden a la actualización continua de los grafos a los que se han de enfrentar los agentes de exploración, a la modificación de la densidad de recursos para hacer consistente la gestión energética que nos permite controlar la población y a la disminución del nivel de feromonas en cada arista para evitar una convergencia prematura hacia las primeras rutas encontradas, es decir, para reducir la explotación del algoritmo de búsqueda de rutas y por consiguiente aumentar la exploración. Por consiguiente, los eventos asociados a los grafos son:

Regeneración: evento que se ejecuta asociado a un grafo cuando éste agota sus recursos o su tiempo de simulación. En ese momento es substituido por un nuevo grafo generado de forma aleatoria pero con el mismo número de aristas y de nodos que el que desaparece. Más concretamente, cuando un grafo agota sus recursos, pasa a un estado desactivado en el cual todos los agentes se dirigen hacia el nodo origen para desalojar ese grafo y situarse en otro. Durante el tiempo que dura este desalojo del grafo no se permite la entrada a agentes de exploración, pero el nivel de energía sigue aumentando. Esto es necesario para hacer consistentes los balances de energía, y una vez que ya no hay agentes en el grafo, se genera el grafo nuevo manteniendo el valor de energía del grafo anterior, con lo que no hay ninguna variación de energía a mayores de la tasa de entrada establecida.

Actualización del tamaño aparente: este evento tiene como objeto llevar a cabo la regulación energética del sistema mediante el ajuste de la densidad aparente de recursos en el entorno de exploración, es decir, el tamaño aparente del grafo se modifica con respecto a la cantidad de energía disponible. Basándonos en una superficie inicial del entorno, este ajuste se realiza de manera que, cuando baja la cantidad de energía, baja también la densidad de energía, y por tanto, aumenta el recorrido

medio entre diferentes fuentes de energía. Este ajuste y otros relativos a los balances de energía se tratarán en detalle en el apartado 5.5, pero en definitiva, el tamaño aparente se modificará de manera inversamente proporcional a la cantidad de energía en el grafo.

Por último, existe una reacción encargada de reducir el nivel de feromonas en las aristas:

Evaporación de feromonas: tanto en los sistemas reales, como en los entornos virtuales bioinsipirados en comportamientos de hormiga, un proceso fundamental es la evaporación de feromonas en las rutas realizadas para evitar saturar las primeras rutas encontradas por los agentes y aumentar la exploración en el algoritmo de búsqueda.

Siguiendo la metodología expuesta en el apartado 4, a continuación vamos a tratar los cuatro niveles de interacción entre elementos del escenario para asegurarnos de que se realizan cumpliendo las restricciones impuestas para una resolución distribuida y descentralizada y para explicar en detalle cómo se han implementado.

- **Información de entorno:** el flujo de información entre los agentes de exploración y el entorno se realiza mediante la reacción *percibir*:

  Percibir: se encarga de actualizar los valores de los cuatro sensores del agente, cuando este alcanza un nodo. La información que requieren los sensores se extrae de parámetros del grafo y de la memoria de itinerario.

  La información de entorno utilizada se obtiene mediante la extracción del nivel de feromonas y de la longitud para cada una de las aristas. Esta información es completamente local y distribuida en cada uno de los nodos. Un agente no tiene acceso a la información de todo el grafo, sino que sólo tiene acceso a la información disponible en el nodo en el que se encuentra.

- **Información de grupo:** la actualización del sensor de número de individuos en una arista es el único flujo de información directo entre agentes. Representa un modo de conocer cierta información del

resto de la población, pero en este caso, también de forma parcial y local. Debemos resaltar el hecho de que el tipo de comunicación 'stigmergy', aunque permite obtener información del grupo, se realiza a través del entorno y es por eso que la hemos incluido en el apartado anterior.

- **Actuación sobre el entorno:** Los agentes poseen asociados dos eventos que modifican parámetros del entorno, uno es la acción de liberación de feromonas que constituye una de las acciones de los individuos:

  Liberar feromonas: la acción de liberar feromonas se realiza solo en el camino de regreso desde el destino hasta el origen, y el sistema de control decide la cantidad de feromonas que produce cada agente en función de un parámetro evolucionable propio de cada elemento.

  El otro evento es la llegada al nodo de destino, en el cual los agentes reciben un valor de energía prefijado, y por consiguiente, el grafo ve reducida su energía.

- **Actuación en el grupo:** la única interacción que existe entre miembros de la población es la de reproducción. Esta se explicará en el apartado de selección reproductiva.

Estudio de la utilidad privada y utilidad global

En primer lugar debemos establecer con claridad cuál es la utilidad global de nuestro sistema. En un problema de búsqueda del camino más corto está claro que el objetivo es que la ruta propuesta por el solucionador del problema tenga un valor de distancia asociado lo menor posible. En un problema de generación de un comportamiento colectivo de búsqueda del camino más corto en un determinado tipo de grafos como el que nos ocupa, el objetivo es que la distancia de la ruta óptima proporcionada para cada grafo sea la menor posible en cada paso de tiempo, pero esto nos conduce a un problema multiobjetivo o al establecimiento de una función de utilidad para generar una solución de compromiso. De forma más clara, los objetivos límite que plantea el escenario son la obtención

de forma rápida de una solución de calidad media o bien poblaciones que obtengan soluciones de mayor calidad más tarde.

En el escenario propuesto, los agentes substraen una cantidad fija (e_i) al valor de energía de un grafo cuando alcanzan el destino: cuanto más rápido realizan la ruta más veces extraen energía al grafo, y por tanto, menor será su nivel de energía. En definitiva, la medida de utilidad global que se extrae de este planteamiento se obtendría a partir del sumatorio del nivel de energía de cada uno de los grafos.

$$G(z) = \sum_{grafos} e_i \tag{5.5}$$

El objetivo será reducir este nivel de energía, pues reducir este valor implica que los grafos se están recorriendo de la manera más eficiente. Si tenemos en cuenta el análisis de utilidad privada explicado en el capítulo del CeAS, para calcular la utilidad diferencial que allí se propone debemos definir un segundo término. Recordemos que representaba el valor de utilidad del entorno si eliminamos los efectos un agente concreto. Para nuestro caso, si un agente alcanza un punto de destino, mejora el valor de utilidad en el valor de recompensa energética predefinido (r_e) y si no lo alcanza esta valor no se modifica. Por lo tanto, la utilidad diferencial será:

$$G(z_{-i} + c_i) = \begin{cases} \sum_g e_i - r_e & \text{: entrega} \\ \\ \sum_g e_i & \text{: no entrega} \end{cases} \tag{5.6}$$

$$D_i = \begin{cases} r_e & \text{: entrega} \\ 0 & \text{: no entrega} \end{cases} \tag{5.7}$$

Con respecto al debate planteado anteriormente, esta solución implica que se está generando una función de utilidad que finalmente tenderá a generar una población con una solución de compromiso entre velocidad y optimalidad. El resultado de esta solución de compromiso estará determinado por ciertos parámetros con los que hemos definido el entorno y que van determinar cómo se asignan las recompensas. Los parámetros que son de relevancia son, principalmente, el *tiempo medio de vida de un grafo* y la *cantidad de energía que puede almacenar un agente*. Si el grafo

tiene un tiempo de duración grande, esto permitirá a las estrategias que generan valores próximos al óptimo imponerse durante un gran intervalo de tiempo, orientando la evolución hacia ese tipo de comportamientos. Si el valor de energía de los agentes posee un nivel de variabilidad muy alto, las situaciones transitorias en las que se explora un grafo pueden extinguir los agentes con comportamientos lentos en obtener soluciones, pues en esta etapa reciben pocas recompensas.

Gestión energética

En este escenario, la energía se utilizará para controlar las dinámicas de población, el tamaño de la población resultante y la supervivencia de cada individuo, y por tanto, la probabilidad de entrar en la selección reproductiva.

Los agentes poseen un nivel de energía y otro de tiempo de simulación. Cuando se quedan sin energía o alcanzan su tiempo máximo de simulación son eliminados. Para llevar a cabo el estudio y el control de la estabilidad del sistema, se introduce en el entorno un control de la supervivencia de los agentes mediante la existencia de un nivel de energía instantáneo en cada agente. Esta gestión de la energía, que más adelante describiremos en detalle, nos permitirá realizar un análisis de las dinámicas que se producen durante la simulación por medio de los balances de energía a nivel de entorno y de población, y sobre estos análisis generar la combinación de parámetros de consumo y la tasa de entrada de energía que nos interesan para llevar a la población al punto de equilibrio deseado. A continuación estudiamos los cuatro puntos que especifica la metodología del CeAS con respecto a la gestión energética:

- **Entrada de energía:** a medida que avanza la simulación, la energía entra en el sistema con una tasa especificada por el diseñador. Esta energía estará disponible para cualquier agente, pero la única manera que tienen de recibirla es mediante las recompensas asignadas a la realización la tarea con éxito.

- **Salida de energía:** el nivel de energía decrece a medida que pasa el tiempo y también cuando se realiza la reproducción, pues se les transmite parte de esta energía a los descendientes.

- **Intercambio energético:** el intercambio energético entre agentes se produce sólo durante la reproducción. Cuando se produce la generación de un nuevo individuo, este reemplaza a uno de los existentes y se le suministra un valor de energía que es la mitad del promedio del valor de energía de los dos individuos base.

- **Asociación energía-utilidad:** tal y como comentamos, un agente recibe una recompensa por haber finalizado la tarea de búsqueda, por tanto, cuanto mayor sea su velocidad para resolver la tarea, mayor tasa de energía obtendrá.

En este escenario utilizaremos los flujos de energía en el entorno para controlar la supervivencia y la reproducción de los individuos, medir el nivel de utilidad de cada uno de ellos, y generar un entorno que aumente de forma gradual su nivel de dificultad para permitir una evolución más dirigida y menos aleatoria.

Debido a la entrada y salida de energía en cada grafo, pueden ocurrir dos cosas: que la energía que le substraen los agentes sea superior a la que se introduce de forma constante en el grafo durante el tiempo suficiente como para que se agote la energía de un grafo y sea substituido, o que el balance sea positivo, y por tanto, la energía crezca gradualmente en el grafo.

Con este procedimiento de substitución conseguimos evitar que las soluciones sean particulares para uno o varios grafos, y además se evita que la población converja precipitadamente cuando la tarea es sencilla -y por tanto la frecuencia de reproducción es más alta- pues cuanto más fácil es resolver un grafo, antes agota su energía y antes es substituido.

Finalmente, en el caso de que el nivel de energía sea bajo, aparece un incremento considerable del componente estocástico en la asignación de recompensa a individuos eficientes. Esto ocurre debido a que la cantidad de energía que se obtiene al realizar un trayecto es constante, y cuando el nivel de energía es bajo, muchos de los agentes que realizan el trayecto hacia el destino no obtienen recompensa, pues esta se agota antes de que lleguen. Aunque existe un sesgo que aumenta la probabilidad de que los individuos ineficientes sean los que se queden sin recompensa (este

es proporcional a la relación entre las longitudes de las distintas rutas escogidas por los agentes, las diferencias son bajas y a medida que avanza la evolución disminuyen), en muchos de los casos, esto no ocurre porque hayan escogido el camino menos eficiente, si no porque han entrado en el grafo en el momento inadecuado.

Con el objeto de paliar este efecto se ha establecido una relación entre la energía disponible en el entorno y el tamaño aparente del mismo. A medida que disminuye la energía aumenta el tamaño aparente. Esto genera dos efectos que son de interés. En primer lugar, a medida que el nivel de energía disminuye, el tiempo necesario para alcanzar el destino aumenta. Esto complica la tarea y hace que el balance de energía para cada agente sea menor al conseguir alcanzar el objetivo, es decir, aumentan las diferencias entre la longitud de las trayectorias utilizadas por cada agente y se acentúa más el sesgo hacia los más eficientes y por tanto se reduce la aleatoriedad.

Para cada agente, el instante de llegada al destino viene determinado por la siguiente ecuación.

$$T_{llegada\_destino} = T_{entrada\_al\_grafo} + k_{aparente}\frac{Longitud\_trayectoria}{velocidad} \tag{5.8}$$

El instante de entrada al grafo $T_{entrada\_al\_grafo}$ es una variable aleatoria sobre la que no tenemos control y la longitud de la trayectoria escogida es inversamente proporcional a la eficiencia de cada agente. Al aumentar el valor del coeficiente de tamaño aparente $k_{aparente}$ damos más valor a la eficiencia que a la aleatoriedad del instante de entrada, y por tanto, la energía puede crecer más evitando la desaparición del grafo. En último caso, los agentes que llegarán a un nodo destino sin energía serán, con mayor probabilidad, aquellos con eficiencia menor.

Cuando un grafo posee una configuración de nodos y aristas complicada o bien la población no es demasiado eficiente, la tendencia es a que la energía en ese grafo crezca indefinidamente. El segundo efecto derivado del aumento del tamaño aparente es que conseguimos que en el caso de que la energía aumente, la distancia promedio necesaria para alcan-

zar el destino disminuya, y por tanto, aumente la frecuencia con la que los agentes lo alcanzan evitando que diverja el valor de energía del grafo.

Selección reproductiva

Para asociar el objetivo del escenario a la evolución de la población de agentes, *la llegada al punto de destino estará asociada al evento de reproducción*, y a su vez estará asociada también, para premiar la velocidad, a un incremento en el tiempo de vida de los agentes. Por lo tanto, cuanto más rápido se realice la ruta, más veces se podrá llevar a cabo la reproducción, y en consecuencia, las características de ese individuo se transmitirán un mayor número de nuevos individuos. Esto provoca que las características que mejoran la eficiencia tendrán una mayor probabilidad para perpetuarse entre la población.

A continuación definimos cada uno de los siguientes apartados dentro de la selección reproductiva:

- **Proceso de selección:** cuando un agente alcanza el destino, recorre las aristas que ha seguido en su trayectoria en sentido inverso hasta alcanzar de nuevo el nodo origen. Al llegar a la arista que conduce al nodo origen, el agente puede reproducirse con otro agente que se encuentre en la misma situación. El agente comprueba todos los agentes que se encuentran en esa última arista y selecciona aleatoriamente uno de ellos para reproducirse. El tipo de cruce que se utiliza es el cruce bipolar descrito en el apartado 4.1 y los parámetros que entran en el cruce son los parámetros del sistema de control, los coeficientes de ponderación de la evaluación de las aristas y el coeficiente de liberación de feromonas. El cruce genera un nuevo agente que se sitúa en el punto origen.

- **Asociación selección/nivel de utilidad:** no existe una asociación directa entre la selección y el nivel de utilidad, pero sí indirecta, ya que la selección se produce al llevar a cabo la tarea objetivo. El nivel de utilidad permite a los agentes vivir más tiempo, y por tanto, tener más probabilidades de entrar en la selección.

- **Parametrización de la selección reproductiva:** el único parámetro que interviene en el proceso de selección y que permitiría regularlo es la longitud de la arista que llega al origen, pero esta longitud

se genera aleatoriamente para cada grafo, con lo cual no podemos modificar este proceso de selección de ningún modo.

Evaluación

No existe evaluación dentro de la selección, los individuos base se eligen aleatoriamente.

Combinación genética

Tal y como hemos explicado en el capítulo 4, se utilizará el cruce bipolar con los parámetros de ajuste por defecto. El nivel de exploración y de explotación vendrán definidos por estos parámetros que representan la mayor o menor amplitud de las funciones de probabilidad que generan los nuevos genes a partir de los genes de los individuos base. Los parámetros que son evolucionables en cada agente de este modelo serán los coeficientes del sistema de control y la tasa de producción de feromonas. Serán, por tanto, los que se transmitan al nuevo individuo generado.

Reemplazo

En este escenario, el reemplazo viene determinado por el tiempo de vida máximo de un agente. La distribución de energía es variable como ya hemos visto, y por tanto, en determinadas situaciones de escasez de recursos, el nivel de reemplazo será más alto, y en otras situaciones en las que la densidad de recursos por agente sea más alta, el reemplazo será menor.

A continuación vamos a explicar cómo se ha realizado la implementación del escenario propuesto en el simulador WaspBed.

5.3. Planteamiento en el WaspBed

En este apartado expondremos, siguiendo la estructura definida por el WaspBed, el modelo que se ha creado para simular este experimento (ver figura 5.2). En primer lugar se definen los tipos de elementos a partir de los archivos XML. Las tablas mostradas en las figuras 5.3 y 5.4 muestran

la definición de los elementos *Agente de Exploración* y *Grafo*.

A continuación se definen los archivos XML de creación. En el caso de los agentes solo indican la población inicial y en el caso de los grafos, generan los 6 grafos teniendo en cuenta que todos comparten el nodo origen y que cada uno posee una zona asignada en la que puede situar sus nodos, como se puede ver en la figura 5.5 correspondiente a un paso de la simulación.

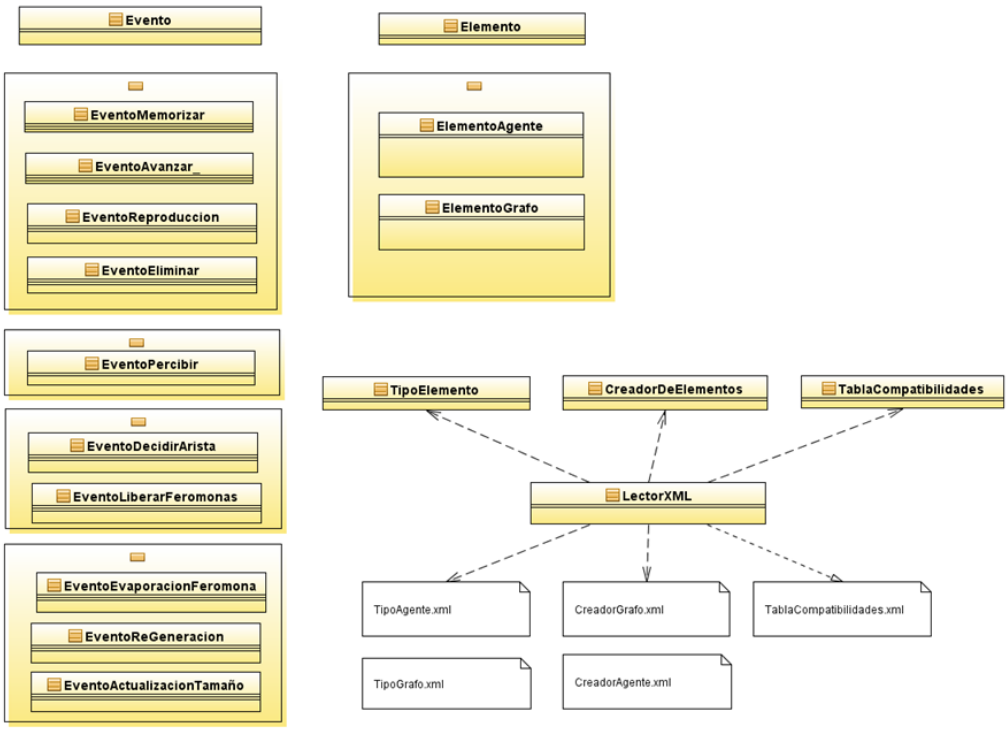


Figura 5.2: Diagrama de clases del escenario.

Posteriormente definimos los los eventos que crearán las interacciones entre estos elementos. Para cada uno de los elementos, crearemos las acciones, reacciones y percepciones y las asociaciones entre estos vendrán detalladas en la tabla de compatibilidad (figura 5.6).

La tabla de la figura 5.7 muestra una descripción breve de cada uno de los eventos generados.

Elemento Agente de Exploración	
Parámetros de tipo de elemento	**Valor**
Coeficientes SC	4x[-5,5]
Tasa producción feromonas	[0,5]
Ángulo de visión celdas	135º
Alcance de visión reproducción	150
Edad Madurez	100
Edad Máxima	1000
Alcance de visión vigilantes	150
Parámetros descriptivos de elemento	
Coeficientes del sistema de control	Evol [-1,1]
Longitud memoria itinerario	Evol [1,100]
Parámetros de estado	
Posición	[0,1000][0,700]
Longitud de la ruta recorrida	[0,∞]
Nivel energía	[1,1000]
Edad	[0,5000]
Sensor de feromonas	[0,1]
Sensor de agentes	[0,,1]
Sensor de distancia	[0,1]
Sensor de memoria	[0,1]
Memoria itinerario	M x [Ind Arista]

Figura 5.3: Definición del tipo de elemento Agente de Exploración.

Elemento Grafo	
Parámetros descriptivos de tipo	**Valor**
Número de aristas	[25]
Número de nodos	[12]
Tasa entrada energía	[5]
Consumo unitario	[1]
Edad máxima	[5000]
Parámetros descriptivos de elemento	
Coordenadas de los nodos	12x[0,1000][0,700]
Matriz de conectividad	12x12x{0,1}
Tasa entrada energía	[0,100]
Consumo unitario	[0,10]
Parámetros de estado	
Longitud de la ruta óptima	[0,∞]
Nivel feromonas por arista	25x[1,100]
Edad	[0,5000]
Coeficiente de tamaño aparente	[0,50]

Figura 5.4: Definición del tipo de elemento Grafo.

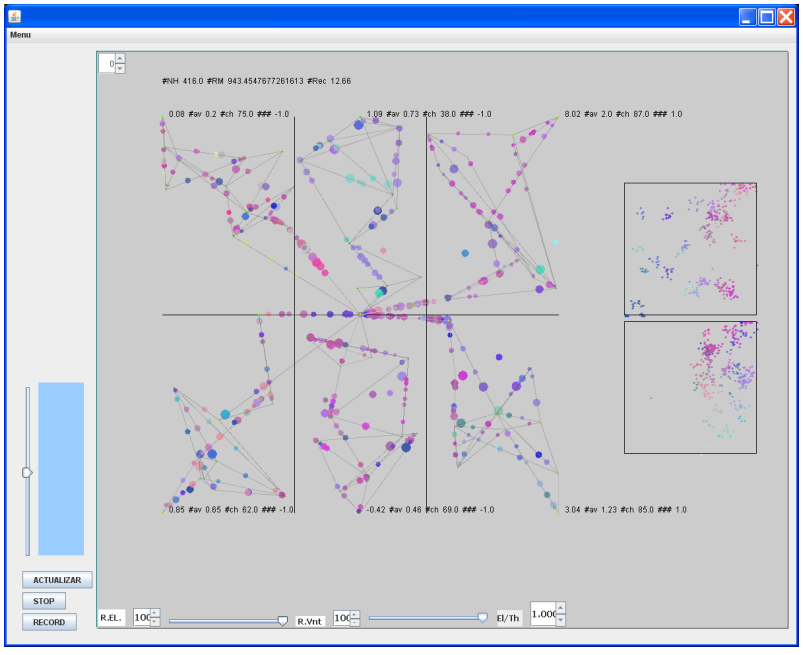


Figura 5.5: Simulación del escenario en WaspBed.

Figura 5.6: Archivo XML de la tabla de compatibilidades.

Evento	Descripción
Reproducción	Genera un nuevo individuo usando el cruce bipolar y los parámetros evolucionables especificados
Avanzar	Avanza en dirección al destino
MemorizarRuta	Actualizar la memoria del itinerario.
Regeneración	Desactiva el grafo y una vez que no hay ningún agente dentro genera una nuevo conjunto de vértices y una nueva matriz de conectividad
Percibir	Actualiza los valores de los sensores con la información del nodo actual y de la memoria de itinerario
Eliminar	Elimina un individuo y su representación gráfica
DecidirArista	Ejecuta el sistema de control sobre los valores de los sensores y asigna un nuevo destino
LiberarFeromonas	Si el sistema de control decide liberar feromonas suma el valor de su tasa de feromonas al nivel de feromonas de la arista que acaba de abandonar cuando llega a un nodo
ActualizarTamaño	En función del nivel de energía de un grafo, cambia el tamaño aparente del grafo.
EvaporarFeromonas	En función del valor de tiempo sin cobertura, se genera un valor de riesgo que servirá para medir el riesgo total en cada instante en todo el escenario.

Figura 5.7: Lista de eventos del escenario de búsqueda del camino más corto en un grafo.

5.4. Resultados

Una vez que se ha generado el escenario de simulación y se han definido los procedimientos de evolución del CeAS y del escenario, se deja evolucionar el sistema hasta que la población haya convergido a una configuración estable que solucione el problema de forma satisfactoria. En este momento se extraen los parámetros de toda la población y se estudia cuáles son las pautas que guían el comportamiento generado.

En un primer experimento vamos a utilizar la configuración base expuesta en el apartado anterior. A continuación compararemos los resultados con los que se esperan de un algoritmo de colonia de hormigas.

Posteriormente realizaremos otras pruebas variando algunos de los parámetros de configuración del escenario con el fin de comprobar la capacidad del modelo diseñado, del software de simulación y del CeAS para generar nuevas soluciones a otros problemas.

5.4.1. Configuración básica

En primer lugar, la evolución utiliza las reglas que hemos descrito para el escenario y que se explicaron en el apartado de configuración experimental. Los parámetros de inicialización se muestran en las tablas de los creadores de elementos (ver figura 5.8). Se permite al escenario evolucionar hasta que se alcanza una población estable que resuelve el problema, es decir, hasta que no se producen variaciones significativas en la configuración de la población. Esto ocurre en dos o tres minutos de ejecución en tiempo real que corresponden en este caso a 11.000 pasos de evolución. La representación en el WaspBed se ha hecho de tal manera que durante la ejecución de la evolución podemos ver la configuración genética de la población, como se muestra en la figura 5.9.

En dicha figura podemos ver, a la izquierda, la distribución genética de la población para una de las evoluciones tras los primeros 300 pasos. Podemos ver que la distribución de los parámetros ya no es la distribución aleatoria homogénea inicial, si no que aquellos individuos con un valor

del parámetros de memoria negativo han desaparecido con lo que suponemos que esos agentes resultaban altamente ineficientes, más adelante estudiaremos el motivo. Se muestra también a la derecha de la figura la configuración de la población a los 11.000 pasos, en los que vemos que la población ha convergido hacia una composición relativamente homogénea.

Para el experimento llevado a cabo con la configuración base, los parámetros genéticos de la población una vez alcanzada la estabilidad se muestran en la figura 5.10.

Creador Agente de Exploración **NEI: 300**	
Parámetros de estado	
Posición	[0,1000][0,700]
Longitud de la ruta recorrida	[0,∞]
Nivel energía	(1000)
Edad	(0)

Creador Grafo **NEI: 1**	
Parámetros de estado	
Longitud de la ruta óptima	(0)
Nivel feromonas por arista	25x(0)
Edad	(0)
Coeficiente de tamaño aparente	(1)

Figura 5.8: Creadores de la configuración base para los tipos de elemento: Agente de Exploración y Grafo.

Tal y como se muestra en la figura 5.10, los parámetros más relevantes son el coeficiente de memoria y el coeficiente de concentración de feromonas. Este resultado representa una población que selecciona, en primer lugar, de entre las aristas disponibles, aquellas que están más atrás en la memoria de itinerario. Es decir, aquellas por las que hace más tiempo que ha pasado o aquellas por las que no ha pasado antes.

Este resultado es muy similar a la estrategia utilizada en el algoritmo bioinspirado ACO que utiliza una lista tabú de un tamaño prefijado.

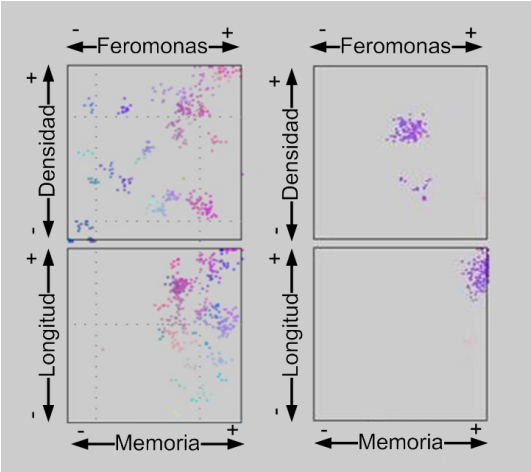


Figura 5.9: Representación genética de la población. Izquierda: tras 300 pasos. Derecha: tras 10.000 pasos.

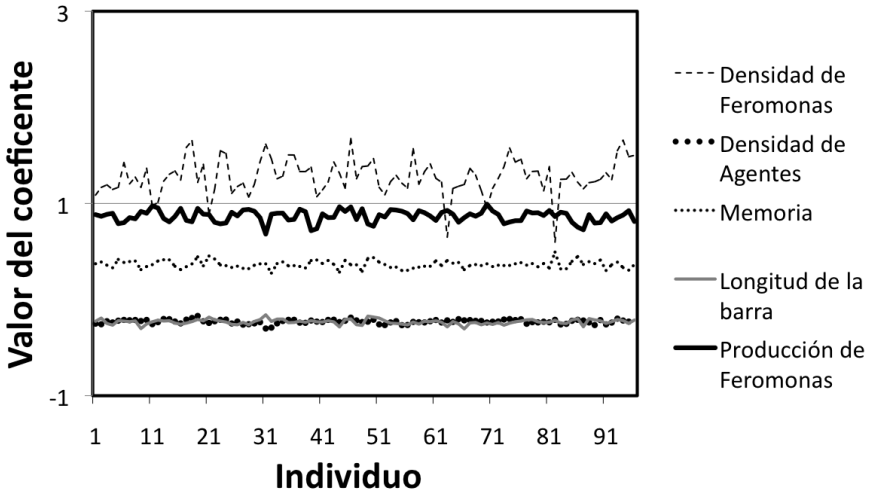


Figura 5.10: Parámetros de la poblacion resultante utilizando las reglas originales.

En segundo lugar, el agente selecciona aquellos nodos con mayor valor de densidad de feromonas, pues tal y como ocurre en el comporta-

miento de los insectos reales, y tal y como contempla el ACO también, las rutas con mayor rastro de feromonas han sido visitadas con más frecuencia, y eso da una idea de que estas rutas son más eficientes para alcanzar el nodo destino.

En tercer lugar, la densidad de agentes obtiene un valor pequeño y negativo, lo que resulta también de interés para nuestro caso ya que explica el comportamiento que poseen los agentes cuando inician su búsqueda frente a un grupo de grafos o aristas que todavía no han sido exploradas(tal y como ocurre en la inicialización): se reparten equitativamente entre los distintos grafos o aristas nuevas. Así, el primer agente elige un camino al azar, el siguiente hace lo mismo pero descarta el camino del primer agente, el tercero descarta las opciones de los dos anteriores y así sucesivamente obteniendo un comportamiento de exploración en el caso de que exista información temporal o rastros de rutas anteriores.

El coeficiente de distancia tiene un valor próximo a cero lo que indica que no es relevante a la hora de decidir la ruta. Debemos indicar que, en realidad, este parámetro no ofrece información de interés a la toma de decisiones, pues la ruta óptima en cada uno de los grafos generados aleatoriamente no posee ningún sesgo hacia aristas cortas o largas. Una arista larga puede guiar directamente hacia las proximidades del destino, pero también puede llevar a una zona muy alejada. En cualquier caso, un punto próximo al destino puede no tener una combinación de aristas para alcanzarlo, y viceversa para un punto alejado.

Por último, en cuanto a la tasa de producción de feromonas, esta presenta un valor positivo y próximo a 1, lo que nos indica que los agentes están haciendo uso de su capacidad para dejar rastros en su camino de vuelta al origen.

La realización de este primer experimento ha servido para comprobar el funcionamiento general del CeAS para evolucionar una población con un objetivo prefijado en un entorno concreto. También ha permitido comprobar el funcionamiento del WaspBed en cuanto al uso de las herramientas que llevan a cabo la evolución de la población, la gestión del sistema de control y de los eventos asociados a estos sistemas de control, y finalmente, nos ha permitido poner a prueba la metodología de transfor-

mación que propone el CeAS. Los resultados han sido completamente satisfactorios.

5.4.2. Variación del escenario

A continuación se va a estudiar la capacidad del CeAS para generar nuevas soluciones ante variaciones en la configuración base de diseño del escenario en cuanto a las características de los agentes y del entorno. Esta es una de las principales propiedades de la técnica desarrollada a nivel práctico, ya que permite de forma sencilla estudiar el comportamiento del sistema ante cambios en parámetros básicos. Esta es una característica muy relevante en problemas reales de ingeniería ya que permite analizar qué estado estable alcanza un sistema complejo ante modificaciones en sus parámetros de diseño, lo cual repercute en un refinamiento de dicho diseño al comprobar el 'peso' real de estos parámetros.

5.4.2.1. Cambios en las capacidades de los agentes

Inicialmente, se ha planteado una nueva situación en la cual los agentes no tengan la capacidad de liberar rastros de feromonas. La figura 5.11 muestra los creadores para este experimento de los elementos Agente de Exploración y Grafo.

Creador Agente de Exploración NEI: 300	
Parámetros de estado	
Posición	[0,1000][0,700]
Longitud de la ruta recorrida	[0,∞]
Nivel energía	(1000)
Tasa producción feromonas	(0)
Edad	(0)

Creador Grafo NEI: 1	
Parámetros de estado	
Longitud de la ruta óptima	(0)
Nivel feromonas por arista	25x(0)
Edad	(0)
Coeficiente de tamaño aparente	(1)

Figura 5.11: Tabla del creador del Agente de Exploración y Grafo.

Los parámetros de evaluación una vez que la población converge a un estado estable, genéticamente hablando, se muestran en la figura 5.12.

Al no permitir rastros de feromonas, hemos forzado a que no pueda existir el modo de comunicación que se muestra en los sistemas reales, la comunicación a través del entorno o *stigmergy*. Tal y como puede extraerse de la gráfica 5.12 de coeficientes de la población, el comportamiento obtenido en este caso se guía por la memoria del itinerario y por la densidad de agentes. El comportamiento que se observa en la simulación es similar al obtenido en el caso anterior, aunque su eficiencia global es más baja, principalmente en grafos complejos, debido a que los agentes han perdido la capacidad de exploración inicial que les permitía el valor negativo del coeficiente de densidad de agentes en la configuración anterior. Lo que ocurre en este caso, es que se ha sustituido una comunicación explícita a través del entorno por una comunicación implícita unidireccional y menos persistente que indica qué camino está siendo más utilizado en un instante determinado.

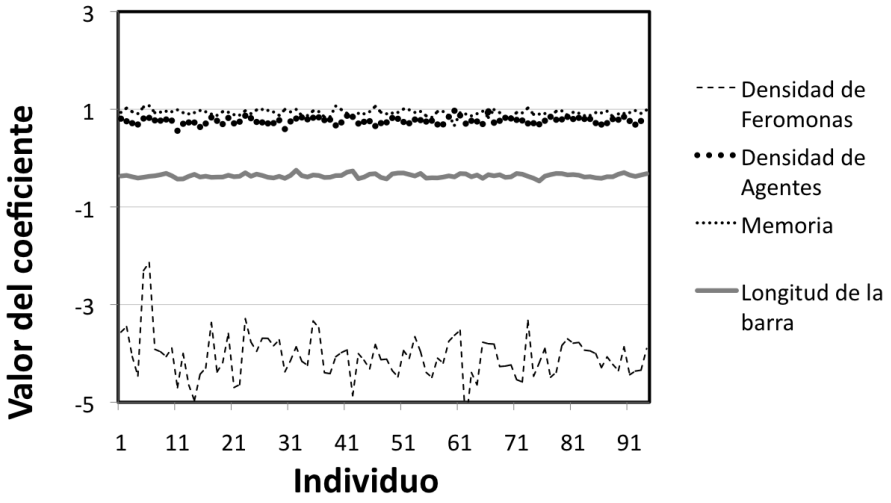


Figura 5.12: Parámetros resultantes para una población sin liberación de feromonas.

Los restantes coeficientes no son relevantes, el coeficiente de feromonas no aporta valor a la evaluación y es por eso que fluctúa dentro de la población, y el valor de longitud de los ejes vuelve a tener un valor pequeño y que no es significativo.

Como vemos, desde un punto de vista práctico, este nuevo algoritmo no mejora al anterior, pero lo que es relevante aquí es que con el CeAS hemos sido capaces de comprobar de forma muy sencilla la gran importancia que tiene la comunicación en este problema.

5.4.2.2. Cambios en la configuración del entorno

En este caso se ha complicado ligeramente el problema disminuyendo significativamente el tiempo de vida máximo. Es decir,ahora los grafos no tienen tiempo a quedarse sin energía, pues son desactivados antes. Se permite, de nuevo, el uso del rastro de feromonas que hemos visto que resulta muy relevante. El creador es el mismo que el utilizado en la configuración base, y simplemente se modifica un parámetro descriptivo de tipo de los elementos grafo (figura 5.13).

Elemento Grafo	
Parámetros descriptivos de tipo	
Edad máxima	[1000]

Figura 5.13: Variación en la definición del tipo de elemento grafo para el tercer experimento.

Tal y como ya hemos comentado, el algoritmo de búsqueda resultante dependerá de factores como el tiempo medio de duración de un grafo para generar poblaciones más rápidas y menos eficientes, o por el contrario más cercanas al óptimo pero más lentas. Es decir, poblaciones que busquen una solución de eficiencia media en un tiempo de exploración corto o bien aquellas que busquen una solución más eficiente y por tanto empleen más tiempo en ello o bien una solución de compromiso entre ambos extremos. Hemos buscado, en este experimento, variar la configuración del entorno de manera que la población resultante modifique su configuración final.

La primera consecuencia de esta modificación es que, en este caso, se reduce enormemente el tiempo de explotación de un grafo tras encontrar los caminos óptimos. Así, básicamente, los agentes están la mayoría del tiempo explorando nuevos grafos y buscando soluciones con la mayor velocidad posible. Los coeficientes obtenidos se muestran en la figura 5.14, y el comportamiento obtenido es similar al de la primera configuración, con una diferencia.

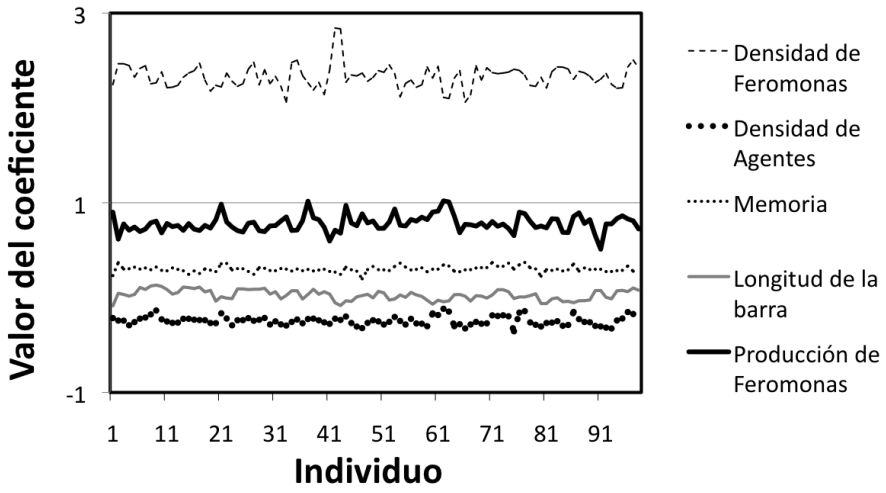


Figura 5.14: Población resultante tras reducir el tiempo máximo de vida de los grafos.

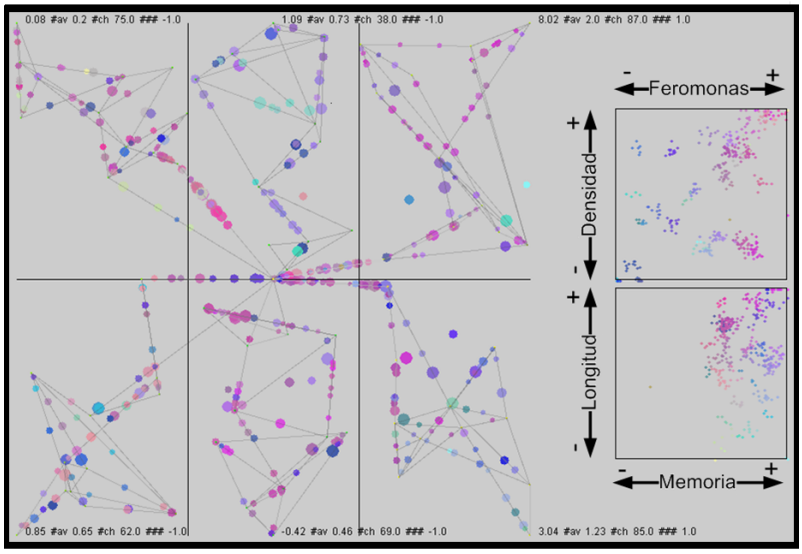
En este caso, el valor del coeficiente de feromonas es dos veces mayor para un mismo nivel de producción de feromonas y un valor de coeficiente de memoria similar. Esto implica que se otorga mayor importancia relativa al nivel de feromonas de las rutas posibles. Entendemos que esto es debido a dos motivos: en primer lugar, que los grafos se desactiven antes implica que no hay tiempo para dejar un rastro con una concentración de feromonas alta, y por tanto, debemos compensarlo dando el mismo valor a rastros más débiles. En segundo lugar, en cuanto alguno de los agentes resuelve el grafo y regresa liberando feromonas, los otros agentes dan prioridad a seguir esta ruta aún cuando pueda no ser la ruta óptima, pues no hay tiempo para hacer una búsqueda más exhaustiva. El resto de parámetros tienen un valor similar al de la primera configuración, y por

tanto, su explicación es la misma que dio en el apartado anterior.

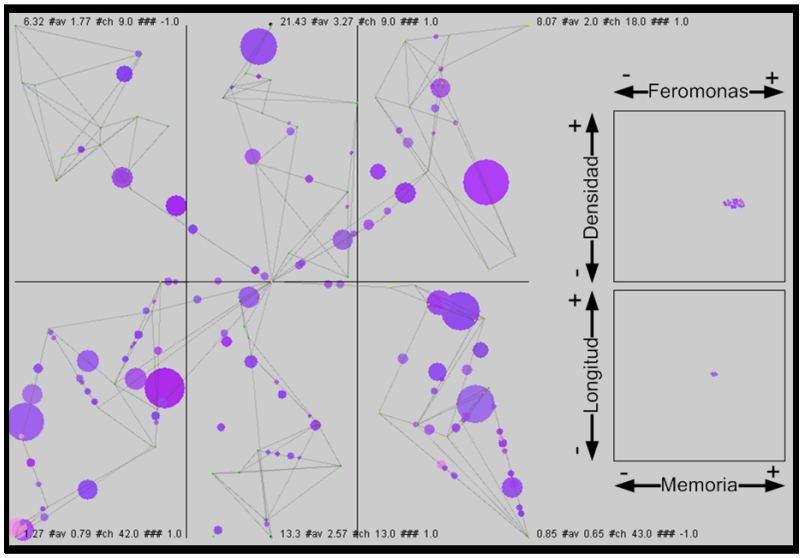
Se representa en la figura 5.15 el escenario en WaspBed utilizado en todos estos experimentos, y concretamente, esta figura muestra el resultado de evolución de este último experimento, desde los primeros pasos de la evolución (imagen superior) a los últimos pasos cuando la población ya ha convergido (imagen inferior). Esta figura sirve para ilustrar que un resultado únicamente visual no es suficiente en este tipo de problemas con gran número de agentes, pero aún así la representación basada en colores utilizada sirve de primera indicación sobre el comportamiento global del sistema.

En la parte derecha de la ventana de simulación, se muestra una representación de los cuatro coeficientes de la población completa. Tal y como dijimos en al inicio de este apartado se observa en todas las configuraciones que hemos probado que, tras pocos pasos de simulación, son eliminados aquellos agentes que tienen un coeficiente de memoria negativo, el motivo es que los agentes que presentan esa configuración poseen un comportamiento que tiende a elegir aristas sobre las que ya ha pasado, y por tanto, realizan trayectorias cíclicas lo cual representa un comportamiento extremadamente ineficiente.

Como conclusión de este primer escenario de aplicación diremos, primeramente, que hemos comprobado que el procedimiento definido por CeAS ha conseguido guiar la evolución de la población hacia un comportamiento que permite optimizar una función objetivo definida por el diseñador. Hemos visto también que el uso del CeAS y del WaspBed aporta gran flexibilidad a la hora de realizar cambios en el diseño de un escenario. Esto nos da muchas posibilidades de cara a estudiar cómo estas variaciones afectan al nuevo comportamiento generado, qué nuevas estrategias se siguen para optimizar la misma función y si las capacidades de los agentes son adecuadas para obtener una población eficiente.



a



b

Figura 5.15: Simulación en los primeros pasos de evolución del último escenario(a) y tras alcanzar la estabilidad (b).

Hemos comprobado también que, en muchas de las simulaciones, la población se extingue con cierta facilidad, o se alcanzan tamaños de po-

blación muy elevados lo cual aumenta considerablemente el tiempo de ejecución, llegando incluso a colapsar la aplicación. Esto es debido a las dinámicas de población que se crean en torno a los equilibrios energéticos que se establecen en el escenario. Si las condiciones iniciales están muy alejadas de los puntos de equilibrio, si las dinámicas que se generan son muy oscilantes, o simplemente, si el punto de equilibrio está fuera del rango definido para la aplicación, el sistema se convierte en inestable. Adicionalmente, hemos visto que el proceso de aprendizaje está muy influido por estas dinámicas pues repercuten en parámetros tan relevantes como la presión selectiva, -fuertes oscilaciones en la relación del número de agentes frente a la energía disponible provocan períodos con una presión selectiva muy fuerte y otros casi nula-, el tamaño de la población, el tiempo medio de evaluación, etcétera.

Con el objeto de dar un primer paso de cara a estudiar estas dinámicas de población en el CeAS y así permitir mejorar el proceso de configuración, diseño y control del escenario y del proceso evolutivo de la población, se ha realizado un análisis formal de los balances energéticos creados en este primer modelo. Así, se ha formulado un modelo matemático sintético a partir de estos balances de energías que nos permite generar las dinámicas de población previstas para unas condiciones concretas prefijadas.

5.5. Análisis energético

5.5.1. Análisis general de los balances de energía en un sistema abierto

El análisis de poblaciones que vamos a realizar basado en los niveles de energía está inspirado en los modelos de Lotka-Volterra o en los modelos que siguen las Ecuaciones de Crecimiento Logístico[1]. Lo primero que hemos de tener en cuenta para realizar un análisis de flujos de energía equivalente a estos modelos, es asegurarnos de que las tasas de crecimiento para las distintas especies del escenario están relacionadas con la densidad de individuos de otras especies. En muchos de los modelos

[1]Modelos que describen las dinámicas de un sistema biológico y basándose en las densidades de las diferentes especies o recursos que intervienen.

de simulación que vamos a usar necesitaremos adaptar las interacciones entre especies para que se cumpla esta condición:

$$\Delta\rho_{especies\_n} = \varphi(\rho_{especies\_n}, \rho_{especies\_i}, \rho_{especies\_j}, \ldots) \tag{5.9}$$

En efecto, en un modelo real, como en el caso de un modelo simple de depredador-presa, la probabilidad de que una presa encuentre una fuente de recursos, está relacionada directamente con la densidad de presas (recursos) y, consecuentemente, cuando los recursos decrecen, esta densidad también decrece, y por tanto, el tiempo medio necesario para encontrar recursos se ve incrementado.

A continuación se definen los balances que controlarán los flujos de energía del sistema y el número de individuos de cada especie. El primer balance de energía está relacionado con la energía disponible en el entorno en cada paso de tiempo mediante las siguientes ecuaciones:

$$\Delta E_{entorno} = R_{entrada} - R_{consumo} - R_{salida} \tag{5.10}$$

$$R_{consumo} = \sum_i R^i_{consumo} \tag{5.11}$$

El diseñador define una tasa de entrada de energía ($R_{entrada}$) en el entorno y una tasa de salida (R_{salida}) de recursos. Por otro lado, la tasa de consumo ($R_{consumo}$) de recursos debida a los individuos de todas las especies puede modelarse en función de tres parámetros, el número de individuos de una especie (N_{ind}), el tiempo medio ($T_{promedio}$) para alcanzar un recurso y el consumo unitario (C_{unit}) cuando se alcanza un recurso a través de la siguiente ecuación:

$$R_{consumo} = \frac{N_{ind} C_{unit}}{T_{promedio}} \tag{5.12}$$

A partir de la ecuación de energía del entorno, en un estado de equilibrio, la variación de energía debe tener valor cero. Esto nos proporciona la primera ecuación de equilibrio y relaciona todas las variables en juego. Por tanto:

$$\Delta E_{energia} = 0 \tag{5.13}$$

$$R_{entrada} = R_{consumo} + R_{salida} \tag{5.14}$$

El siguiente balance que vamos a hacer se establece para la energía de los individuos de todas las especies.

$$\Delta E^i_{poblacion} = R^i_{consumo} - N^i_{ind} C_{perdidas} - L^i_{tareas} \tag{5.15}$$

siendo N^i_{ind} el número de individuos de una especie i, $C_{perdidas}$ representa la perdida de energía inherente de cada agente por unidad de tiempo, $R^i_{consumo}$ representa el consumo de energía de una especie, pero en este caso representa una entrada de energía y finalmente L^i_{tareas} representa las pérdidas de energía extra que están asociadas al desarrollo de una tarea concreta como reproducirse, moverse, etc.

De nuevo, en el estado de equilibrio, la variación de energía debe ser cero, por tanto:

$$\Delta E^i_{poblacion} = 0 \tag{5.16}$$

$$R^i_{consumo} = N^i_{ind} C_{perdidas} + L^i_{tareas} \tag{5.17}$$

A partir de estas ecuaciones genéricas, que son las que usaremos para analizar cualquier entorno en el que utilicemos los flujos de energía para controlar y parametrizar la población, se puede realizar la particularización para cada modelo concreto con lo que obtenemos las relaciones generales entre los parámetros del estado estable.

En la próxima sección se presentan los detalles de la implementación de estos balances particularizados para el modelo de búsqueda de algoritmos de rutas que hemos expuesto en este capítulo.

5.5.2. Análisis energético para el escenario de búsqueda de ruta óptima

En primer lugar, y con el objeto de satisfacer la ecuación 5.9, que establece que el tiempo medio, y por tanto también la longitud del camino medio, para alcanzar un punto de recursos está directamente relacionada con la densidad de recursos, definimos en primer lugar una superficie fija equivalente de área (S) que representa el área del territorio en el que

el sistema evoluciona. Esto, junto con la cantidad de recursos disponible ($R_{cantidad}$) nos lleva a un valor de densidad de recursos aparente (D_{ap}).

$$D_{ap} = \frac{R_{cantidad}}{S} \tag{5.18}$$

A partir de esta expresión, definimos que la longitud promedio para alcanzar un recurso será proporcional a una distancia característica promedio entre puntos de recurso. Esta será proporcional a la raíz cuadrada del área de la superficie por unidad de recursos y finalmente, inversamente proporcional a la raíz cuadrada de la densidad de recursos:

$$L_{promedio} = \frac{c_0}{\sqrt{D_{ap}}} \tag{5.19}$$

y teniendo en cuenta que la superficie del escenario es constante:

$$L_{promedio} = \frac{C}{\sqrt{R_{cantidad}}} \tag{5.20}$$

Como consecuencia, cuando los recursos decrecen, la densidad de recursos decrece también y el camino promedio necesario para encontrar un recurso se ve incrementado y viceversa.

Lo que ocurre en muchos escenarios, y así es en el problema de rutas que planteamos, es que el número de proveedores de recursos es siempre el mismo y se encuentran siempre en la misma posición. Como consecuencia, la densidad de recursos es siempre la misma, y adicionalmente, en el caso de que alguno de los puntos fijos agote sus recursos, el éxito de los agentes, que no pueden predecir este tipo de situaciones, pasa a tener una componente aleatoria dominante como ya hemos comentado en el apartado 5.2 .

Para resolver este tipo de situaciones, se propone que, en lugar de modificar la longitud real de ciertas partes del escenario como función de la densidad de recursos, lo cual es harto más engorroso, se modificará la velocidad de avance para que el tiempo promedio para alcanzar un punto de recursos sea el equivalente.

Por tanto, suponiendo una longitud promedio inicial (L_0) y una velocidad de avance inicial preestablecidas (V_0), y teniendo en cuenta que el

efecto de variar la velocidad en lugar de la longitud característica debe ser equivalentes, es decir el tiempo promedio debe ser el mismo, por tanto la velocidad instantánea vendrá dada por la siguiente expresión:

$$T_{promedio} = \frac{L_{promedio}}{V_0} = \frac{L_0}{v_{avd}} \Rightarrow v_{avn} = \frac{L_0 v_0}{L_{promedio}} = C' \sqrt{R_{cantidad}} \quad (5.21)$$

Por tanto, de acuerdo al modelo general para los dos balances de recursos que controlan las dinámicas de población, y que están representados en las ecuaciones 5.10 y 5.15. Las simplificaciones respecto al modelo real que hemos impuesto en nuestro modelo energético para el escenario de optimización de rutas en grafos son las siguientes:

- Los recursos disponibles crecen con una tasa que es constante respecto al tiempo.
- Los individuos pierden energía con una tasa unitaria constantes.
- Cuando un individuo alcanza un punto de recursos, este incrementa su nivel de energía y disminuye el nivel de energía del entorno en una cantidad prefijada.
- No se considera la salida de recursos del sistema.

Por lo tanto, el nivel de energía disponible en el entorno para este escenario puede ser expresado de forma discreta como:

$$E_{i+1} = E_i + R_{entrada} - R_{consumo} \quad (5.22)$$

La entrada de recursos es conocida, vamos a estimar el valor del ratio de consumo de recursos a partir de la ecuación 5.12.

El tiempo promedio ($T_{promedio}$) para un instante concreto se obtiene a partir de la longitud promedio que emplean los agentes para recorrer el grafo ($L_{promedio}$) y la velocidad de avance (a_{avn}):

$$T_{avg} = \frac{L_{promedio}}{v_{avn}} \quad (5.23)$$

La longitud de avance la conocemos, pero para estimar la longitud promedio debemos descomponerla en dos términos, la longitud promedio

individual (L_{prom_indiv}), que es la longitud promedio para recorrer el grafo si hubiese una cantidad de recursos infinita y el factor de interferencia (F_i), que representa el efecto de que un agente alcance un punto de recursos y este ya esté agotado, y por tanto, provoca un aumento de la longitud promedio real para alcanzar un recurso. El factor de interferencia cambia con el número de individuos y con la densidad de recursos. Este factor será siempre igual o superior a uno. Sustituyendo en 5.12 el valor del tiempo promedio y el de la velocidad de avance:

$$R_{consumo} = \frac{N_{ind} C_{unit}}{L_{prom\_indiv} F_i} C' \sqrt{R_{cantidad}} \tag{5.24}$$

Para el equilibrio, tal y como se establece en la ecuación 5.14, la tasa de consumo y de entrada serán iguales, por tanto:

$$R_{entrada} = R_{consumo} = \frac{N_{ind} C_{unit}}{L_{prom\_indiv} F_i} C' \sqrt{R_{cantidad}} \tag{5.25}$$

Así, a partir del balance de energía del entorno obtenemos la siguiente expresión que nos ligará el número de individuos con la cantidad de recursos:

$$N_{ind} = \frac{C'' L_{prom\_indiv} F_i}{\sqrt{R_{cantidad}}} \tag{5.26}$$

con

$$C'' = \frac{R_{entrada}}{C_{unit} C'} \tag{5.27}$$

En la expresión anterior, la tasa de entrada es conocida y constante. Los términos que no conocemos ahora son el factor de interferencia, que debe ser obtenido de forma experimental y el término de la longitud promedio individual que no puede ser considerado un valor fijo, sino una distribución de valores con un valor determinado de media y desviación estándar.

Para obtener de forma experimental los parámetros que necesitamos para definir el comportamiento de la población, hemos lanzado la simulación para distintos valores fijos de número de individuos desactivando la reproducción y la eliminación de individuos, es decir, eliminando las asociaciones entre los agentes de exploración y los eventos eliminar y

reproducir de la tabla de compatibilidad. Se obtuvieron los siguientes resultados tras 10.000 pasos de evolución:

$R_{cantidad}$	8.27	8.09	9.12	15.19	20.92	32.91	69.77	141.4	272.7	495.9	911.5	4240	17122
N_{ind}	512	362	256	181	128	90	64	45	32	22	16	8	4

Figura 5.16: $R_{cantidad}$ y N_{ind} tras 10.000 pasos de evolución.

Hemos representado estos datos en la gráfica que muestra la figura 5.17, relacionando el número de individuos con los recursos disponibles. Hemos representado también la desviación estándar que se obtuvo para los datos que se muestran, de manera que el área entre las curvas de desviación representa dónde se sitúan el 75 % de los valores obtenidos.

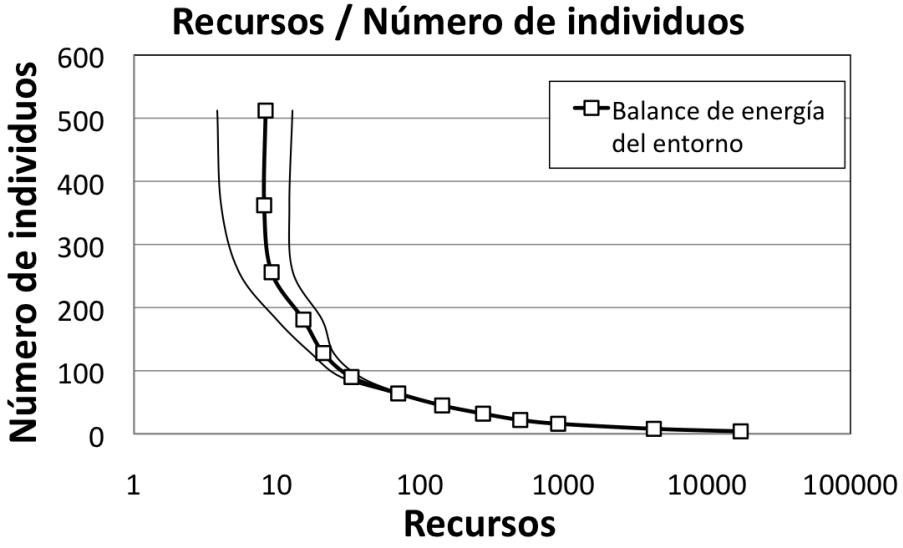


Figura 5.17: Balance de energía del entorno

De los datos obtenidos en la tabla anterior podemos extraer un valor experimental para el resultado de multiplicar la longitud promedio individual por el factor de interferencia, es decir, la longitud promedio efectiva $L_{efectiva}$:

$$L_{efectiva} = L_{prom\_indiv} F_i = \frac{N_{ind}\sqrt{R_{cantidad}}}{C''} \tag{5.28}$$

En este caso los valores obtenidos son:

N_{ind}	512	362	256	181	128	90	64	45	32	22	16	8	4
L_{exp}	8679	6067	4555	4157	3450	3042	3150	3152	3113	2886	2846	3069	3084

Figura 5.18: N_{ind} $L_{efectiva}$ tras 10.000 pasos de evolución.

A continuación estudiaremos el balance energético para la población de individuos, de acuerdo a la ecuación 5.15 del mismo modo que lo hemos hecho para el balance anterior y con las siguientes consideraciones particulares:

- En cada paso de tiempo, cada individuo tiene una perdida de energía de una unidad.
- No se contemplan pérdidas de energía asociadas a tareas concretas.
- La recompensa otorgada por completar un grafo es un valor fijo (e_{unit})

Considerando el término obtenido en la ecuación 5.24 para la tasa de consumo, substituyendo en 5.15 la energía de los agentes en cada instante (A_i) de tiempo es:

$$A_{i+1} = A_i - N_{ind} + \frac{N_{ind} e_{unit} C'' \sqrt{R_{cantidad}}}{L_{efectiva}} \tag{5.29}$$

En el equilibrio, tal y como se establece en la ecuación 5.17 obtenemos la siguiente expresión:

$$L_{efectiva} = K\sqrt{R_{cantidad}} \tag{5.30}$$

$$K = e_{unit} C' \tag{5.31}$$

Esto nos indica que existe una relación implícita entre el número de individuos y la cantidad de recursos disponible y, a partir de los datos

experimentales obtenidos en la tabla de la figura 5.18, que se muestran en la figura 5.19 y que pueden ser ajustados a una curva polinómica de segundo orden con gran exactitud (coeficiente de regresión R=0.994).

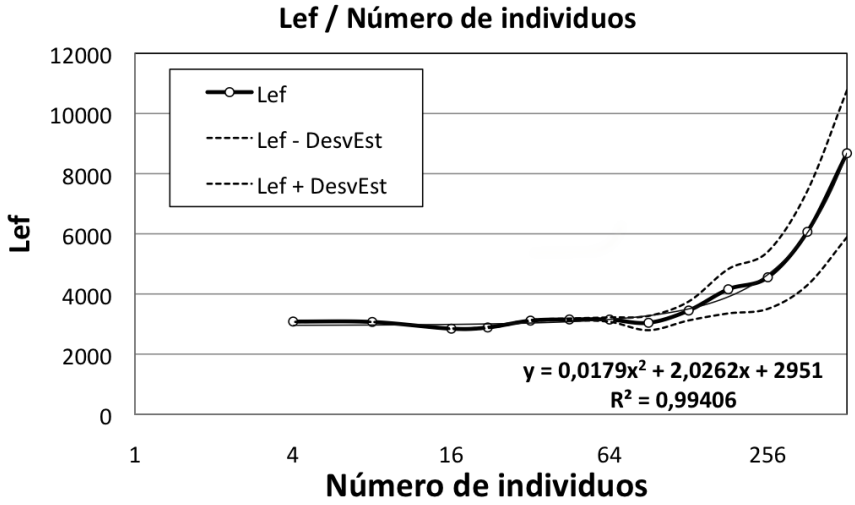


Figura 5.19: Valor de la Longitud Efectiva frente al número de individuos.

Y por tanto, esto nos indica que el balance energético sigue una curva del tipo:

$$N_{ind} = a + \sqrt{b + \sqrt{R_{cantidad}}} \tag{5.32}$$

Tal y como se muestra en la figura 5.20 a partir de los datos obtenidos experimentalmente.

A partir de las dos curvas obtenidas en las figuras 5.17 y 5.20, podemos generar la gráfica mostrada en la figura 5.21.

En esta representación, se muestra el punto de equilibrio del sistema, con la configuración utilizada (el punto en el que las dos curvas se intersecan), así como la zona estable del sistema, que es aquella zona a la que tiende el sistema en la mayoría de sus estados. Variaciones en la calidad de la población desplazan ambas curvas a la derecha o izquierda produciendo un incremento o decremento de la densidad de recursos, lo cual es un indicador de la eficiencia de la población.

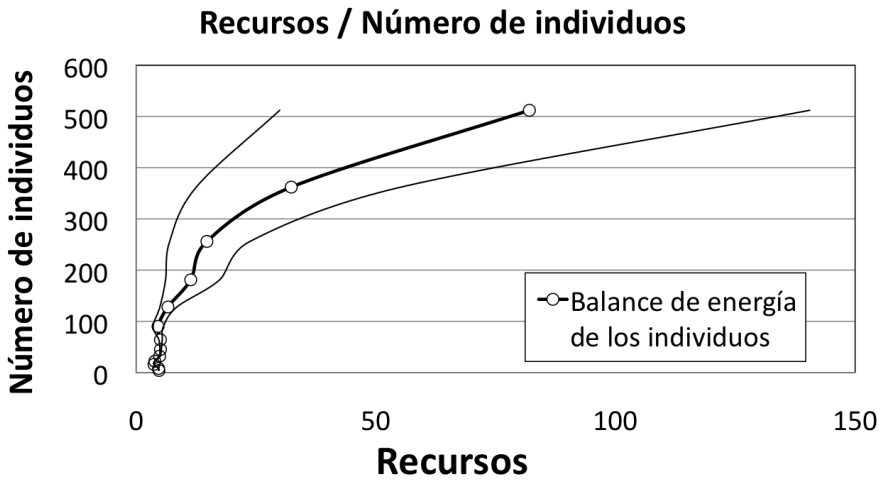


Figura 5.20: Relación entre los recursos y el número de individuos para el balance energético de la energía de la población de agentes.

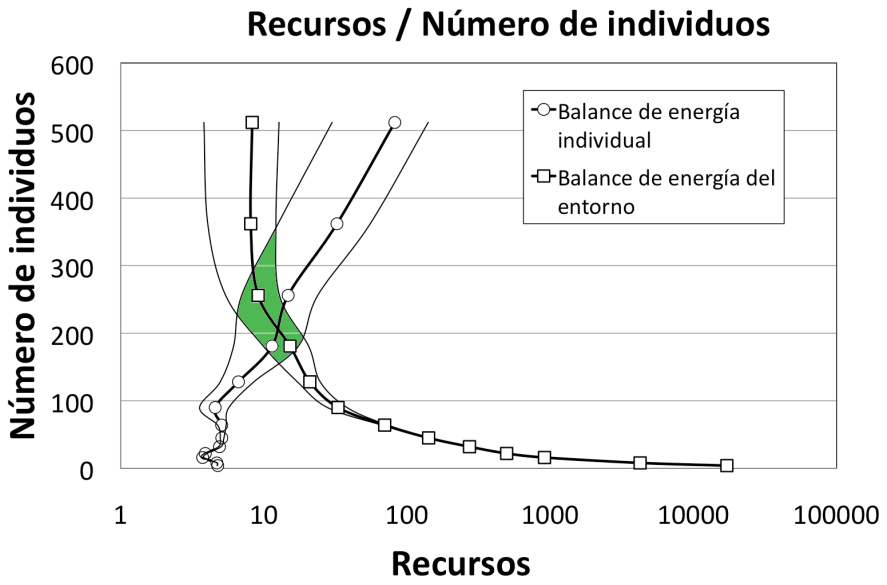


Figura 5.21: Representación del punto de equilibrio y de la zona estable del sistema.

5.5.2.1. Generación de un simulador a partir del modelo energético

Finalmente, a partir de las ecuaciones discretas 5.10 y 5.15 y de los parámetros de configuración concretos del escenario, se generan una simulación de la evolución de los valores de energía del entorno, energía de la población de agentes y número de agentes. Es decir, se genera un proceso iterativo basado en esas dos ecuaciones que permite obtener el comportamiento dinámico de la evolución del escenario.

Es necesario introducir también un valor estimado del tiempo promedio de la población para recorrer el grafo ($T_{promedio}$). Para la simulación dinámica que estamos generando, hemos utilizado un valor de tiempo promedio al inicio de la población y cuando han transcurrido la mitad de los pasos de tiempo. Debido a que este valor es altamente variable entre los distintos agentes de la población, se ha introducido también una variable que representa la desviación estándar en el tiempo promedio.

Debido a que las ecuaciones de equilibrio en las que se basa la simulación (ecuaciones 5.10 y 5.15), que son las que funcionan como atractores del sistema complejo que constituye el escenario y que lo conducen, en algunos casos, hasta alcanzar el punto estable, han sido formuladas y parametrizadas con los datos extraídos de las condiciones concretas de diseño de nuestro experimento, el punto de equilibrio será el mismo que el de la simulación real y los flujos de energía seguirán las mismas tendencias.

Debemos, sin embargo, destacar que la curva sintética que reflejará la secuencia de la combinación de parámetros que definen el estado del sistema no va a coincidir, o al menos no necesariamente, con la curva obtenida en una simulación. Esto es debido a que la componente aleatoria en la realización de las tareas en el grafo, así como el proceso de aprendizaje en cada evolución, serán diferentes y no modelizables. Con esto se quiere señalar que las curvas sintéticas obtenidas servirán de referencia para conocer cuán oscilante o amortiguada será la dinámica del sistema para determinadas configuraciones y condiciones iniciales en comparación con otras. También servirán para conocer cuál será el efecto de determinadas variaciones de los parámetros en esta dinámica y en las condiciones de

equilibrio finales, pero nunca substituirán a la evolución real.

Una evolución de este tipo de escenarios que se definen como Sistemas Complejos, como ya venimos señalando a lo largo de toda esta tesis, posee un número de elementos y de interacciones entre ellos que hacen imposible la determinación exacta de su comportamiento o la evaluación de todas las posibles estados que surgen de la infinidad de combinaciones que son susceptibles de generarse. La figura 5.22 muestra la evolución de la energía del escenario y del número de agentes durante la simulación. Se muestra en la figura 5.23 también, la representación de esta evolución en un diagrama de fases junto con las curvas de equilibrio correspondientes.

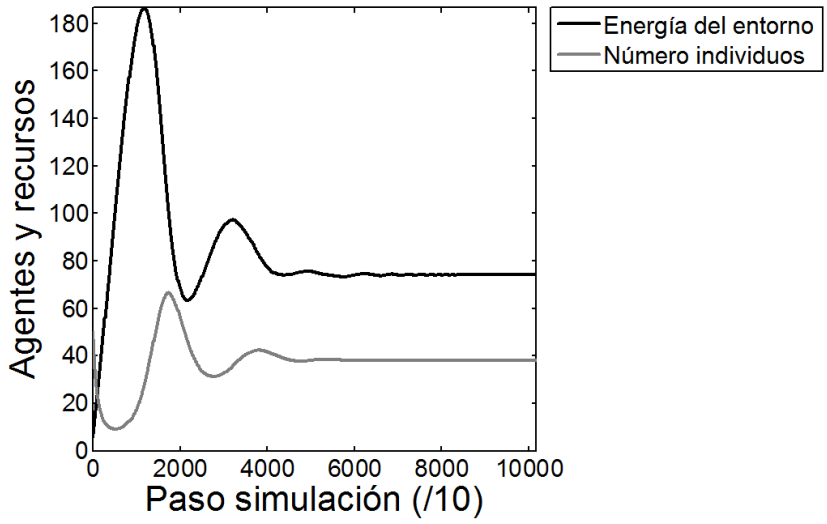


Figura 5.22: Evolución de la energía del escenario y del número de agentes durante la simulación.

A continuación, en las figuras 5.24 y 5.25 se muestra una evolución sintética en la que se ha incluido un valor no nulo de desviación estándar a lo largo de la evolución. Esto nos permite comprobar la variación que se produce en el proceso de simulación teniendo en cuenta las variaciones en el nivel de eficiencia debidas a variaciones en el nivel de dificultad de los grafos y/o a una población no homogénea.

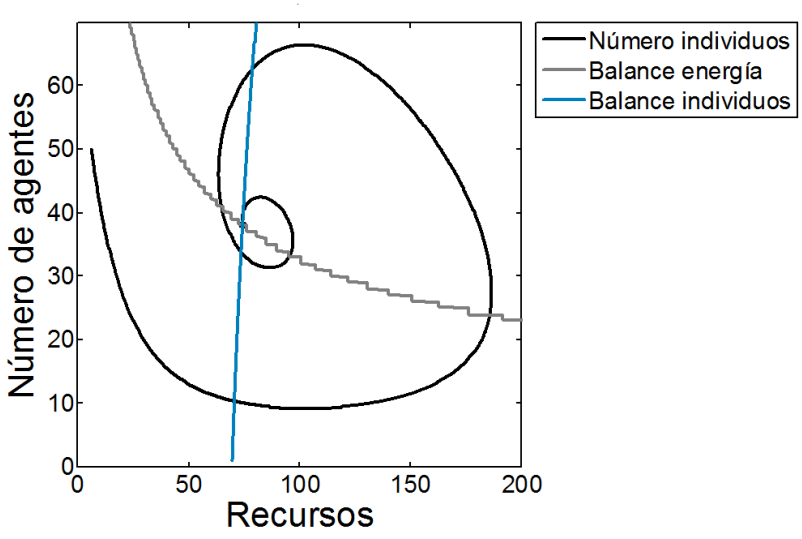


Figura 5.23: Diagrama de fases para la representación de la evolución del número de agentes frente a la energía del entorno (recursos).

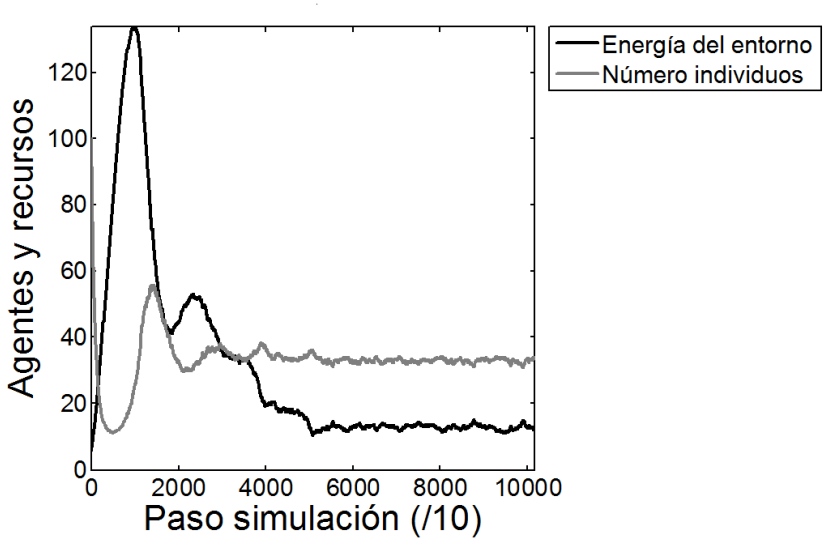


Figura 5.24: Evolución de la energía del escenario y del número de agentes durante la simulación.

Tal y como hemos visto en este análisis, utilizando las ecuaciones presentadas en este apartado y las curvas empíricas obtenidas para determinadas configuraciones fijas del escenario podemos, por un lado, conocer la dinámicas generales de la población sin necesidad de simular todos las posibles configuraciones, y por otro lado, estudiar cómo puede afectar cada uno de los parámetros que intervienen en el modelo energético al punto de equilibrio, tamaño de la zona estable, etcétera. Adicionalmente, mediante el uso de los modelos sintéticos que hemos creado, facilitamos enormemente el proceso de configuración de parámetros relevantes y generamos una herramienta de control que es de gran interés. Su principal implicación es que nos sirve para modificar los parámetros de manera que la solución resultante se estabilice en un tamaño de población y un nivel de recursos prefijado, también permite hacer un análisis de la sensibilidad ante la variación de cualquier parámetro. Finalmente este tipo de simulaciones es de utilidad para fijar configuraciones iniciales que nos lleven con mayor o menor velocidad hacia la zona estable, en función de si nos interesa o no acelerar esa convergencia.

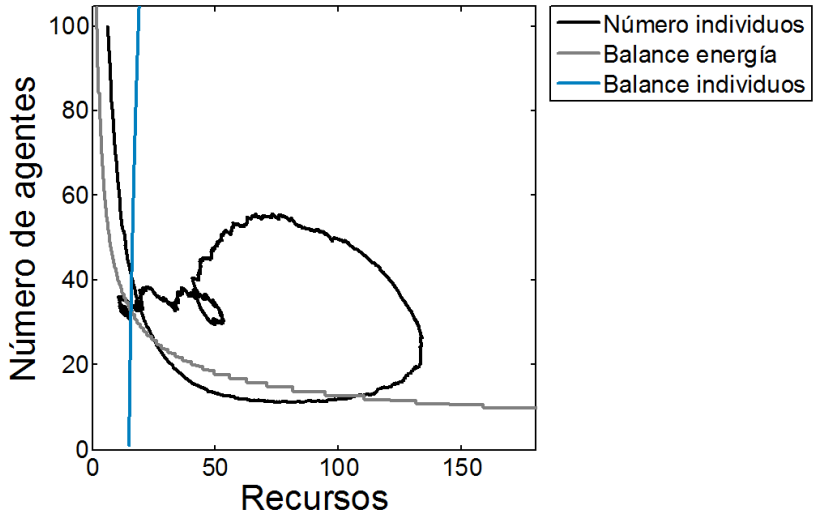


Figura 5.25: Diagrama de fases para la representación de la evolución del número de agentes frente a la energía del entorno (recursos).

5.6. Conclusiones

En este primer caso de estudio del CeAS en la resolución de un problema dinámico descentralizado, se ha generado un escenario que representa un problema de búsqueda de ruta óptima en un grafo. En una primera configuración, hemos comprobado el funcionamiento general del CeAS para evolucionar una población de agentes que realicen la tarea dentro del diseño de escenario. El resultado obtenido ha sido satisfactorio, proporcionando una solución similar a la que se utiliza en algoritmos bioinspirados como el ACO. En consecuencia, se ha probado también la aplicabilidad del procedimiento metodológico propuesto para el CeAS. Se ha profundizado también en el funcionamiento del WaspBed con respecto a las herramientas de evolución, cruce, gestión energética y monitorización de lo que ocurre en el escenario.

En un segundo y tercer experimento, se ha comprobado la flexibilidad tanto del diseño de escenarios mediante el CeAS como del uso de WaspBed para generar modificaciones en el escenario y estudiar las nuevas soluciones que se generan. Para ello, se llevaron a cabo modificaciones en las capacidades de los agentes y en la configuración del entorno, tras las cuales se analizó la adaptación de la población a estas nuevas condiciones mediante la obtención de un nuevo procedimiento de búsqueda.

Capítulo 6

Estudio del CeAS en un problema de exploración

6.1. Introducción y Objetivos

Como segundo ejemplo para el estudio del CeAS hemos utilizado otro de los problemas dinámicos descentralizados típicos en ingeniería, el problema de exploración autónoma de un entorno. Este problema consiste en recorrer un entorno desconocido con el objetivo de conocerlo, es decir, con el objetivo de recopilar información sobre él mismo.

De cara a tratar un problema de aplicación real más directa, hemos planteado la resolución de un subproblema dentro de la exploración, que es el problema de *vigilancia*. Un problema de vigilancia consiste, de un modo general, en establecer las estrategias para que un grupo de agentes (robots, sensores tales como cámaras, etcétera) detecten una serie de eventos que podrían generarse dentro de la superficie de exploración.

En este tipo de problema, el modelo sobre el que se trabaja está formado por un grupo de un número fijo de agentes que denominaremos *vigilantes* y cuya misión será la de cubrir, mediante el alcance de sus sensores, la superficie de una zona que requiere ser vigilada con la mayor frecuencia posible. Existen también una serie de zonas no alcanzables dentro de la superficie del entorno que podrían ser obstáculos o simplemente superficies que quedan fuera de la zona de exploración y que com-

plican la geometría de la zona de búsqueda.

El primero de los motivos de la elección de este problema es tratar un caso de aplicación real directa. La definición del problema estará, por tanto, centrada en reflejar, en la medida de lo posible, las condiciones en las que se podría aplicar a un problema real: número de agentes fijo, limitaciones en cuanto a sensorización, aplicabilidad de los procedimientos de evolución en tiempo de ejecución, etcétera. Una característica básica en este aspecto que debemos resaltar con respecto al modelo que vamos a utilizar en este ejemplo es que, al igual que la mayoría de los problemas de ingeniería reales, este es un problema con restricciones. Aunque no existan restricciones explícitas, el hecho de que el entorno posea unos límites y unas zonas no accesibles representa un conjunto de restricciones en el rango de las variables, y por otro lado, el tamaño de población de agentes es fijo, lo cual representa también una restricción de diseño en el número de recursos disponibles, que es común a muchas aplicaciones reales. Es decir, a diferencia del modelo anterior en el que el número de agentes exploradores de un grafo podía variar, en este caso la población está fijada de antemano y se mantendrá constante durante toda la simulación.

Además, este problema podemos encuadrarlo dentro de lo que definimos en el primer capítulo como problemas modelados mediante sistemas dinámicos, puesto que una de sus características más relevantes es el gran número de interacciones que vamos a encontrar entre los distintos elementos del escenario y que van a condicionar el comportamiento del mismo. En segundo lugar, entra dentro de los problemas de optimización combinatoria, pues el número de elementos existentes y de estados que puede presentar cada uno de ellos hacen que las posibles combinaciones de estos últimos que pueden generarse en el escenario sea muy elevada. Por un lado, cada uno de los agentes percibe al resto de agentes que se desplazan por el entorno y también es influido por las acciones de estos, y por otro lado, el uso de un escenario acotado tanto por los límites de entorno como por los obstáculos que hay en su interior, elevan el número de interacciones que se generan entre los vigilantes y entre los vigilantes y el escenario. Constituye, además, un problema idóneo para ser abordado mediante la simulación computacional, pues los métodos de análisis

alcanzarían un grado de complejidad que los haría irresolubles de forma práctica.

Otro de los objetivos que se pretende alcanzar con el problema de vigilancia, es el de resolver un problema de estructura descentralizada. A diferencia del anterior, en el que se buscaba encontrar un algoritmo de resolución distribuido y descentralizado para un problema no distribuido, en este caso, la definición del problema es inherentemente distribuida, pues se trata de un grupo de vigilantes que deben en conjunto cubrir una zona de vigilancia, y adicionalmente, el modelo que se presenta es descentralizado pues no se introduce ningún elemento de control o gestión de información centralizado. En efecto, el problema es fácilmente descentralizable si no se permite una interacción entre vigilantes que no sea a nivel local, tanto para la transmisión de información del entorno como para el intercambio de información genética.

Por otro lado, se pretende en este apartado llevar a cabo un estudio de las características del CeAS como algoritmo evolutivo en su implementación para este caso concreto, de manera que podamos caracterizar el algoritmo en términos de una serie de parámetros básicos, que aglutinen varios de los coeficientes que intervienen en el proceso de modelado del problema a resolver y de los elementos que se generan para transformar dicho problema al paradigma del CeAS. Lo que se busca con este análisis, al igual que con el estudio energético realizado en el caso anterior, es sintetizar el gran número de parámetros que intervienen en el modelo generado para cualquier escenario y transformarlos en un conjunto más reducido de parámetros que representarán conceptos de más alto nivel, y que pueda ser común a cualquier otra implementación del CeAS y por tanto, ayudar en la definición de la misma y la traducción de un problema al paradigma CeAS.

Así, en primer lugar, se podrá facilitar en gran medida la interpretación de lo que está ocurriendo en una simulación en la que el gran número de elementos y parámetros que intervienen hace muy compleja dicha interpretación. En segundo lugar, permitirá facilitar los mecanismos de control sobre la evolución, es decir, conocer qué parámetros son relevantes para conseguir determinados efectos en el proceso evolutivo (aumentar la exploración o explotación, disminuir la aleatoriedad en la evaluación, etcéte-

ra). En último lugar, se buscará poder extrapolar los criterios de análisis y los métodos de control entre diferentes escenarios.

Se pretende también utilizar este caso de aplicación para estudiar comparativamente la eficiencia del CeAS frente a otras aproximaciones encontradas en la bibliografía para obtener comportamientos colectivos: un algoritmo genético clásico y un algoritmo híbrido basado un evolutivo para poblaciones de robots autónomos (Groβ y Dorigo, 2008) que ha dado buenos resultados. Esta comparación es básica para tener una validación formal del uso del CeAS en la evolución de poblaciones y se ha utilizado este ejemplo de vigilancia para llevarla a cabo por dos motivos: en primer lugar, porque estamos trabajando con un tamaño de población fijo, requisito para utilizar las aproximaciones que planteamos como comparación. En segundo lugar, el problema de vigilancia puede ser resuelto mediante una población homogénea de agentes, sin una merma considerable de la eficiencia, lo que permite obtener una comparativa más justa al no tener en cuenta la capacidad de los algoritmos para manejar individuos heterogéneos.

Tal y como habíamos visto en el apartado 3.2.1, si pretendemos evolucionar una población de agentes, el uso de algoritmos evolutivos convencionales no resulta adecuado a menos que la población sea homogénea, porque o bien el espacio de búsqueda aumenta enormemente o se requieren simplificaciones que solo dan buen resultado en tareas sencillas. De este modo, aunque el CeAS permitiría la evolución de una población de tamaño variable y heterogénea, utilizamos este entorno en el que la población es fija y las tareas del entorno no están fuertemente diferenciadas. En consecuencia, en este escenario, la formación de especies de individuos no aporta una ventaja evidente y por tanto nos permite comparar en igualdad de condiciones las tres estrategias de evolución. Esta comparación se explicará en detalle y se expondrán los resultados en el apartado 6.5.2.

Debido a que el tamaño de la población es fijo en este caso, el análisis energético no será de aplicación para el control y estudio de las dinámicas de población. Sí será necesario el estudio de la distribución de recompensas que recibirán los vigilantes en función de los parámetros que se consideren relevantes, tal y como se explicó en el apartado 3.2.1.5 para

obtener el comportamiento colectivo deseado en función de lo comportamientos individuales de cada agente vigilante.

El problema de vigilancia planteado presenta, por tanto, una estructura idónea para ser tratado con el CeAS y servirá para profundizar y estudiar el funcionamiento y aplicación del algoritmo. Pasamos a continuación a presentar con detalle la configuración experimental que se ha diseñado.

6.2. Configuración experimental

En este apartado pasamos a describir los elementos que van a formar parte del escenario de vigilancia y cuáles son las interacciones que se producirán entre estos de acuerdo a la metodología de diseño propuesta en el capítulo 4. Básicamente, tal y como hemos comentado, el problema está constituido por los *vigilantes*, una *superficie* que requiere de vigilancia y una serie de zonas inaccesibles que denominaremos *obstáculos*. Veremos, a continuación, cómo se ha diseñado el escenario para representar el problema que queremos resolver y para permitir la generación de soluciones válidas y transferibles al problema real.

Definición de los individuos

Al igual que en el ejemplo anterior, respecto a la definición de los individuos vamos a describir, fundamentalmente, los sensores que poseen, las acciones que pueden ejecutar, las características del sistema de control y los parámetros evolucionables que forman su genotipo.

Teniendo en cuenta que la tarea que se pretende optimizar está distribuida en un grupo fijo de vigilantes, los individuos estarán representados por cada uno de estos agentes de vigilancia. La acción básica de los vigilantes es la de desplazarse sobre el escenario para llevar a cabo la observación de las distintas zonas. Por tanto, estos agentes se desplazan sobre una malla de celdas que discretiza la superficie observable. A medida que las distintas celdas entran dentro del alcance del rango de sensorización de los agentes, se considera que estas son vigiladas.

En cuanto a la información que reciben los vigilantes del entorno y del resto de la población de agentes, estos son capaces de detectar la presencia de otros agentes en las inmediaciones y las colisiones con obstáculos. Se ha optado por esta sensorización para representar una implementación en agentes reales como en el caso de un grupo de robots, una flota de aviones autónomos, etcétera. Las sensorizaciones básicas serían las que detectan obstáculos o zonas no accesibles y también la presencia de otros vigilantes. El nivel de vigilancia de cada una de las celdas sería de gran utilidad, pero no es fácilmente extraíble en un problema real y por eso no vamos a incluirlo en el diseño inicial.

La sensorización con la que se han diseñado los vigilantes tiene como objetivo:

- Permitir cierto nivel de coordinación mediante la detección de otros vigilantes al tener en cuenta las zonas cercanas que están siendo más o menos vigiladas.
- Detectar si las órdenes del sistema de control están teniendo un reflejo real en el estado del agente dentro del entorno, es decir, si los actuadores están siendo efectivos o si alguna restricción está limitando su funcionamiento (obstáculos o límites del entorno).
- Permitir el uso de información temporal mediante una memoria de acciones.

Los sensores que se han generado son, por tanto:

1. El **sensor de detección de intrusos**, este sensor realiza la tarea que denominamos de vigilancia, esto es, detecta la presencia de algo anómalo en una celda. Posee un área de visión como la que se muestra en la figura 6.1. Los parámetros que afectan a este sensor son: el alcance del área de visión y el ángulo de visión. El área observada en cada instante es función de la posición y orientación de cada agente. Esta superficie está definida por una elipse, cuyos focos están situados, uno en la posición del vigilante y otro en un punto situado a una distancia fija del primer foco y en la dirección marcada por la orientación del agente en cada instante. La longitud del eje principal de la elipse y la suma de las distancias a los

focos de los puntos que pertenecen a la elipse se obtienen a partir del valor del parámetro alcance de visión que posee cada vigilante. Adicionalmente, se define el ángulo de un punto dentro del área de visión como el ángulo formado entre el eje mayor de la elipse y el vector que resulta de unir la posición del vigilante con la posición del punto en cuestión. Para que un punto esté dentro del área de visión, este ángulo debe ser menor que el ángulo de visión definido en los parámetros del vigilante.

2. Un **sensor de detección de vigilantes**. Este sensor posee un área de observación con la misma forma y orientación que el sensor de visión de celdas, pero el eje mayor de la elipse y la suma de las distancias de los puntos interiores a esta elipse no tienen el mismo valor: por defecto son cinco veces mayores. El parámetro que regula el tamaño de esta área de observación se denomina alcance del sensor de detección de vigilantes y es fijo para todos los agentes. Adicionalmente se han diferenciado cuatro sectores de visión para el área de detección de vigilantes en función de un rango de ángulos tal y como se muestra en la figura 6.1: los dos primeros sectores corresponden a aquellos puntos situados con un ángulo menor de 45º bien hacia la derecha o hacia la izquierda. Los otros dos sectores van desde un ángulo de 45º hasta el ángulo de visión definido por cada agente. Sobre estos cuatro sectores se definen cuatro valores de sensorización que serán enviados al sistema de control y que representan el número de agentes que un vigilante detecta en cada uno de sus cuatro sensores de visión.

3. Otro sensor es el denominado **sensor de trayectoria**. La información que almacena y envía al sistema de control con respecto a la memoria almacenada, se obtiene mediante la generación de un vector que une la posición actual del vigilante con la posición n instantes de tiempo atrás. Se calcula el ángulo y el módulo del vector y se obtienen las dos entradas del sistema de control que corresponden a esta memoria de itinerario. Cabe destacar que la orientación del vector es, debido a que la coordenadas de la trayectoria son relativas, relativa a la orientación actual del vigilante, y así debe ser pues la salida del sistema de control es un ángulo que representa la variación respecto al ángulo actual.

4. El **sensor de obstáculos**, que almacena información relativa al ángulo de choque con respecto al obstáculo y a la distancia del obstáculo con respecto a la posición inicial del vigilante. Se basa en el vector que resulta de unir el punto de destino inicial para el movimiento del agente, es decir, el punto que se alcanzaría si no se encontrase con el obstáculo, con el punto final realmente alcanzado. Lo denominaremos vector de variación de trayectoria. La proyección de este vector sobre la dirección perpendicular al avance del agente nos dará una de las entradas al sistema de control (ver figura 6.2). Por lo tanto, el signo de esta proyección nos indica si el obstáculo nos lo hemos encontrado por la derecha o por la izquierda, y su valor absoluto indica principalmente el ángulo de choque con el obstáculo. Cuanto menor es este valor absoluto, la dirección del vigilante está más cerca de ser perpendicular a la pared del obstáculo.

5. Un último sensor es el sensor de **detección de riesgo**. Este sensor es equivalente al sensor de detección de intrusos en cuanto al área de acción, pero posee, al igual que el sensor de detección de vigilantes 4 sectores diferenciados. Su función es la de enviar al sistema de control el valor de riesgo que detecta en cada uno de los cuatro sectores. De cara a la realización de experimentos, llevaremos a cabo pruebas con el sensor activado y desactivado y esto es debido a que dentro de los problemas de exploración podemos encontrar aplicaciones como la de vigilancia, en la que no es posible detectar por parte de los vigilantes el nivel de riesgo de una zona o aplicaciones de recogida de materiales o limpieza en la que sí es posible. En función de la activación o no de este sensor obtendremos un nivel de exploración u otro tal y como veremos en los resultados.

Cualquier celda que caiga dentro del área de visión de alguno de los vigilantes pasa a estar vigilada, y por tanto, reinicia su marcador de tiempo sin ser observada. Al mismo tiempo, la cobertura de una celda genera una recompensa energética para el vigilante. Estos parámetros que modifican la zona de visión de los agentes son fijos para toda la población y no se han añadido a la lista de parámetros evolucionables, por tanto, no cambiarán con el paso de las generaciones, sino que poseen un valor definido por el diseñador, aunque podrían evolucionarse para combinar una evolución morfológica y una evolución de comportamiento.

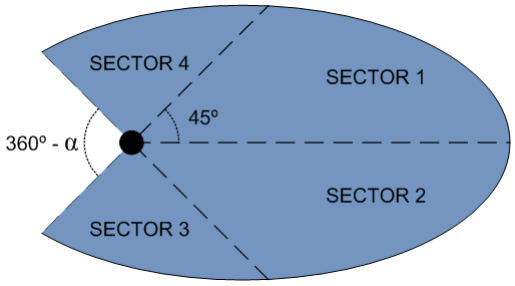


Figura 6.1: Sensores de detección de vigilantes.

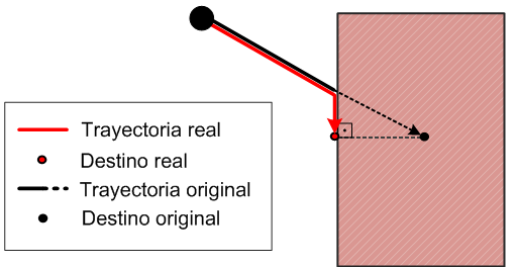


Figura 6.2: Modificación de la trayectoria en una colisión y sensor de obstáculos.

En cuanto a los movimientos que realizan los agentes en el escenario, y que realmente definirán la eficiencia en la realización de la tarea de vigilancia, están regulados por dos parámetros: la velocidad de avance (v_a) y la velocidad angular ($\hat{\theta}$). La velocidad de avance es un parámetro fijo para toda la población. De no ser así tendríamos que asociar algún tipo de coste (en términos de eficiencia) al aumento de velocidad, pues existe una relación directa e independiente frente al resto de los parámetros entre la velocidad y el número de celdas vigiladas por unidad de tiempo, de modo que la población convergería hacia el máximo permitido de velocidad. La velocidad angular es modificada por el sistema de control en cada paso de tiempo y es la que realmente define el comportamiento de cada agente.

La trayectoria realizada es una función de las sensorizaciones que recibe el sistema de control. Una vez que está definido el nuevo ángulo (θ_{t+1}), la variación en la posición del vigilante ($\overrightarrow{\Delta P}$) viene dada por un vector cuyo ángulo es el nuevo ángulo y cuyo módulo es la velocidad de avance (v_a):

$$\overrightarrow{\Delta P} = v_a(\cos\theta_{t+1}, \sin\theta_{t+1}) \tag{6.1}$$

En el caso de que aparezca un obstáculo que interrumpa la trayectoria de avance de un agente, éste avanza paralelo a la pared del obstáculo hasta alcanzar en esta trayectoria la proyección en dirección perpendicular a la pared del punto inicial de destino sobre la pared del obstáculo tal y como se en la figura 6.2. La línea discontinua indica el tramo de trayectoria inicial que se adentra en el obstáculo, la línea negra indica la trayectoria real generada y la línea gris indica la proyección que determina el nuevo punto de destino.

Los agentes poseen también una *memoria de itinerario* que será utilizada como sensor interno para el sistema de control. Esta memoria contiene los desplazamientos realizados por el agente en los últimos n pasos de tiempo. Los vigilantes no poseen un sistema de posicionamiento que les indique su situación espacial en cada instante, pero sí conocen la variación del ángulo entre su posición anterior y la actual. Esto les permite conocer los desplazamientos relativos realizados en cada movimiento, y esto es lo que se almacena en la memoria. El número de pasos n que se almacena en la memoria es un parámetro descriptivo de cada agente

y es evolucionable. En función de este parámetro, el agente decide darle importancia a los movimientos realizados en los últimos pasos para seleccionar su próximo movimiento o bien utilizar un valor que represente la variación de la trayectoria a más largo plazo.

En cuanto al sistema de control de los vigilantes, necesitamos una estructura de decisión que nos permita generar estrategias o comportamientos lo suficientemente complejos como para realizar de forma satisfactoria las tarea de vigilancia a partir de las sensorizaciones definidas. Teniendo en cuenta que las entradas de este decisor van a ser continuas y que el comportamiento debe considerar interrelaciones entre sus entradas y no valores absolutos o independientes de las mismas hemos encontrado que las redes de neuronas artificiales cumplen las especificaciones requeridas y son por tanto, adecuadas para nuestro sistema de control. Concretamente, el tipo de red seleccionada para este ejemplo es una *Red Neuronal de Base Radial* (RBN) que posee la siguiente estructura (ver figura 6.3):

- 7 neuronas en la capa de entrada: 5 para los sensores de entrada (sensor de colisión y 4 sensores para los sectores de visión) y 2 más representando la variación de posición que nos proporciona la memoria de itinerario.
- 1 neurona de salida que proporciona la variación del ángulo de orientación del vigilante.
- 3 neuronas en la capa oculta: para decidir el tamaño de la capa oculta se han probado distintos tamaños en la configuración base, y en función del nivel de eficiencia conseguido en la realización de la tarea de vigilancia, se ha seleccionado éste como el más idóneo.

Esta configuración requiere un total de $7x3$ pesos para conectar la capa de entrada con la capa oculta, 3 desviaciones para las funciones de base radial de las neuronas de la capa oculta y $3x1$ pesos para conectar la capa oculta con la capa de salida. Esto hace un total de 27 valores reales, que junto con la longitud de la memoria de trayectoria, representan los 28 genes del cromosoma de cada uno de los individuos. El sistema de control se ejecutará en cada paso de tiempo para cada agente después de haber actualizado los valores de los sensores, y a continuación, se ejecutará el

cambio de posición del vigilante y se realizará la tarea que actualiza los valores de la superficie vigilada.

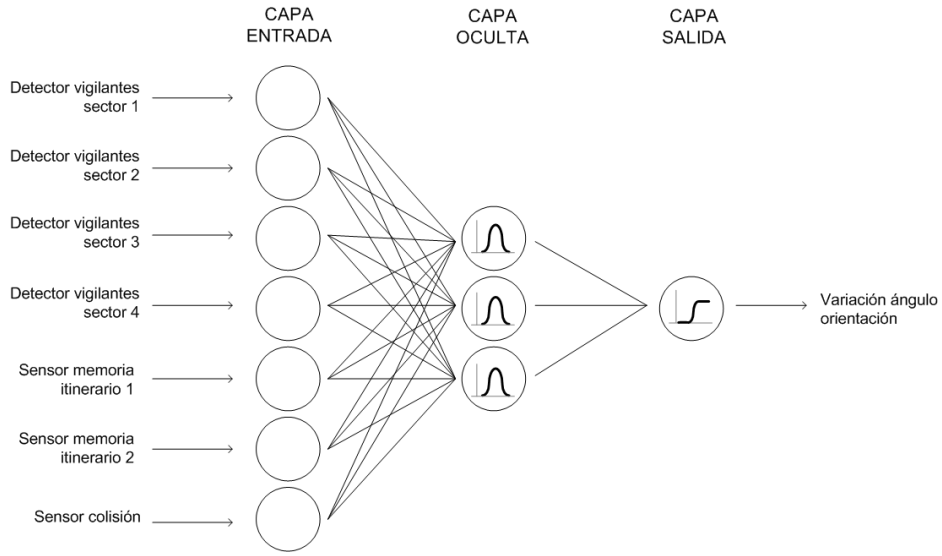


Figura 6.3: Sistema de control de los vigilantes.

Modelado distribuido y descentralizado

El entorno en este escenario está representado por dos elementos: los obstáculos y la malla de las celdas (ver figura 6.4). A continuación definiremos cada uno de estos elementos y las acciones que hemos establecido para cumplir los requerimientos de interacción descentralizada.

Los obstáculos: son elementos rectangulares cuya posición y tamaño se genera aleatoriamente en cada escenario de simulación. Suponen restricciones para el avance de los vigilantes en el escenario y permiten generar geometrías de la superficie de exploración más complejas. La posición es fija durante cada simulación.

Los límites del escenario se han modelado también como obstáculos que ocupan todo el perímetro del entorno. Cuando un agente se encuentra con un obstáculo cambia su trayectoria y este cambio es detectado por

los sensores de colisión del agente.

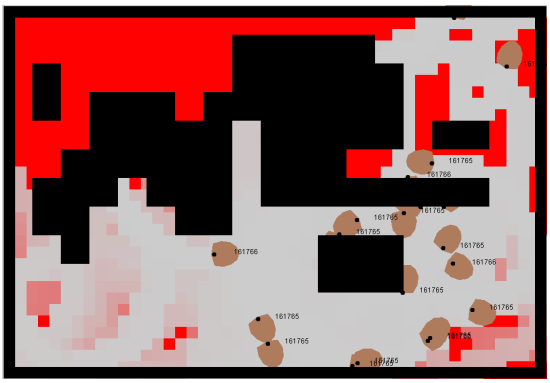


Figura 6.4: Escenario de vigilancia. Los robots se han representado mediante un punto negro y un área que indica su rango de detección de intrusos. Los obstáculos mediante zonas negras y la malla de celdas mediante cuadrados que cambian su color en función del nivel de riesgo que poseen, a mayor riesgo, mayor componente roja en el color de cada celda.

Malla de celdas: : el escenario está cubierto por una malla de celdas que lo discretizan espacialmente. Esta discretización permite simplificar los cálculos y el análisis del nivel de vigilancia en cada instante frente al uso de un modelo continuo y no representa una pérdida de generalidad. Visto de otro modo, cada una de las celdas representa los elementos que deben ser visitados regularmente para llevar a cabo la tarea de vigilancia.

Las celdas poseen una variable de tiempo que se reinicia cuando el centro de la celda es cubierto por el rango de visión de algún vigilante. Asociado a esta variable de tiempo existe un valor de riesgo que sigue una función que variará dependiendo de los requerimientos del problema, de manera que el riesgo global del entorno es el sumatorio de los valores de riesgo de todas las celdas del escenario. Por otro lado, asociado a este valor de riesgo se genera un valor de recompensa que será el que reciban los agentes cuando cubren una celda.

Las interacciones entre los elementos y los agentes en el escenario se estudian con respecto a los cuatro niveles especificados en la metodología:

- **Información de entorno:** la información que reciben los agentes de vigilancia respecto al entorno es la que obtienen a partir de los sensores de colisión cuando se encuentran con un obstáculo. No existe información con respecto al nivel de riesgo de las celdas cercanas y es importante tener esto en consideración porque complica la tarea de manera importante.
- **Información de grupo:** el flujo de información entre individuos se realiza de forma automática y unidireccional a través de la sensorización de los vigilantes cercanos. La sensorización es local y parcial, y permite obtener el número de agentes situados en cada uno de los cuadrantes del sensor de visión de agentes. Esto permite a los vigilantes tener una idea de las zonas cercanas que están siendo más o menos vigiladas.
- **Actuación sobre el entorno:** las interacciones que se realizan entre los individuos y el entorno son las derivadas de la tarea de vigilancia, es decir, la actualización del valor de riesgo de cada celda. Por cada celda vigilada, el vigilante recibe una recompensa que es proporcional al nivel de riesgo de dicha celda.
- **Actuación en el grupo:** la única interacción que existe entre miembros de la población es la de reproducción. Esta se explicará en el apartado de selección reproductiva.

Estudio de la utilidad privada y utilidad global

La asignación de recompensas de manera que el comportamiento individual de los vigilantes conduzca hacia el objetivo global propuesto por el diseñador, no es una tarea trivial tal y como expusimos en el apartado 3.2.1.5. Para este problema concreto lo hemos resuelto de la forma siguiente: tal y como hemos definido el problema, la utilidad global se relaciona con el nivel de riesgo en todo el escenario, a menor nivel de riesgo, mayor utilidad. El nivel de riesgo del escenario será, por tanto, la suma de todos los niveles de riesgo de cada una de las celdas en las que se divide la malla del escenario.

$$R_e^t = \sum^{celdas} R_n^t \tag{6.2}$$

$$R_n^t = F_r(T_n) \tag{6.3}$$

Siendo:

- R_e^t el nivel de riesgo global del escenario para un instante concreto.
- R_n^t el nivel de riesgo de la celda n en el instante t.
- F_r la función que relaciona el nivel de riesgo con el tiempo sin vigilancia de una celda.
- T_n el tiempo transcurrido desde el último instante en el que la celda n fue vigilada hasta el instante actual.

Algunas de las formulaciones del problema de vigilancia proponen como función de utilidad global el valor máximo de riesgo de todas las celdas del escenario. Este tratamiento sin embargo, no es adecuado en nuestro caso ya que no permitiría una resolución descentralizada, pues cada uno de los agentes necesitaría de información global para obtener el valor de recompensa real, por lo que no será utilizada aquí.

El valor de riesgo asociado al tiempo sin cobertura de cada una de las celdas lo determinan las especificaciones de diseño del problema. En una primera aproximación, la función que regula el riesgo es la que se muestra en la figura 6.5. Tal y como se ve, esta relación genera un nivel de riesgo bajo hasta alcanzar el inicio del intervalo crítico, en este caso en torno a 300 pasos de tiempo. A partir de ahí, el riesgo crece de modo aproximadamente lineal hasta alcanzar final del intervalo crítico. Esta relación se basa en la distribución de nivel de riesgo asociado a un problema real de vigilancia.

El inicio del intervalo crítico representa el tiempo necesario para llevar a cabo una intrusión si esta fuese realizada en el instante cero, el momento en el que el vigilante abandona una zona. Antes de alcanzar el intervalo crítico, el tiempo que una zona permanece sin vigilancia no es suficiente para una intrusión, por tanto, el riesgo es prácticamente nulo. Una vez superado el límite que supone el inicio del intervalo crítico, las

probabilidades de sufrir una intrusión crecen linealmente, pero a medida que pasa el tiempo las probabilidades de intrusión se mantienen constantes, pues el intruso valora como más probable la llegada de un vigilante cuanto más tiempo lleve una zona sin vigilar y prefiere esperar a que el vigilante vuelva a pasar.

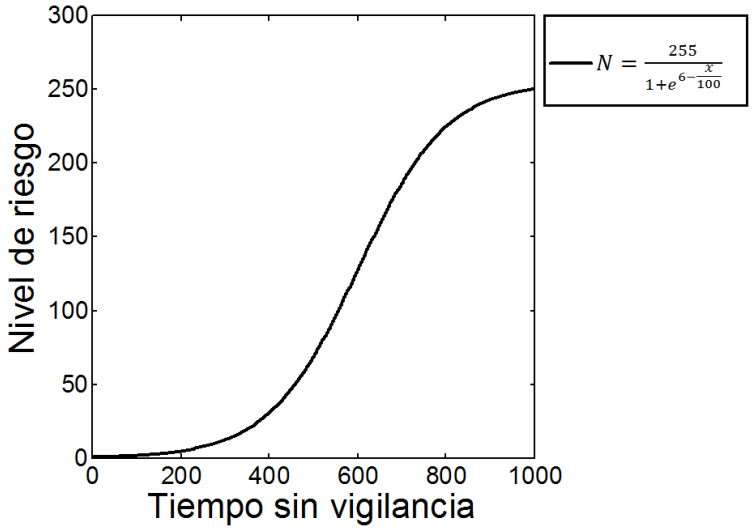


Figura 6.5: Representación del nivel de riego con respecto al tiempo sin vigilancia para cada celda.

A continuación debemos crear una función de calidad individual que ha de cumplir las siguientes condiciones: (ver apartado 3.2.1.5 y artículos Agogino y Tumer (2008); Wolpert y Tumer (2000))

- La utilidad individual debe estar alineada con la utilidad global, es decir, que un individuo mejore su utilidad individual ha de implicar que ha habido una mejora en la utilidad global
- La utilidad individual debe ser más sensible a las acciones propias que a las del resto de la población.

En consecuencia, utilizar la utilidad global como utilidad individual, aunque cumpliría la primera condición, no cumpliría la segunda condición.

Según la metodología del COIN, una posible medida de utilidad sería lo que se denomina utilidad diferencial, que correspondería a usar la utilidad global obtenida tras una acción realizada por un agente menos la utilidad global eliminando los efectos de las acciones de este agente:

$$D_i = G(z) - G(z_{-i} + c_i) \tag{6.4}$$

Siendo D_i el nivel de calidad global actual y $G(z_{-i} + c_i)$ el nivel de calidad global eliminando los efectos de las acciones del agente i.

Para nuestro problema, en primer lugar, utilizamos un nivel de riesgo en lugar de un nivel de calidad, con lo cual nuestro objetivo será reducir el valor de riesgo.

La ecuación 6.5 muestra en el primer término el riesgo global, que está representado por el sumatorio del riesgo de todas las celdas. El segundo término representa el valor de calidad global eliminando los efectos de las acciones de vigilancia de un agente concreto, siendo:

- $celdas^t_{-i}$ el grupo de celdas que no han sido visitadas por el agente i en el paso de tiempo t;

- $celdas^t_i$ el grupo de celdas visitadas por el agente i en el paso de tiempo t.

$$D_i = \sum^{celdas} R^t_n - \left(\sum^{celdas^t_{-i}} R^t_n - \sum^{celdas^t_i} R^{t-1}_n \right) \tag{6.5}$$

$$D_i = \sum^{celdas^t_i} R^t_n - \sum^{celdas^t_i} R^{t-1}_n \tag{6.6}$$

Teniendo en cuenta que el riesgo en las celdas vigiladas por el agente i en el instante t es igual a cero, tenemos que:

$$D_i = - \sum^{celdas^t_i} R^{t-1}_n \tag{6.7}$$

Lo cual nos indica que la reducción del riesgo diferencial asignado al agente i en el instante t es el sumatorio del nivel de riesgo que poseían

las celdas vigiladas por el agente en el instante $t-1$. Debido a que el objetivo es reducir el nivel de riesgo, la calidad privada de ese agente i será el valor de la reducción de riesgo que hemos calculado:

$$U_i = -D_i = \sum^{celdas_i^t} R_n^{t-1} \tag{6.8}$$

Esta utilidad posee cierta sensibilidad a las acciones del resto de la población debido a que el nivel de riesgo de una celda es mayor o menor en función de si una zona es más o menos transitada por la población de observadores. Sin embargo, teniendo en cuenta que disponemos sólo de interacciones locales entre los agentes y que la medida de utilidad no puede requerir de un elemento central que obtenga información de toda la población, esta medida se propone como una solución muy adecuada. Por otro lado, tal y como hemos expuesto, es consistente con aquella propuesta por los autores de la metodología COIN, y en efecto, ha generado buenos resultados como expondremos más adelante.

En resumen, la utilidad individual será la suma de los niveles de riesgo que poseían las celdas en el momento de ser observadas por un individuo.

Gestión energética

En este escenario, la tarea principal es la de vigilar las celdas del recinto, y por tanto, el diseño se ha llevado realizado de modo que el valor de energía del que disponen los vigilantes representa las recompensas obtenidas de las celdas. Este valor de energía representa su utilidad en cada instante, y debido a que la tarea requiere de un proceso continuo de vigilancia, existe un gasto energético asociado a cada paso de tiempo.

- **Entrada de energía:** la entrada de energía que ocurre en el escenario se produce a través de las celdas. Estas generan un valor de energía proporcional a su valor de riesgo. Este valor de energía, que se transmite a la población de vigilantes de forma continua, no constituye una entrada constante de energía debido a que la función de riesgo respecto al tiempo no es lineal (ver figura 6.5) y depende del nivel de riesgo de cada celda. Esto complicaría un análisis energético de las dinámicas poblacionales del sistema como el realizado en

el ejemplo anterior, pero en este caso el tamaño de población es fija, por lo que no ha lugar dicho análisis.

- **Salida de energía:** la salida de energía se produce con el gasto energético por unidad de tiempo de los vigilantes. Es decir, teniendo en cuenta que este gasto es fijo y que el número de vigilantes es fijo, este flujo de energía es constante.

- **Intercambio energético:** el intercambio energético entre agentes se produce sólo durante la reproducción. Más adelante explicaremos en detalle el nuevo modo de reproducción que hemos desarrollado para permitir la evolución con una población constante. Comentaremos ahora, para poder explicar este aspecto de la gestión de energía que, cuando se produce la generación de un nuevo individuo, este reemplaza a uno de los existentes y se le suministra un valor de energía que es la mitad del promedio del valor de energía de los dos individuos base.

- **Asociación energía-utilidad:** como ya comentamos, la energía se produce como resultado de un valor de recompensa asociado al nivel de riesgo de cada celda vigilada, por lo que la asociación entre energía y utilidad viene dada por la función mostrada en la figura 6.5 y la proporciona el diseñador.

Selección reproductiva

Debido a la restricción que hemos impuesto para la equivalencia del escenario propuesto en implementaciones reales de utilizar un número de vigilantes fijos, no podemos utilizar el mismo procedimiento de selección reproductiva que en modelos con población variable. Para garantizar que no varía la población, la única solución plausible es que cada vez que se genere un nuevo individuo, otro desaparezca. Al mismo tiempo la selección reproductiva y la generación de individuos base (ver apartado 4.1.2 para su definición) deben estar supeditadas al cumplimiento de ciertas condiciones a lo largo de la evolución del escenario.

Para todo ello, la reproducción de los agentes se realizará mediante un procedimiento que hemos diseñado específicamente para el caso de

poblaciones de tamaño fijo y que hemos denominado **reproducción basada en embriones**.

Reproducción basada en embriones:

El procedimiento es el siguiente: cada agente posee, además de todo el conjunto de sus parámetros, una copia de los parámetros evolucionables de un nuevo agente potencial que denominaremos **embrión** (ver figura 6.6) y que tiene asociado una valor de pre-utilidad que representa una estimación de la utilidad que tendrá el agente potencial que se generará con ese embrión. Inicialmente, cuando se genera un nuevo vigilante, se crea también un embrión asociado mediante una mutación de los parámetros de este vigilante y se le asocia un valor de utilidad que será la mitad de la utilidad del vigilante asociado.

A lo largo de la vida (tiempo de simulación) de un agente, este puede modificar el embrión que tiene asociado. Esto ocurre cuando un agente se encuentra a otro, es decir, está situado dentro de un área de observación para la reproducción de aquel. Esta nueva área, al igual que en el caso del sensor de detección de vigilantes y del de observación de celdas, posee también un parámetro que es fijo para todos los agentes y que se denomina alcance del área de reproducción. Por tanto, una vez que un agente detecta a otro en su área de reproducción, comprueba mediante el nivel de energía (utilidad) del otro agente la idoneidad de reproducirse con él. En caso de valorarlo positivamente, es decir, que la pre-utilidad del nuevo embrión sea superior a la actual, se cruzan los códigos genéticos para generar un nuevo conjunto de parámetros evolucionables, y el embrión actual es sustituido por el nuevo embrión.

La pre-utilidad de un nuevo embrión se estima a través de la media de utilidad entre los dos vigilantes implicados en la reproducción. Es claro que esta estimación tiene pocas probabilidades de coincidir con la utilidad real que luego resultará cuando se genere el nuevo vigilante, y en efecto, no se basa en ningún análisis del modo de cruce ni tampoco en un estudio estadístico sobre simulaciones anteriores. Sin embargo, esto no es relevante dado que lo que estamos haciendo con ese valor de pre-utilidad es valorar, no al futuro agente, sino a los agentes que intervienen en el cruce para realizar un tipo de torneo asíncrono basado en la utilidad de

cada agente que participa en el torneo, que más adelante estudiaremos con más profundidad.

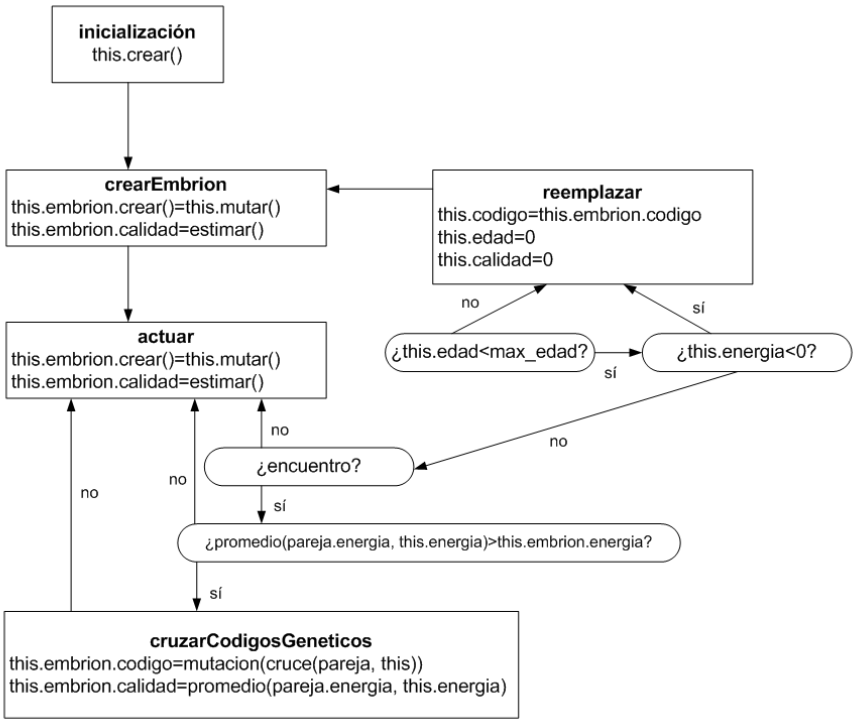


Figura 6.6: Esquema de aplicación del CeAS al problema de vigilancia.

Tras este proceso de substitución del embrión, que ocurre un número indeterminado de veces a lo largo de la vida de cada vigilante, la reproducción propiamente dicha se produce cuando un agente agota su tiempo de vida o su nivel de energía baja hasta cero. En ese momento, los parámetros evolucionables del embrión substituyen a los del vigilante. Los parámetros de estado del vigilante permanecen excepto el valor de energía (utilidad), que pasa a ser la mitad del valor de pre-utilidad. Finalmente, el tiempo de vida se reinicia a cero. Mediante este modo de generación de nuevos individuos, nos aseguramos de que el tamaño de la población no varíe. También es así en el caso de un algoritmo evolutivo

tradicional, pero mediante este procedimiento, además se realiza de una forma descentralizada y asíncrona, condiciones intrínsecas al CeAS que debemos mantener.

Por otro lado, con respecto al criterio de selección que guía la evolución de la población, cuanto mayor sea la utilidad que un vigilante posee durante la evolución del escenario, mayores probabilidades tiene de cruzarse, tanto porque es más probable que salga vencedor en el torneo embebido que se realiza mediante los embriones en cada vigilante, como porque el tiempo de vida será mayor, y por tanto, aumentará el número medio de torneos en los que participará.

Finalmente y de acuerdo a la metodología del CeAS pasamos a analizar los siguientes puntos que son relevantes en el apartado de selección reproductiva:

- **Proceso de selección:** la selección debe ser descentralizada y asociada a la tarea objetivo, tal y como corresponde al procedimiento propuesto por CeAS. En efecto, la reproducción basada en embriones posee un sistema de selección basado en la proximidad espacial entre vigilantes. Aquellos vigilantes que entren dentro del rango de visión de reproducción de otro vigilante, participaran en la selección reproductiva.
- **Asociación selección/nivel de utilidad:** esto lleva a que aquellos vigilantes con un nivel de movilidad bajo tengan menos probabilidades de participar en procesos de selección. Lo que conseguimos con esta selección es, por un lado, penalizar a aquellos agentes que se queden en posiciones fijas o que se desplacen poco e interaccionen poco con el resto de vigilantes, y por otro lado, provocamos cierto localismo en la evolución genética. Es decir, se promueve que se creen varias evoluciones paralelas en distintas zonas del escenario que estén separadas espacialmente. En los resultados veremos cómo esto permite la generación de especies en ciertas condiciones en las que las tareas poseen una separación espacial.
- **Parametrización de la selección reproductiva:** los parámetros que intervienen en el proceso de selección son los que delimitan el rango de visión reproductiva, el alcance y el ángulo de visión, y la densidad

de vigilantes en el escenario. El aumento en el área cubierta por el rango de visión y el aumento de la densidad conducen a un promedio mayor de elementos dentro del rango de selección y viceversa.

Evaluación

La evaluación de los individuos que entran en la selección reproductiva se produce de forma continua y asíncrona. De acuerdo al procedimiento expuesto de la reproducción mediante embriones, cuando se propone un nuevo individuo base se compara su nivel de energía (que corresponde a su utilidad), y si es superior a los anteriores, se genera un nuevo embrión.

- **Estudio de los criterios de evaluación/asignación de utilidad:** en este caso, la evaluación mide exactamente el valor de energía que hemos asociado a la utilidad de un individuo, con lo que la asignación es directa.

  Con respecto a la asignación de un valor de utilidad al nuevo embrión que se genera como mutación del vigilante en el momento de su activación, aunque pueda parecer un parámetro poco relevante, las primeras simulaciones nos mostraron que es realmente importante. Asignar un valor correspondiente a la mitad de la utilidad del padre en lugar del 100 % de la utilidad, tiene como finalidad prevenir que se mantengan indefinidamente códigos genéticos correspondientes a embriones que fueron incorrectamente asignados como de alta utilidad. Esto significa que, por algún motivo, de vez en cuando, las circunstancias hacen que un vigilante obtenga un alto valor de energía (y por tanto un nivel de utilidad alto) sin poseer un comportamiento realmente eficiente.

  Si se usa el 100 % de la pre-utilidad previa (incorrectamente alta), es improbable encontrar otro vigilante para la reproducción (en el procedimiento de selección de reproducción se busca un individuo para la reproducción con valores de utilidad mayores que el valor de pre-utilidad del embrión actual, de manera que cuanto más alto sea el valor de utilidad, menos frecuentemente se encontrará a una pareja) e incluso más improbable si el comportamiento ineficiente hace que

el vigilante se quede parado en alguna zona del escenario reduciendo de nuevo la probabilidad de entrar en la selección reproductiva y manteniéndose el embrión de igual manera sin cambios mientras el tiempo de vida se agota.

Tras un ciclo similar, se dan nuevas versiones mutadas del embrión con una asignación incorrecta de alta pre-utilidad implicando, por tanto, bajas posibilidades de reproducción. Este comportamiento cíclico puede reproducirse indefinidamente. Sin embargo, si se usa menos del 100 % de la pre-utilidad previa, en este caso el 50 %, se evita entrar en este bucle, eso sí, a costa de disminuir la explotación y aumentar la exploración del algoritmo resultante.

- **Estabilidad de la utilidad:** debido a que el comportamiento de un vigilante es invariable durante su tiempo de vida y a que las celdas recubren todo el escenario, las variaciones en el nivel de energía de un agente se deben a cambios en la distribución de la población sobre el recinto. Si los agentes se mueven de forma no ordenada, variando la densidad de agentes en cada zona, esto provoca cambios en la energía disponible y aparecen inestabilidades. Estas se reducen aumentando el rango de visión de celdas o aumentando el rango de visión de los vigilantes, lo cual permite evitar con más facilidad zonas más o menos pobladas. Se utiliza, también, un tiempo de madurez durante el cual no se permite a su nivel de energía descender por debajo de cero para ayudar a superar el transitorio inicial, en el cual unas condiciones adversas podrían hacer desaparecer a un agente aún si este tuviese un comportamiento eficiente, y por tanto, obtendríamos una utilidad menos precisa.

Combinación genética

Tal y como hemos explicado en el capítulo 4, se utilizará el cruce bipolar con los parámetros de ajuste por defecto. El nivel de exploración y de explotación vendrán definidos por estos parámetros que representan la mayor o menor amplitud de las funciones de probabilidad que generan los nuevos genes a partir de los genes de los individuos base. Los parámetros que son evolucionables en cada agente de este modelo serán los coeficientes del sistema de control y la longitud de la memoria de la

trayectoria. Serán, por tanto, los que se transmitan al embrión y sobre los que se basará la evolución de la población.

Reemplazo

El nivel de reemplazo de la población en este escenario, es decir, el tiempo medio en el que una población es reemplaza completamente por una nueva generación, vendrá determinado por el tiempo de simulación (tiempo de vida) medio de cada agente que a su vez vendrá determinado por el tiempo de simulación máximo permitido de cada agente y por la relación entre la energía media disponible y el nivel de consumo de cada agente. El nivel de reemplazo estará marcado por el uso de un tiempo de evolución máximo tras el cual el vigilante es substituido por su embrión, y por la relación entre la energía disponible y la energía consumida en el escenario, que provoca un nivel medio de energía más bajo, y por tanto, mayor probabilidad de agotar el nivel de energía y ser substituido por el embrión.

Con esto se da por terminada la explicación de la configuración experimental de este problema de vigilancia. A continuación pasamos a detallar su planteamiento en el entorno de simulación.

6.3. Planteamiento en el WASPED

Para llevar a cabo la implementación en el WaspBed del modelo que hemos expuesto, se han generado un conjunto de eventos y elementos a partir de la definición CeAs que se presentó anteriormente y que expondremos a continuación. El diagrama de clases de la implementación de este ejemplo se muestra en la figura 6.7.

La definición de los tipos de elemento se genera a partir de los archivos XML y se muestran en la tabla de la figura 6.8. Los parámetros descriptivos del elemento, que en este caso son los coeficientes de la red neuronal, son también los parámetros evolucionables que intervienen en el cruce y los que se almacenan en el embrión.

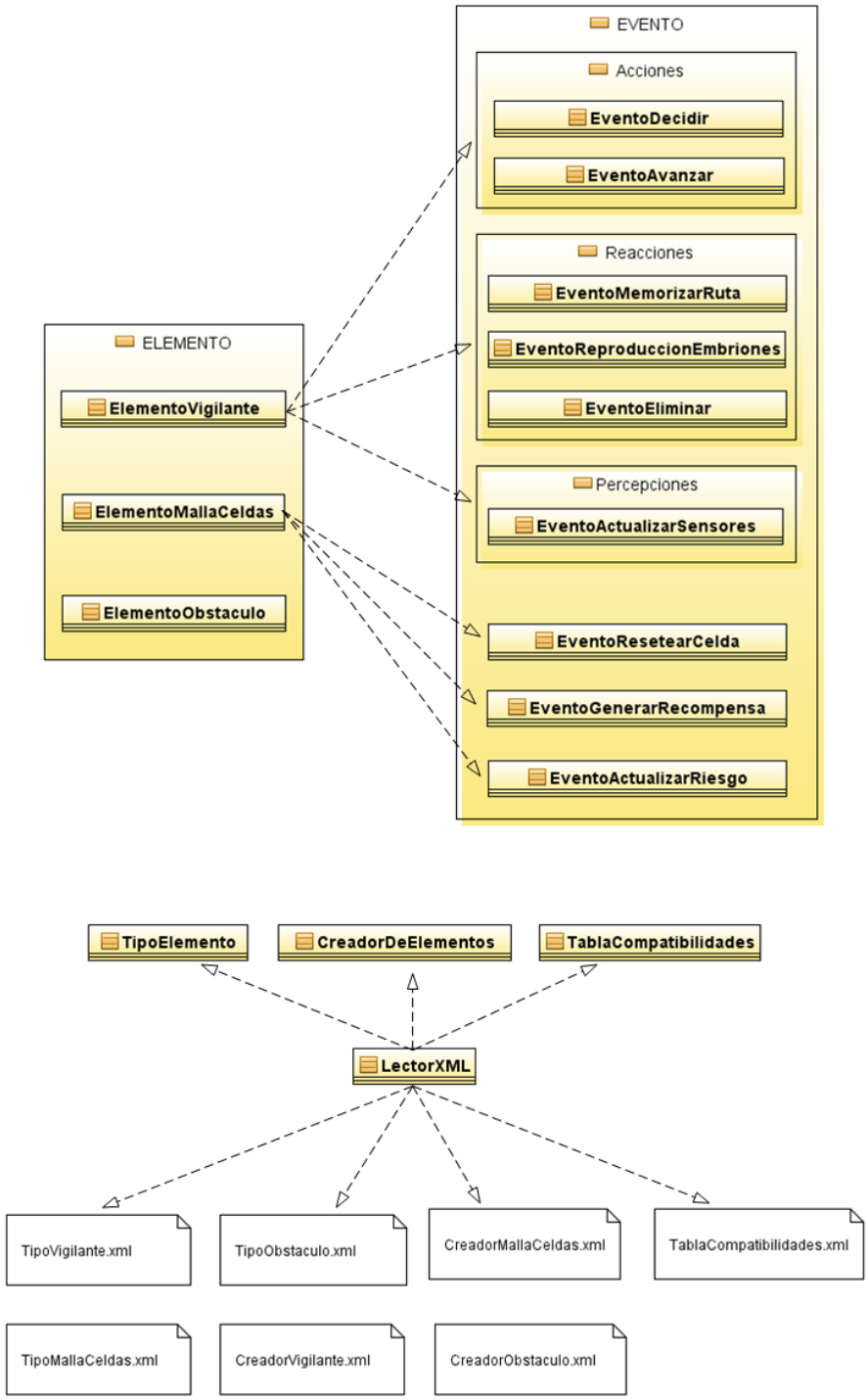


Figura 6.7: Diagrama de clases del escenario.

Elemento Vigilante	
Parámetros de tipo de elemento	**Valor**
Velocidad	5
Alcance de visión celdas	50
Ángulo de visión celdas	135º
Alcance de visión reproducción	150
Edad Madurez	100
Edad Máxima	1000
Alcance de visión vigilantes	150
Parámetros descriptivos de elemento	
Coeficientes del sistema de control	Evol [-1,1]
Longitud memoria itinerario	Evol [1,100]
Parámetros de estado	
Posición	[0,1000][0,700]
Angulo	[0,360]
Memoria Itinerario	[1,100]
Edad	[0,1000]
Energía	[0, ∞]
Coeficientes SC Embrion	[-1,1]
Longitud Memoria Embrión	[1,100]
Pre-Calidad	[0, ∞]
Detectores de Vigilantes	[0,1]
Sensor de Colisión	[-1,1]
Sensores de memoria	[-1,1]

Figura 6.8: Definición de los tipos de elemento vigilante.

Elemento Obstáculo	
Parametros descriptivos de elemento	**Valor**
Posición	Aleat[0,1000][0,700]
Tamaño	Aleat{50,100,150}

Figura 6.9: Definición de los tipos de elemento obstáculo.

Elemento MallaCeldas	
Parametros descriptivos de elemento	**Valor**
Posición	(0,0)
Numero de filas	33
Numero de columnas	48
Ancho de celda	20
Alto de celda	20
Parametros de estado	**Valor**
Valores de tiempo celdas	[0,10000]
Valores de riesgo celdas	[0,255]
Valores de recompensa celdas	[0,255]

Figura 6.10: Definición de los tipos de elemento malla de celdas.

Para los elementos obstáculos generamos la siguiente configuración de parámetros de la figura 6.9. En este caso, los únicos parámetros que tenemos son propios de cada obstáculo y son generados aleatoriamente dentro del rango de cada parámetro al principio de cada simulación. Por último, el elemento de la malla de celdas está definido tal y como se muestra en la figura 6.10.

A continuación se definen los archivos XML de creación. En el caso de los vigilantes se van a generar 20 vigilantes con los parámetros de inicialización por defecto. En el caso de los obstáculos se generarán 4 obstáculos en los que se definirán en el creador sus dimensiones y posición para que representen el perímetro del escenario, tal y como se ve en la figura 6.4. Además, hay 15 bloques repartidos aleatoriamente por el interior del escenario.

A continuación, se crean las clases Evento correspondientes a las acciones que hemos definido para los vigilantes y para la malla de celdas y se configura la tabla de compatibilidades (figura 6.11) para crear las asociaciones entre los eventos y los elementos. La tabla de la figura 6.12 muestra una descripción breve de cada uno de los eventos generados.

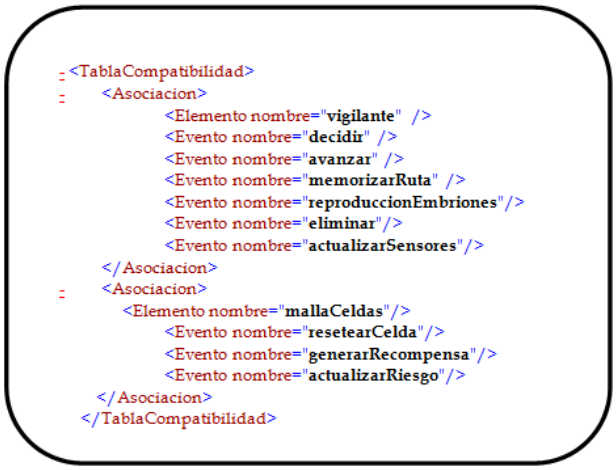


Figura 6.11: Asociaciones Elemento-Evento.

Evento	Descripción
Decidir	Ejecuta el sistema de control para generar una variación en la trayectoria.
Avanzar	Modifica la posición del agente y gestiona las colisiones con los obstáculos.
MemorizarRuta	Actualizar la memoria del itinerario.
ReproducciónEmbriones	Comprueba si existen vigilantes dentro del rango de reproducción y si éstos tienen un nivel de calidad adecuado para cruzar la información genética y actualizar los valores del embrión.
ActualizarSensores	Consulta la memoria de itinerario, comprueba si se produce una colisión y si existen agentes en el rango de visión y actualiza los sensores.
Eliminar	Comprueba si se han alcanzado las condiciones de edad o energía para la eliminación de un agente y en ese caso, este es reemplazado por su embrión y se genera un nuevo embrión a partir de la mutación del nuevo agente.
ResetearCelda	Comprueba si una celda está siendo vigilada por un agente, y si es así reinicia el contador de tiempo sin vigilancia.
GenerarRecompensa	En función del valor de tiempo sin cobertura genera un valor de recompensa por vigilar esa celda.
ActualizarRiesgo:	En función del valor de tiempo sin cobertura, se genera un valor de riesgo que servirá para medir el riesgo total en cada instante en todo el escenario.

Figura 6.12: Descripción de eventos.

6.4. Análisis de los parámetros de evolución del CeAS

Tal y como comentamos en los objetivos de este apartado, a lo largo de los distintos experimentos llevados a cabo, tanto en el estudio del ejemplo de vigilancia como en el ejemplo de rutas en grafos, hemos comprobado que una de las dificultades que entraña el uso de este tipo de evoluciones viene dada por la gran cantidad de parámetros que entran en juego en la simulación, y que afectan de una forma importante al proceso evolutivo.

En el proceso de diseño previo de este tipo de modelos, es obvio que todos y cada uno de los parámetros han de ser tenidos en cuenta y debemos asignarle un valor o un rango en el que estos pueden tomar valores. Sin embargo, no siempre es fácil determinar qué valores queremos utilizar para estos parámetros y como estos valores o determinadas combinaciones de estos afectarán a la evolución.

Utilizaremos por tanto el análisis presentado en la metodología CeAS para establecer una serie de parámetros de alto nivel, que comprendan a todos los elementos que juegan un papel relevante, y que aglutinen el resto de parámetros, para con ello permitir actuar sobre los parámetros que sí son relevantes y poder generar unos determinados cambios en la evolución, mejorar el control que tenemos sobre ella y además permitir extrapolar estos resultados a otros casos de estudio.

Se muestran sobre estos dos grupos de parámetros las asociaciones entre ellos y el efecto de la modificación de los parámetros de configuración en los parámetros de análisis. En la figura 6.13 mostramos las relaciones que hemos mencionado y si la dirección es directa o inversa (para claridad en la interpretación se han utilizado flechas azules en el caso de relaciones directas y rojas en caso de inversas).

Una vez que construimos esta tabla teniendo en cuenta los parámetros de configuración que hemos definido en el problema y los parámetros de análisis que establece el CeAS utilizaremos estas asociaciones cuando necesitemos realizar alguna variación en el comportamiento de la simulación. Por ejemplo, a la vista de la tabla del escenario de vigilancia si nos

interesase aumentar el tamaño del torneo, es decir aumentar el número de elementos que entran en la selección reproductiva deberíamos aumentar el número de agentes por unidad de superficie y por unidad de velocidad, o bien podríamos aumentar la velocidad de los agentes.

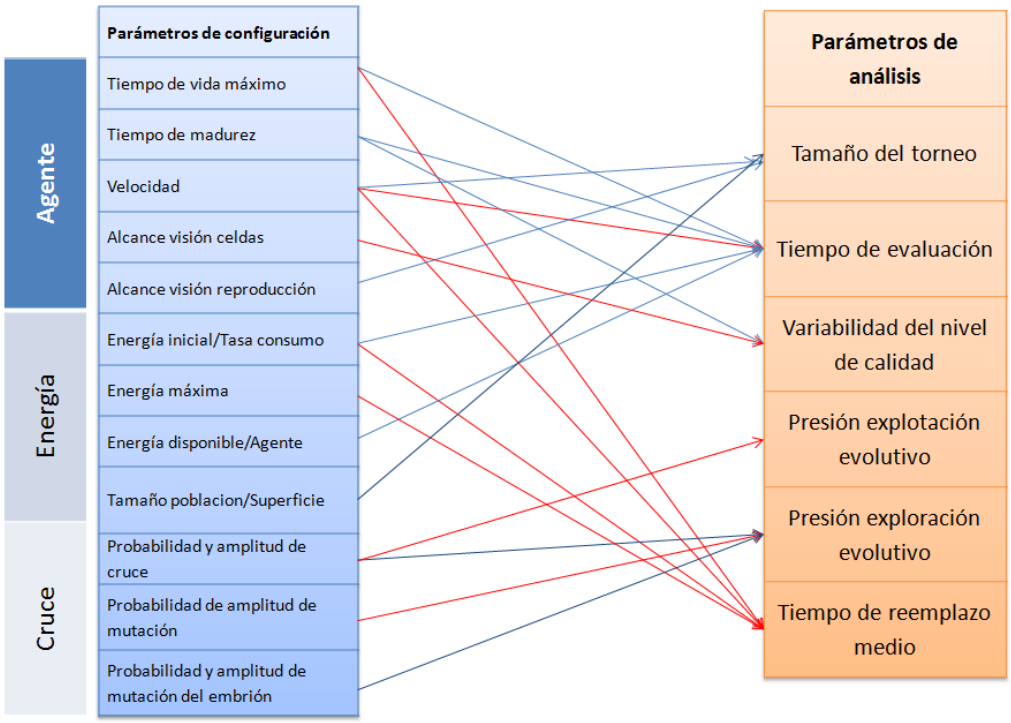


Figura 6.13: Estudio de las relaciones entre los parámetros de configuración y los parámetros de análisis.

6.5. Resultados

6.5.1. Configuración Básica

Una vez que hemos definido el entorno de simulación para este ejemplo y se ha modelado utilizando el WaspBed, estudiaremos los resultados que se obtienen permitiendo al entorno evolucionar durante 30.000 pasos de tiempo. Decidimos utilizar ese tiempo de evolución tras comprobar du-

rante varias ejecuciones que a partir de unos 20.000 pasos, el valor de riesgo se estabiliza, por lo que permitir al sistema un 50 % más de tiempo de evolución parece razonable.

Creador MallaCeldas **NIE = 1 (número inicial de elementos)**	
Parámetros de estado	**Valor inicial**
Valores de riesgo celdas	(125)
Creador Vigilantes **NIE = 20 (número inicial de elementos)**	
Parámetros de estado	**Parámetros de estado**
Edad	(0)
Energía	(1000)

Figura 6.14: Tabla creadores vigilancia 1.

Para la obtención de las curvas de evolución del riesgo se lanzan 25 ejecuciones y lo que se presenta es el valor promedio de todas las evoluciones. El resultado que obtenemos mediante la utilización de la estrategia de evolución CeAS se muestra en la figura 6.15.

Los parámetros de inicialización se presentan en la tabla de creación (ver tabla 6.14). El valor de riesgo comienza en un nivel de 200.000. Esto es así porque inicializamos los valores de riesgo de las celdas con un nivel intermedio, y lo hacemos por dos motivos: en primer lugar, el valor de riesgo representa un valor de recompensa para los agentes. Si este valor inicialmente es cero, las condiciones iniciales son realmente adversas para ellos, y por tanto, se introduce más aleatoriedad en el sistema y una explotación inicial muy fuerte, lo cual no resulta interesante ya que la evaluación sería más ruidosa y la selecciona natural inicial eliminaría mucha diversidad en la población. Por otro lado, si la curva de riesgo comenzase en un nivel cero, eso no indicaría como es la eficiencia de la población en los primeros pasos. Para hacer una representación más realista comenzamos con un nivel intermedio y así podemos estudiar las variaciones que se producen inicialmente. En efecto, en los primeros 1000 pasos de tiempo se produce una disminución de más de un 50 % del riesgo.

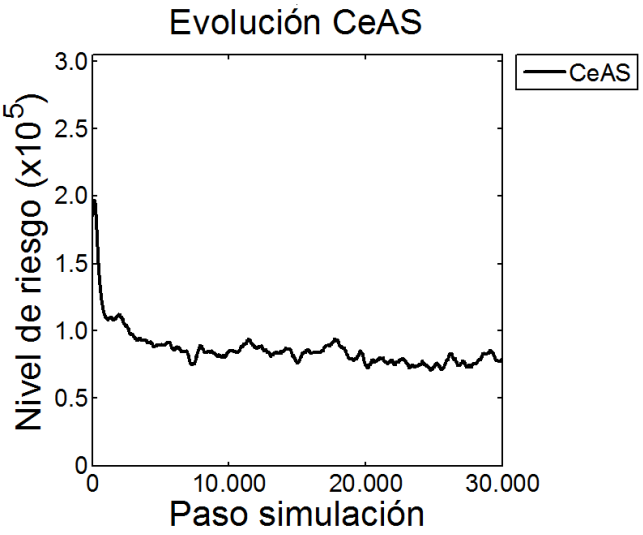
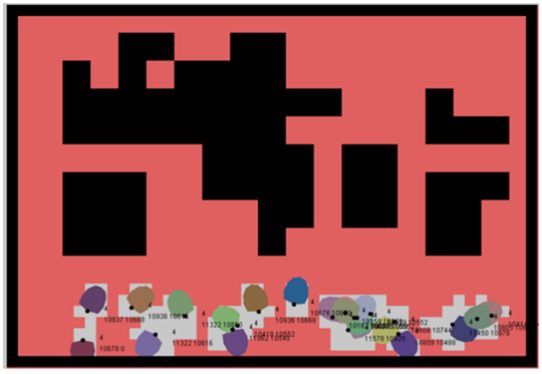


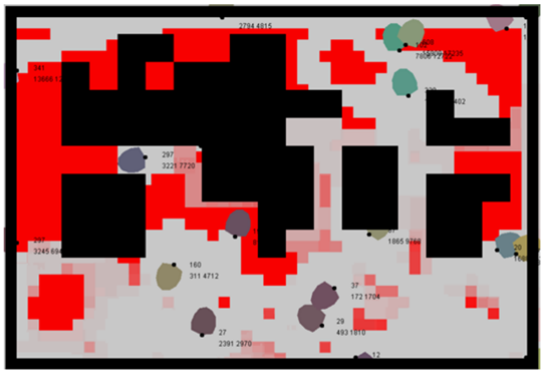
Figura 6.15: Nivel de riesgo frente a pasos de evolución en estrategia CeAS

Podemos observar que, aproximadamente, a los 6000 pasos de tiempo se alcanza un valor más o menos estable de riesgo y un valor ya muy próximo al que se obtendrá finalmente. Es en el paso de tiempo 20.000 cuando el sistema alcanza su valor de riesgo más bajo, en torno a 80.000 unidades. Debido a que el entorno está compuesto por 1.584 celdas (48 columnas y 33 filas), este nivel de riesgo corresponde a un nivel de riesgo medio para cada celda de 50 unidades lo cual, a su vez, corresponde a un tiempo medio de observación por celda de 450 pasos de tiempo. Podemos ver una representación de la evolución del comportamiento obtenido en la figura 6.5.1, el nivel de riesgo de cada celda se indica con nivel de color rojo de cada celda.

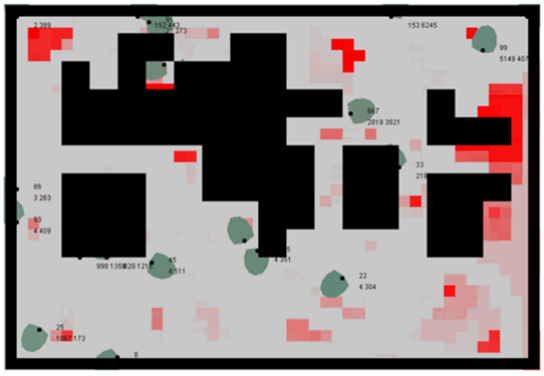
Por otra parte, es importante destacar que, aunque inicialmente el conjunto de controladores en los vigilantes era heterogéneo tal y como se refleja en la figura a través del color de los individuos, cuando la población se estabiliza, éste se ha vuelto homogéneo (sin ningún tipo de especiación) de una manera autónoma. Este hecho es importante e indica que si no hay nada en el entorno o tarea que implique una ventaja evolutiva para la especiación, las poblaciones suelen tender a controladores homogéneos.



Instante 1



Instante 2



Instante 3

Figura 6.16: Tres instantes diferentes en la simulación en las que se aprecia como disminuye el nivel de riesgo en la malla de celdas y como se reparten los agentes en el escenario de simulación.

En cuanto al tiempo medio de observación por celda, como referencia para valorar este resultado, podríamos definir el tiempo medio de observación para un comportamiento aleatorio con el objeto de generar un límite inferior con el que comparar los resultados obtenidos, como es habitual en muchos algoritmos de búsqueda. Sin embargo, para este problema, generar un comportamiento aleatorio como el obtenido en la población inicial, por ejemplo, resulta en que la mayoría de las partes del escenario quedarán sin ser vigiladas por mucho tiempo que transcurra, y por tanto, el tiempo medio entre observaciones para las celdas aumentará indefinidamente con el tiempo de simulación. Por esto, hemos optado por calcular el mínimo teórico para el tiempo entre observaciones que podría conseguirse teniendo en cuenta el área de observación de cada vigilante, la velocidad de avance, la superficie a observar y suponiendo un comportamiento cíclico ideal y una superficie que permitiese realizar ese recorrido de manera secuencial y periódica (o sea, sin obstáculos y con los vigilantes con conocimiento global y actualizado de su entorno). Teniendo en cuenta todos estos factores, y para las condiciones del experimento, el tiempo mínimo obtenido sería de 125 pasos de tiempo entre observación, esto es, unas 3.6 veces inferior al obtenido. Este valor es un mínimo teórico y el mínimo para la geometría que definen los obstáculos será superior con toda seguridad.

Por otro lado, y lo que es más importante, debemos tener en cuenta la sensorización que poseen los vigilantes. De cara a coordinar las zonas por las que se desplazan con respecto a otros agentes, son capaces de sensorizar el número de agentes en cada uno de los cuadrantes y no su posición exacta. Teniendo en cuenta el tamaño de los sectores de visión esto no resulta una medida muy precisa, y adicionalmente, los agentes no poseen ningún sensor que les indique el nivel riesgo de las celdas vecinas, con lo cual su trayectoria no sería guiada por la función de recompensa que conduce la evolución, sino por las relaciones indirectas que pueden extraer de la memoria de itinerario, la densidad de agentes circundantes y la detección de obstáculos.

La siguiente prueba dentro de la configuración base ha sido la de añadir el *detector del nivel de riesgo* con el objeto de comprobar la mejora en eficiencia que supone disponer de un indicador local en tiempo de ejecución que indique en que zona es posible obtener mejor recompensa.

Adicionalmente, se prentede estudiar la variación del nivel de eficiencia para diferentes tamaños de población (5, 10 y 20) por lo que, utilizando esta misma configuración, se han lanzado también evoluciones con diferente número de vigilantes en este escenario.

Cabe destacar que en la mayoría de las aplicaciones de vigilancia el control del nivel de riesgo de una celda determinada no se puede llevar a cabo sin un control centralizado, y es por eso que un medidor del nivel de riesgo no resulta realista. Sin embargo, utilizar el sensor de riesgo tiene un doble interés para nuestro trabajo: por un lado, las condiciones de definición de un problema de vigilancia corresponden también a problemas reales de cobertura de zonas para aplicaciones de limpieza, recogida o reparto de materiales en una red de puntos de entrega y otros, que sí permiten la obtención de esos valores de tiempo entre visitas. Por otro lado, comprobar la variación en el nivel de eficiencia mediante la variación del nivel de información que se proporciona a un agente muestra las capacidades del sistema de control y del proceso de evolución del mismo.

Los resultados obtenidos en estas simulaciones se muestran en la figura 6.17, que corresponden al valor promedio de nivel de riesgo con 10 simulaciones para cada número de agentes, como hemos dicho, 5, 10 y 20 individuos. Se puede apreciar una mejora en la eficiencia a la hora de resolver la tarea en el grupo de 20 agentes con respecto al experimento anterior.

Se observa también en la figura anterior que, como era de esperar, cuanto mayor es la población menor es el nivel de riesgo. Sin embargo, y debido a la forma de la curva de riesgo frente a tiempo sin vigilancia, la representación de esta comparativa frente al nivel de riesgo no es del todo justa. En la figura 6.18, se muestra esta misma comparación en función del tiempo promedio sin vigilancia que corresponde a cada valor de riesgo. Se puede apreciar ahora que las diferencias se han reducido. Si antes el nivel de riesgo que generaban 5 vigilantes estaba en torno a 6 veces el nivel de riesgo con 20 agentes, el tiempo medio sin cobertura es de aproximadamente el doble. Podemos ver también en esta gráfica el mínimo teórico en el tiempo medio entre observaciones que podríamos obtener con cada uno de los tamaños de población.

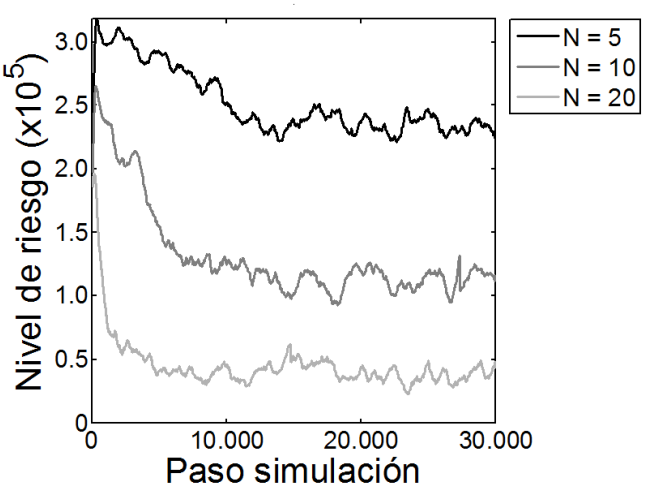


Figura 6.17: Comparación del nivel de riesgo para distintos tamaños de población

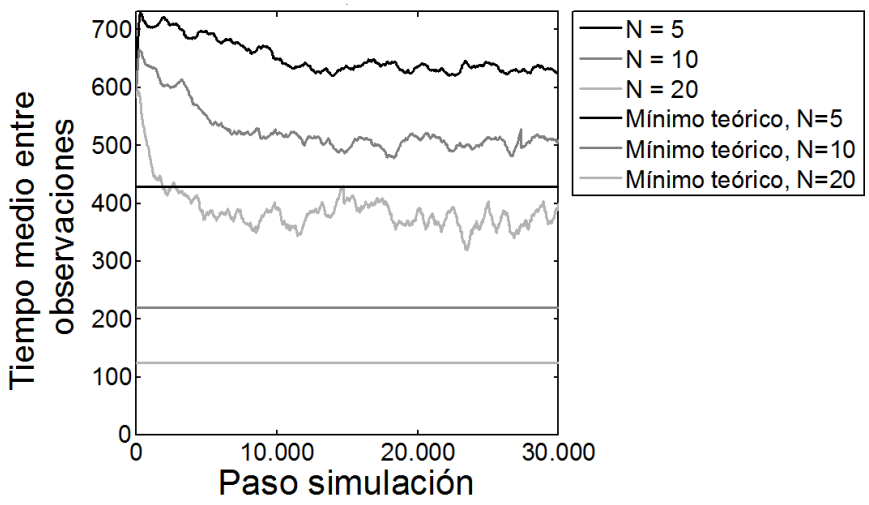


Figura 6.18: Comparación del tiempo medio entre observaciones para distintos tamaños de población

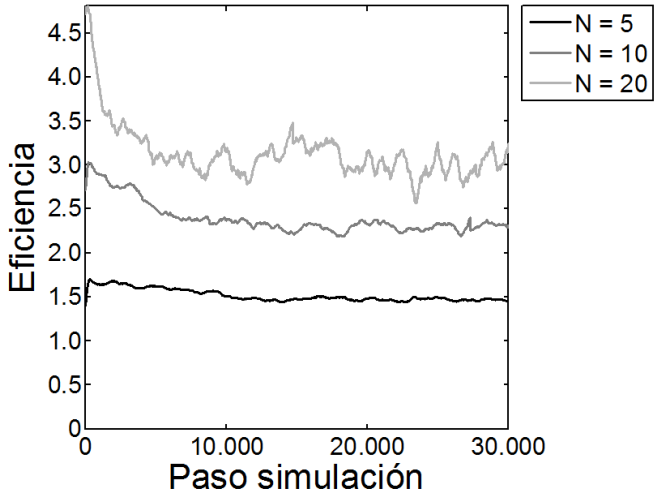


Figura 6.19: Comparación del nivel de eficiencia para distintos tamaños de población

Finalmente, y teniendo en cuenta el valor del mínimo teórico que hemos calculado para cada configuración, hemos definido una valor de *eficiencia real* para cada evolución como el ratio entre tiempo medio entre observaciones obtenido y el mínimo teórico de este tiempo medio.

$$E_f = \frac{T_{medio}}{T_{min}} \tag{6.9}$$

Los resultados se muestran en la gráfica 6.19. En este caso, podemos observar que la configuración más eficiente, tal y como esperábamos, resulta ser la de 5 agentes de vigilancia. Esto se debe a que existe un factor de interferencia más acentuado cuanto mayor es la densidad de agentes. En el caso de una configuración con baja densidad, cada agente se encarga de una zona y evita el contacto con otros agentes. En el caso de una configuración de alta densidad, no es fácil encontrar zonas no cubiertas ni repartirse organizadamente la tarea de vigilancia.

Para concluir este primer experimento, diremos que hemos comprobado la capacidad básica del CeAS para evolucionar una población de agentes de vigilancia mediante el uso de la función de utilidad privada

que hemos desarrollado inspirados en la metodología propuesta por Agogino y Tumer (2008). Se ha mostrado cómo desciende el nivel de riesgo hasta obtener una solución satisfactoria a nivel práctico. Es decir, la teoría de utilidades privada y global que se ha planteado como una de los elementos básicos del CeAS, ha funcionado correctamente en este caso.

Hemos comprobado también la capacidad del CeAS a la hora de proporcionar una solución eficiente a un problema real. A partir de este primer resultado práctico, el CeAS nos ha permitido generar una población de agentes que, sin el uso de la sensorización del nivel de utilidad, obtienen un nivel de exploración cercano al obtenido cuando utilizamos el sensor de riesgo. Este resultado profundiza en la capacidad del CeAS para analizar la respuesta de un sistema complejo a cambios en sus parámetros de diseño.

En este sentido, también hemos podido estudiar el efecto de la variación del número de agentes en el nivel de riesgo y en la eficiencia de exploración. Como hemos visto, para la comparación en el nivel de riesgo, al utilizar 20 agentes en lugar de 5, es decir, al multiplicar por 4 el número de agentes, obtenemos un nivel de riesgo 6 veces inferior, lo que nos lleva a concluir que, si el nivel de riesgo es proporcional a un coste medio asociado a intrusiones, retrasos, etcétera, resulta rentable invertir en aumentar el número de vigilantes. Por otro lado, si lo que estamos midiendo es la eficiencia con respecto al tiempo medio entre visitas sin tener en cuenta la curva de riesgo frente a tiempo, comprobamos que disminuir el número de agentes resulta en una configuración más eficiente[1]. Este tipo de análisis son, como podemos intuir, altamente interesantes en problemas reales de ingeniería.

6.5.2. Comparación con otras aproximaciones

A continuación se muestra un experimento comparativo que se ha realizado sobre este ejemplo de vigilancia para comprobar la eficiencia en el

[1]Debemos tener en cuenta que la función de utilidad que guía la evolución premia la minimización del nivel de riesgo y no del tiempo medio entre observaciones, y que aunque existe una relación directa entre estos parámetros a lo largo de toda la curva tiempo vs. riesgo, no son proporcionales, y por tanto, la evolución podría diferir si el objetivo fuese minimizar el tiempo entre observaciones.

uso del evolutivo CeAS frente a otras técnicas, que en este caso serán, por un lado, un algoritmo genético, en el cual el código genético de cada individuo de la población contendrá los parámetros evolucionables de los vigilantes y será el mismo para todo el grupo de vigilantes. Es decir, evolucionaremos un comportamiento formado por un grupo de agentes intrínsicamente homogéneos. Por otro lado, se utilizará un algoritmo híbrido que posee la estructura de un algoritmo genético, con la diferencia de que la evaluación de los individuos se realiza generando un agente con el código genético de cada individuo, con lo que se genera un grupo de agentes que se introducen en el escenario y se dejan evolucionar, es decir, para la evaluación de la población se utiliza la simulación de estos individuos interactuando unos con los otros en el modelo propuesto, y tras un cierto tiempo previamente fijado, se comprueba el nivel de utilidad y se genera otra nueva población que, al igual que ocurre en un genético estándar, sustituye a la primera.

A continuación exponemos las características más relevantes en cuanto a la evolución de cada una de estas estrategias y el modo en el que se han llevado a cabo las simulaciones.

En cuanto al **algoritmo genético**, en primer lugar se generan aleatoriamente 20 códigos genéticos y cada uno de ellos se evalúa creando un población homogénea de 20 vigilantes con ese código genético que se dejan interactuar en el entorno. Esta simulación es equivalente a la que se realiza en la evolución guiada por el CeAS a excepción de que se deshabilitan los eventos de reproducción o eliminación.

La utilidad asociada al algoritmo es la utilidad global del escenario durante la simulación, con la ventaja de que no nos vemos obligados a diseñar funciones de utilidad individual que guíen a la población hacia la utilidad global.

Se utiliza, al igual que en las otras dos estrategias, un cruce por torneo. Para hacer equitativa la comparación, en primer lugar se extrajo un valor promedio de tamaño de torneo de la evolución CeAS. Simplemente se comprobó el número de comparaciones de utilidad realizadas por los agentes a lo largo de las 30.000 evoluciones y el número de eliminaciones de agentes. El cociente entre estos dos valores indica el tamaño medio de

comparaciones por cada nuevo individuo generado, lo que es equivalente al tamaño medio de torneo. Este valor resultó ser aproximadamente 8. Sin embargo, las primeras evoluciones nos mostraron que, como era de esperar, un tamaño de torneo de 8 en una población de 20 elementos implica un nivel de explotación extremadamente alto y más aún para un espacio de búsqueda de 28 dimensiones. Por lo tanto, utilizamos el tamaño de torneo más bajo posible, y este es 2.

De nuevo, con el objeto de realizar una comparación equitativa, cada una de las evoluciones debería utilizar 30.000 pasos de evaluación. Sin embargo, para evaluar a cada uno de los individuos y obtener un valor de utilidad estable, hemos comprobado que se requieren al menos 800 pasos de tiempo, lo que nos llevaría a 16.000 pasos de tiempo para evaluar a los 20 individuos de cada generación, y por lo tanto, esto permitiría generar simplemente 2 generaciones del algoritmo genético. Es por esto que se optó por una estrategia más eficiente de cara a aprovechar el número de pasos de evaluación. Esta estrategia consiste en utilizar un tiempo de evaluación incremental. La fluctuación en la evaluación de las primeras generaciones no es perjudicial, pues simplemente aumenta el nivel de exploración inicial, y a medida que pasan las generaciones, el tiempo de evaluación aumenta hasta obtener valores razonablemente fiables. Por tanto, se utilizaron tiempos de evaluación para cada uno de los individuos de cada generación de 100, 200, 400 y finalmente 800 pasos de tiempo.

Finalmente, la **evolución híbrida** está basada en un tipo de evolución utilizada en algunos trabajos de coordinación (Groβ y Dorigo, 2008) en la cual se inserta un evolutivo en la población de agentes que están interaccionando en el escenario. Por lo tanto, se diferencia de la evolución CeAS en que la reproducción en este caso, se fuerza cada cierto número de pasos de evaluación y además se produce mediante una selección centralizada y no local y por lo tanto no es situada.

La selección de los agentes para el cruce se realiza también mediante el uso de un torneo. Al igual que en el caso del algoritmo genético, el tamaño se ha fijado a 2 después de haber probado con 8 -el tamaño equivalente al del CeAS- y haber obtenido unos resultados realmente pobres. Esta selección se realiza de forma centralizada, es decir, se seleccionan al azar de entre toda la población 2 individuos para generar cada uno de

los dos torneos en los que se seleccionaran los agentes que se cruzarán.

En cierto sentido, un primer análisis nos sugiere que, como ventaja frente a la selección natural utilizada en el CeAS, en la evolución híbrida los agentes no tienen por que encontrarse en el espacio para reproducirse. Su único objetivo ahora es la mejora de la utilidad, luego un controlador centralizado se encargará de seleccionarlos.

Por último, en cuanto al tiempo de evaluación para los agentes que se encuentran en el entorno, se ha elegido el tiempo equivalente al que se utiliza en el caso de la evolución CeAS. Esto se calcula extrayendo el número de reproducciones que se producen de media en las simulaciones a lo largo de los 30.000 pasos. Este análisis nos indica que, en promedio, se crea un nuevo elemento cada 150 pasos de tiempo, y esto es lo que se utilizará también para el caso de la evolución híbrida. Un torneo de tamaño 2, generará una nueva población de 20 individuos cada 150 pasos de tiempo. Como ya comentamos anteriormente, para realizar la comparativa se ejecutan 25 simulaciones de cada tipo estrategia de evolución y se presenta la media de estas.

Adicionalmente a la evolución estándar en la que estamos usando el mínimo nivel de riesgo para el mejor individuo de cada generación, realizamos una medida de rendimiento más honesta. En paralelo, se deja que se ejecute el mejor individuo de cada generación durante 4000 pasos y se extrae el promedio de los últimos 3500 pasos para obtener un nivel de riesgo real, a lo que denominamos ‘Evolución estándar completa’.

La desviación estándar de las evoluciones es de aproximadamente 20.000 unidades de riesgo para la evolución CeAS y para el algoritmo genético, sin embargo, la desviación de la evolución híbrida es mucho mayor lo cual empeora aún más si cabe los resultados obtenidos.

En la gráfica 6.20 hemos mostrado los valores del nivel de riesgo global conseguido con cada una de las aproximaciones, en el caso del CeAS se muestra el valor del riesgo global durante la evolución, en el caso del genético tradicional se muestra el valor obtenido con el mejor individuo tras cada generación y en la aproximación híbrida se muestra el valor global obtenido en el escenario a medida que se van sucediendo las distintas

generaciones. A la vista de las curvas de resultados, podemos concluir que, tras un número de pasos elevados, las diferencias entre el uso de un algoritmo genético y el uso del CeAS no son notables teniendo en cuenta el valor de desviación obtenida. Sí es apreciable la diferencia en la velocidad de convergencia, favorable al CeAS en todos los casos. En lo que respecta a la aproximación híbrida, los resultados muestran que, para este ejemplo, esta estrategia es claramente menos eficiente, incluso podríamos decir que inapropiada.

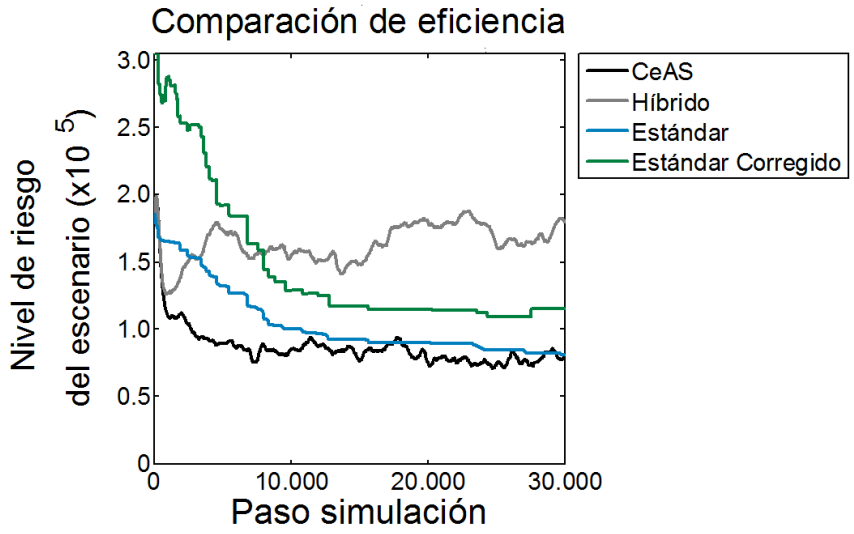


Figura 6.20: Comparación entre varias técnicas evolutivas. Nivel de riesgo frente a pasos de evolución.

Adicionalmente, se ha intentado una nueva variación del acercamiento híbrido, pues sus malos resultados nos resultaron sorprendentes. Uno de los motivos que podrían haber llevado a este nivel de error es que estamos usando sólo 140 pasos de tiempo para evaluar los individuos antes de realizar la selección de torneo. En consecuencia, se ha incrementado en primer lugar a 400 pasos y posteriormente a 750, y sin embargo, no obtuvimos ninguna mejora de manera que finalmente concluimos que la aproximación híbrida no es realmente útil en este problema.

Como conclusión de la comparativa llevada a cabo en este apartado, diremos que los resultados obtenidos han sido plenamente satisfactorios a la hora de comparar el CeAS con otras técnicas de computación evolutiva. Más aún teniendo en cuenta que la comparativa se realiza en unas condiciones muy específicas en las que el CeAS no permite aprovechar todas sus ventajas. La evolución converge claramente más rápido que otras aproximaciones a un valor de riesgo bajo. La única aproximación que se aproxima a los resultados mostrados por el CeAS es la del algoritmo genético tradicional teniendo en cuenta, como hemos comentado ya, que el criterio de evaluación es muy poco preciso y tiende a dar valores superiores a los reales.

6.5.3. Evolución en entornos heterogéneos: formación de especies

Una vez estudiada la capacidad del CeAS para obtener soluciones con las condiciones de diseño propuestas inicialmente, hemos decidido comprobar sus cualidades a la hora de realizar evoluciones que generen automáticamente especies dentro de una misma población de individuos. Con la formación automática de especies se busca una resolución más eficiente de ciertos problemas, y además se incide más aún en una propiedad diferenciadora del CeAS respecto a otros algoritmos poblacionales, en el que la aparición de especies está totalmente causada por el diseñador. En este apartado, además, se probará la eficacia del cruce bipolar que hemos definido en cuanto al mantenimiento de dos (o más) especies diferenciadas.

Para provocar que se formen especies dentro de una evolución, es indispensable que existan tareas diferenciadas y que estas tareas sean disjuntas a efectos de las capacidades de los agentes de vigilancia. Es decir, es necesario que no sea posible obtener un agente que sea capaz de llevar a cabo las tareas de forma eficaz, y que por tanto, la especialización aporte una gran ventaja evolutiva.

Con el objeto estudiar la formación automática de especies, inicialmente, nos centraremos en el planteamiento de un escenario con dos tareas

diferenciadas, y analizaremos en qué condiciones aparecen dos especies. Para ello, se han introducido los siguientes cambios en el escenario.

- En primer lugar se ha añadido a las celdas un nuevo parámetro que permite diferenciarlas y ahora los agentes pueden variar sus reacciones en función de ese parámetro. En términos de la estructura del WaspBed, antes las celdas poseían solo parámetros descriptivos de tipo y parámetros de estado para diferenciar el nivel de riesgo de cada una, ahora hemos añadido un parámetro descriptivo de elemento. De este modo se permite una diferenciación entre zonas del entorno.

  Esta nueva configuración se inspira en el establecimiento de varias zonas dentro de un entorno de vigilancia real en las cuales es más o menos apremiante la vigilancia, es decir, el nivel de riesgo asociado a un determinado tiempo sin observación es diferente entre unas zonas y otras. Un caso claro de este tipo de tareas en entornos reales viene dado por tareas de monitorización estructural en entornos tales como barcos, donde el riesgo en planchas y en uniones con cuadernas es distinto y la detección de una u otra zona es bastante simple para un robot real.

- Adicionalmente, se ha dotado a los vigilantes de un sensor capaz de detectar la transición entre zonas: el sensor indica si se pasa de una zona de mayor riesgo a una de menor riesgo y viceversa. En su aplicación real, los contornos de las diferentes zonas estarían también identificados de algún modo, de manera que el vigilante pudiese percibir el cambio de zona.

Para la primera configuración experimental se han definido dos zonas en el entorno. Una de ellas no posee riesgo asociado, es decir, la vigilancia de esa zona no proporciona ninguna recompensa evolutiva (ver figura 6.21.a). Y la otra zona sí posee un valor de riesgo elevado. De ahora en adelante, para facilitar la explicación de las siguientes configuraciones, denominaremos temperatura de la zona a este parámetro diferenciador.

Figura 6.21: Escenario de vigilancia con dos zonas. En el estado inicial se aprecia la geometría de la zona de riesgo. En el estado final se muestra la disposición de los agentes sobre la zona de riesgo tras el proceso de evolución de la población.

Por lo tanto, este nuevo escenario tendrá una zona neutra y otra zona caliente. La zona caliente ocupará el 15 % del escenario pero concentrará un valor de riesgo equivalente al de todo el escenario en las configuraciones anteriores. El nivel de riesgo inicial será igual a los observados anteriormente, y la cantidad de energía disponible es la misma también. Esto permite mantener, tal y como estudiamos en el apartado 6.4, el tiempo medio de evaluación y el reemplazo en la evolución de la población.

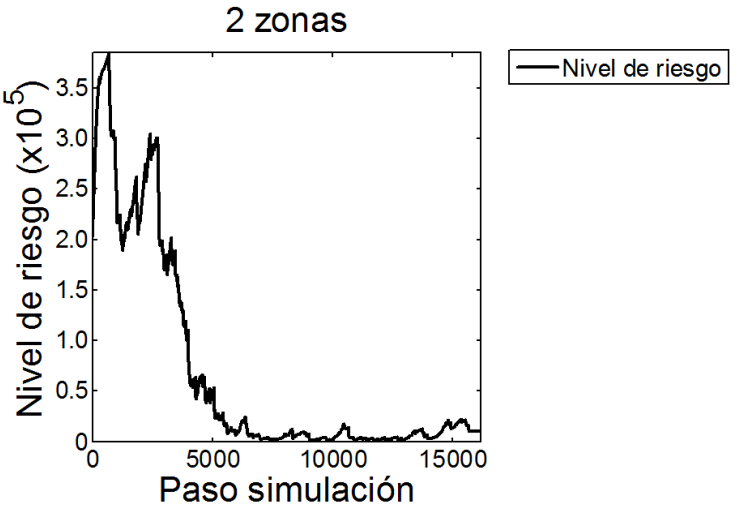


Figura 6.22: Evolución del nivel de riesgo para el escenario con dos zonas.

Esta primera configuración servirá, básicamente, para comprobar la capacidad del CeAS de obtener soluciones adecuadas teniendo en cuenta esta nueva sensorización de los agentes. En la figura 6.21.a se muestra esta nueva configuración, y en la figura 6.21.b se muestra el comportamiento final de los agentes.

Se puede ver la evolución del nivel de riesgo global en la figura 6.22, donde apreciamos cómo se obtiene un nivel de riesgo realmente bajo una vez que el sistema se estabiliza. Esto es debido a que la densidad de los agentes en la zona a vigilar es ahora muy alta, y por tanto, la superficie cubierta por ellos es superior. Si nos detenemos a pensarlo, ahora la

tarea es más sencilla, y en efecto, el valor de riesgo es aproximadamente el valor de riesgo mínimo teórico.

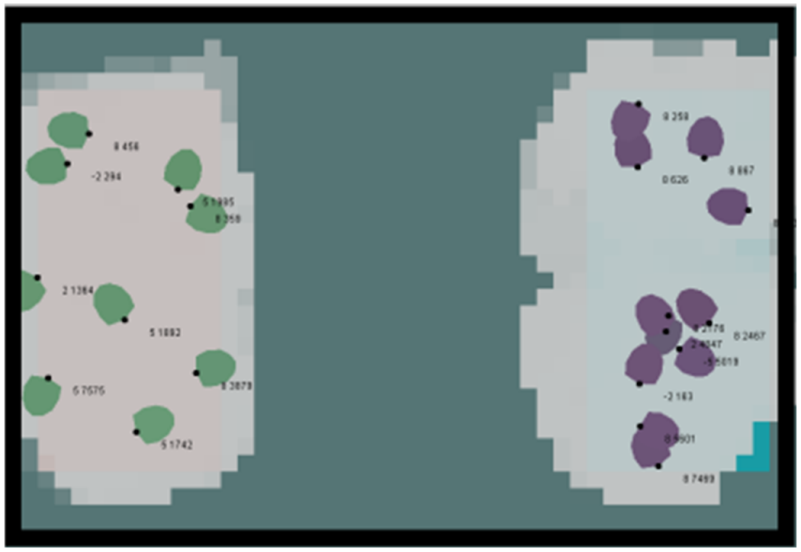


Figura 6.23: Escenario de vigilancia con tres zonas. Estado final.

Una vez que hemos comprobado el funcionamiento del sistema con los nuevos cambios de diseño del escenario, pasaremos a generar otro escenario en el que ya sí establecemos dos tareas diferentes. Como hemos dicho, la configuración establecida tiene dos zonas de riesgo con un valor de temperatura distinto y otra zona neutral. A efectos de los agentes, el cambio de una zona neutral a la zona fría es detectado por los sensores del mismo modo que el paso de la zona caliente a la zona neutral. Igualmente, entrar en una zona caliente es equivalente a salir de la zona fría. En consecuencia, el controlador que genera un comportamiento de entrada en una zona, se escaparía de la otra, y de este modo, los posibles comportamientos que se pueden generar para explorar una zona u otra serán contrarios. Tras dejar evolucionar el sistema durante 30.000 pasos, obtenemos la configuración mostrada en la figura 6.23. Recordemos que el color de los vigilantes representa una combinación concreta de genes, y por tanto, los individuos del mismo color son miembros de la misma especie. En este caso, se han formado dos especies, cada una especializada

en la vigilancia de una de las zonas. Esta especialización ha venido determinada por la capacidad de los individuos de detectar las transiciones que les permiten delimitar su zona. Se puede apreciar en la figura también que el nivel de riesgo es realmente bajo, en este caso el nivel de riesgo viene dado en cada zona por el nivel de color azul o naranja, y a medida que disminuye el riesgo el color es acerca al blanco.

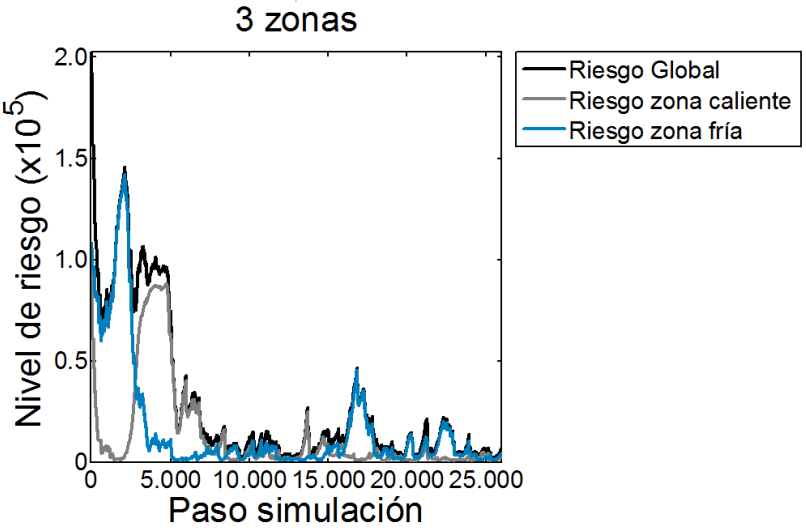


Figura 6.24: Evolución del nivel de riesgo para tres zonas.

La variación del nivel de riesgo para esta evolución y para cada una de las dos zonas se muestra en la figura 6.24, en la que se pueden apreciar las fluctuaciones iniciales de la población hacia agentes que son capaces de resolver una u otra tarea y por tanto la disminución brusca del nivel de riesgo en una u otra zona hasta que en el paso de evolución 600 aproximadamente la población se divide, se obtienen dos especies con dos tareas diferenciadas y se obtiene un nivel de riesgo bajo para ambas zonas.

La obtención de dos especies diferenciadas a la que hemos llegado se debe, por tanto, a la utilización de un cruce que evita la convergencia de la población hacia una sola solución, y también a la separación espacial

de las dos zonas en las que se llevan a cabo las tareas diferenciadas. Para comprobar qué ocurre en caso de que aumente el contacto entre miembros de las dos especies, es decir, al eliminar la separación espacial, cambiamos la configuración para una en la que la zona caliente y la zona fría están en contacto.

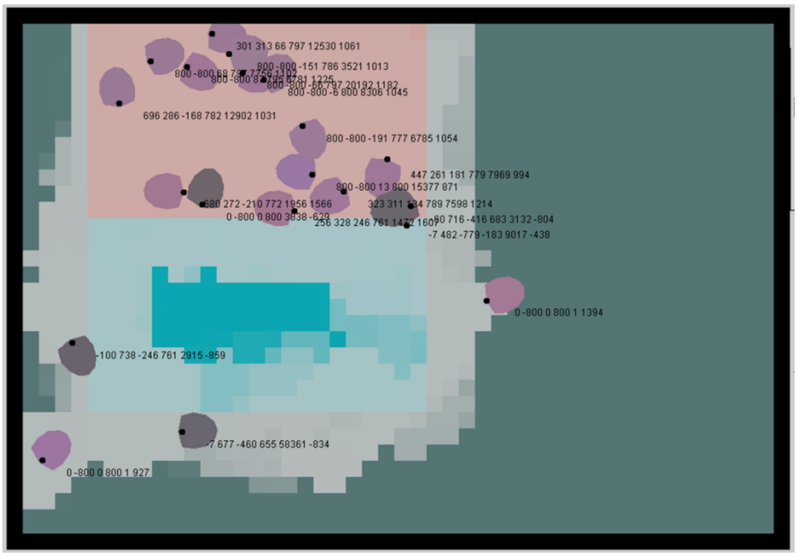


Figura 6.25: Escenario con tres zonas contiguas: Evolución sin coeficiente de afinidad. Estado final.

Tras las primeras evoluciones obtenemos un resultado como el que se muestra en la figura 6.25. La tendencia es a que exista una especie dominante, y de vez en cuando, surgen mutaciones pero que se extinguen ante la fuerte presión de la otra especie. Comprobamos también que las tareas no están en este caso tan fuertemente diferenciadas, ya que un comportamiento que tienda a acercarse a una zona y con una componente errática alta funciona bastante bien, pues cubre la zona fría con cierta frecuencia. En conclusión si las tareas no están claramente diferenciadas y no existe una separación espacial que facilite la especiación, es difícil obtener una división de la población.

Con el objeto de hacer más fuerte la presión hacia la especiación en este último escenario, se ha introducido un nuevo elemento que inter-

vendrá en el proceso de selección de los individuos base. Este será el parámetro de afinidad: se incluye dentro de la evaluación de los individuos elegidos en la selección reproductiva un factor que aumenta la calidad de un individuo de forma proporcional al nivel de afinidad con el otro individuo base.

La forma de evaluar el nivel de afinidad entre dos individuos se realizará mediante la comparación de sus genotipos. Debido a que, para suponer un comportamiento similar entre dos vigilantes la variaciones entre genes deberían ser pequeñas y a que, una vez superado un valor de variación en un gen, el alcance de las modificaciones en el comportamiento de la red neuronal ya no guardan relación con la amplitud de esta variación, se utilizará una comparación que dé mayor importancia al incremento de las variaciones pequeñas. Para ello se va a utilizar la siguiente expresión que representa a la afinidad entre dos individuos A y B cuyos códigos genéticos serán $G^A = (g_0^a, g_1^a, g_2^a, \ldots, g_M^a)$ y $G^B = (g_0^b, g_1^b, g_2^b, \ldots, g_M^b)$:

$$A_f = \frac{\sum_{i=0}^{M} \sqrt{g_i^a - g_i^b}}{M} \tag{6.10}$$

Tras implementar este coeficiente de afinidad, el comportamiento mejora sensiblemente como se puede apreciar en la siguiente captura del escenario (ver figura 6.26). Existe una diferencia genotípica apreciable y además los agentes realizan cada uno la tarea correspondiente de forma visiblemente eficiente.

La evolución de los riesgos en las distintas zonas se muestra en la gráfica 6.27, en la que se presenta el promedio de 10 evoluciones. Se ha hecho este promedio debido a que, aún utilizando el parámetro de afinidad, el comportamiento obtenido no es tan estable como en el caso de existir una sola tarea o dos tareas separadas espacialmente. Se muestra también una comparativa para comprobar los efectos en la evolución del uso del criterio de afinidad y del cruce bipolar. A la vista de la gráfica, comprobamos que a pesar de que todas las configuraciones evolucionan con respecto a la situación, la mejor configuración es aquella que combina ambas estrategias y que por tanto introduce una presión evolutiva hacia la especiación más fuerte.

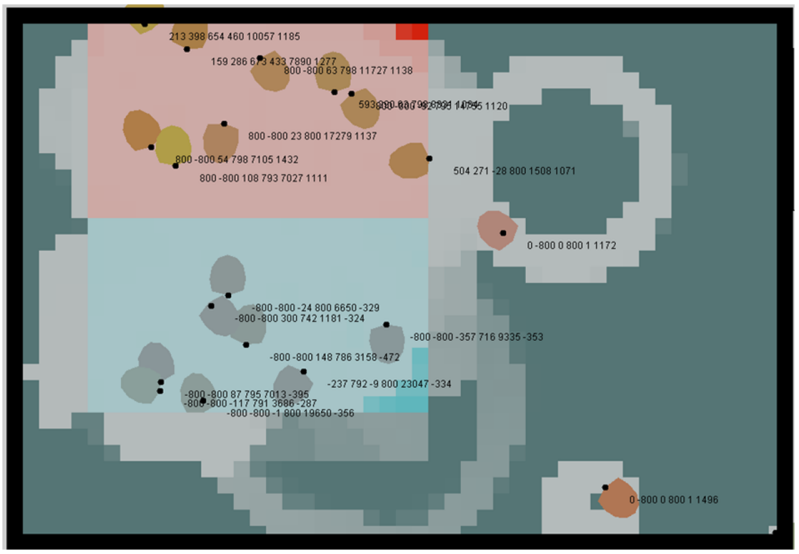


Figura 6.26: Escenario con tres zonas contiguas: evolución con coeficiente de afinidad. Estado final.

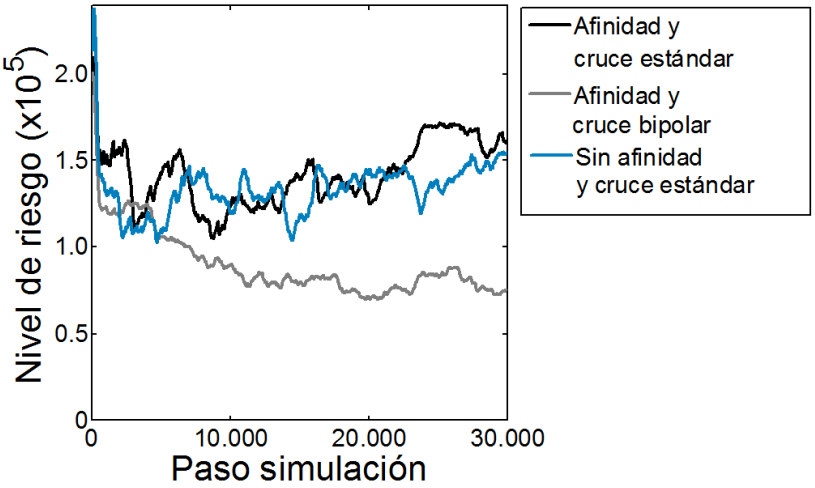


Figura 6.27: Comparativa parámetro afinidad vs. cruce bipolar.

Por último, se activa el sensor de riesgo. Es decir, los agentes detectan la recompensa que va a generar cubrir una u otra zona. El valor de riesgo promedio obtenido finalmente es satisfactorio y apreciablemente más bajo que en el caso anterior (ver figura 6.28). Tal y como vimos en los resultados del primer apartado, el uso de del sensor de riesgo mejora claramente la realización de la tarea e incluso tal y como hemos observado en las simulaciones provoca comportamientos más estables.

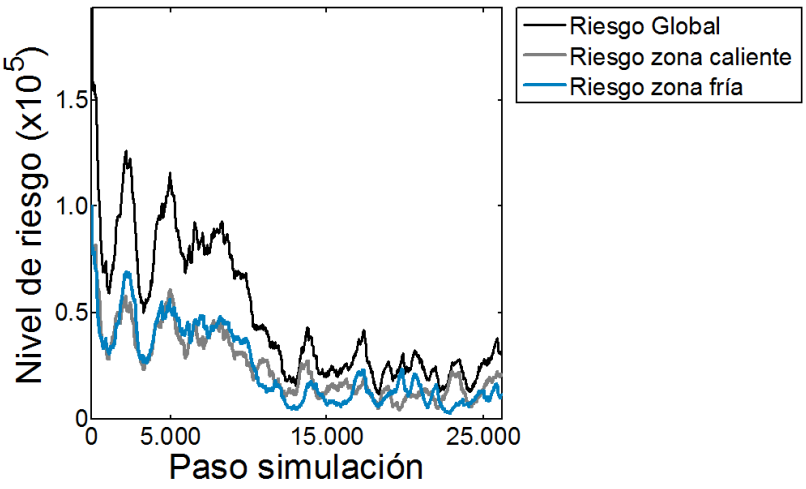


Figura 6.28: Evolución del nivel de riesgo para tres zonas contiguas con detector de riesgo.

Como conclusión a este tercer experimento, el CeAS ha mostrado una capacidad muy interesante, y es que es capaz de obtener de forma automática una especialización en los agentes, si la definición de la tarea así lo requiere. Este resultado es muy relevante a nivel práctico, ya que permite obtener sistemas heterogéneos si esto supone una ventaja a la hora de resolver la tarea.

A nivel de mejora del CeAS, en este experimento se mostrado el correcto funcionamiento del operador de cruce bipolar y el parámetro de afinidad a la hora de formar especies automáticamente.

6.6. Conclusiones

En este segundo caso de estudio del CeAS en la resolución de un problema dinámico descentralizado, se ha generado un escenario que representa un problema de exploración, en concreto, un problema de vigilancia autónoma y distribuida de un entorno. Se ha estudiado la obtención de este comportamiento para diferentes configuraciones en cuanto al número de agentes, el nivel de sensorización y el tipo de tarea y entorno en el que se realiza la tarea, todo ello con el objetivo de analizar las capacidades del CeAS a la hora de proporcionar una solución eficiente a un problema real con independencia de los parámetros de diseño prefijados.

Se ha realizado también una comparativa entre la evolución CeAS y otras aproximaciones para el caso de una población homogénea en un entorno que requiere de la realización de una sola tarea, obteniéndose resultados muy satisfactorios. En consecuencia, se han comprobado las ventajas del uso de la técnica propuesta aún en una configuración que no explota las características diferenciadoras del CeAS respecto a las aproximaciones tradicionales.

Finalmente, se ha hecho un estudio de la capacidad para generar especies de la evolución CeAS, en el que se ha comprobado la influencia de la separación espacial de las tareas que se proponen, el nivel de diferenciación de las tareas y el uso de el cruce bipolar y del parámetro de afinidad.

Capítulo 7

Estudio del CeAS en un problema de abastecimiento

7.1. Introducción y Objetivos

Como tercer ejemplo para el estudio del CeAS hemos utilizado otro de los problemas dinámicos descentralizados típicos en ingeniería, el problema de abastecimiento. Este problema consiste, en líneas generales, en obtener un conjunto de rutas que permitan que un grupo de vehículos abastezcan a un grupo de clientes.

De forma más concreta, dentro de la tipología de problemas de abastecimiento, el caso de estudio que nos planteamos consiste en un *problema de fletes de barcos*, cuyo objetivo de diseño es, por un lado, obtener estrategias de asignación de rutas óptimas para cada uno de los barcos que componen la flota con el objeto de realizar el transporte de materiales desde unos puntos de suministro hasta unos puntos de demanda de la forma más eficiente posible. Por otro, en este caso pretendemos llegar también a la composición ideal de la flota en cuanto al número y tipo de barcos que la componen.

Concretando, el modelo con el que representamos en el marco del CeAS al problema de fletes va a estar compuesto por un grupo heterogéneo de barcos de transporte de mercancías. Cada uno de ellos posee una serie de características de diseño y una estrategia de selección de

destinos propias. Su objetivo es el de satisfacer las peticiones de los distintos puntos de demanda que existen en el entorno mediante el transporte de contenedores de carga desde una conjunto de puntos de suministro.

El problema de fletes que proponemos podríamos clasificarlo, en cierta medida, como un problema VRP con ventanas de tiempo y capacidad limitada aplicado una flota de barcos, al cual se le añaden variables estocásticas. Adicionalmente, este problema, que se basa en una configuración de flota fija, se modifica de forma que se requiere, en paralelo, la optimización de la composición de la flota que va a realizar los repartos. Con respecto a las aproximaciones que pudieran haberse utilizado con anterioridad para resolver un problema como el que estamos tratando, no hemos encontrado ninguna similar.

Hemos seleccionado este problema porque posee las características prototípicas del marco de aplicación del CeAS (dinamismo, optimización combinatoria, tarea distribuida), y además, añade nuevas problemáticas no contempladas anteriormente (modelo con parámetros estocásticos y concurrencia). Respecto a lo primero, el gran número de combinaciones en las que podríamos asignar a cada una de las rutas de transporte uno o más barcos, junto con las distintas configuraciones de una flota en función de los tipos de barco que posea, hacen de este problema un ejemplo claro de optimización combinatoria. De hecho, posee similitudes claras tanto con el problema de asignación, pues se trata de asignar barcos a rutas para minimizar costes, como con el de transporte, en el que se buscan rutas entre puntos de suministro y demanda, dos de los problemas fundamentales de ese campo como ya comentamos en el apartado 3.1.

La generación de las solicitudes de demanda se ha modelado a través de una distribución de probabilidad. El tiempo que transcurre entre las peticiones de demanda es una variable aleatoria, y la cantidad de material asociada a cada petición, es también una variable aleatoria. Esta generación estocástica de demandas es la que genera todo el comportamiento que se produce en el sistema, por lo que, aunque las variables sujetas a aleatoriedad sean sólo dos de los parámetros de los puntos de demanda, estas son de gran relevancia. En efecto, el nivel de demandas afecta tanto a las rutas que se realizan como al tipo de barcos que son más adecuados para esas rutas. La variabilidad en las demandas afecta también de cara

a ser más o menos conservadores en cuanto a explotar la rentabilidad de los barcos asumiendo cierto nivel de riesgo. Concluimos, por tanto, que este problema es un ejemplo claro también de problema estocástico, dentro de los cuales, por cierto, uno de los referentes son los problemas de demanda probabilística.

Consideramos también, que el hecho de que en este problema las estrategias a seguir por los barcos estén supeditadas, por un lado, al coste que genera el transporte de cada uno de los barcos en función de sus características de diseño (velocidad, capacidad, etcétera), por otro al mercado que se genere en el transporte de mercancías (aumento o disminución de la demanda, de los precios de transporte, etcétera), y finalmente, a la influencia de otros barcos de la misma flota o una flota de la competencia, incluye también a este problema dentro de aquellos basados en procesos en los que intervienen situaciones de concurrencia, a los que también se le dedica un apartado especial en el campo de los problemas de ingeniería e investigación operativa.

Finalmente, al igual que en el problema anterior y en la mayoría de los problemas modelados con un sistema multiagente, el problema de fletes es un problema inherentemente dinámico. Esto se debe a que para cada uno de los agentes la existencia del resto de los agentes que interactúan con él hace que el entorno en el que evoluciona esté en constante cambio. Adicionalmente, la combinación de un sistema dinámico con el uso de variables estocásticas para generar las demandas hace que realmente para los métodos de análisis este problema sea inabordable.

Desde un punto de vista práctico, otro de los objetivos por los que se ha elegido este problema para su estudio es el hecho de que abarca varios campos que están presentes en la mayoría de los problemas reales de ingeniería. Así, aunque posee un modelo sencillo, presenta gran complejidad y la estrategia de optimización trabajará a diferentes niveles: sistema de control, diseño individual de cada barco y configuración de toda la flota.

Al igual que en el problema de vigilancia, veremos que este problema permite de forma sencilla realizar la formulación de manera descentralizada y es inherentemente distribuido. Para no depender de un sistema de control central, se ha desarrollado un sistema de cruce basado en los

puntos de demanda que permite una interacción local con el entorno, y así evitamos perder las ventajas de una aproximación descentralizada en cuanto a escalabilidad, adaptabilidad o robustez, y por supuesto, esto permite el uso el CeAS para la evolución de la flota.

En cuanto a las restricciones, característica fundamental dentro de los problemas reales en ingeniería, en este ejemplo podemos considerar como más relevantes las siguientes:

- Las restricciones de carga, que son aquellas que vienen dadas por la capacidad de cada barco y cuyo principal efecto es que limitan la recompensa que puede obtener un barco al alcanzar un punto de demanda.

- Las restricciones de entrega, que son aquellas que limitan la entrega que vamos a realizar, y son generadas por dos factores: el valor de demanda en un instante determinado en el punto de demanda y los planes de entregas de otros barcos que satisfagan previamente parte de la demanda.

- Las restricciones que afectan al rango de valores que pueden tomar variables como la capacidad, velocidad o periodo de revisión de un barco.

Con respecto al análisis energético de la población, en este ejemplo el tamaño de la población es variable al igual que en el ejemplo del capítulo 5, sin embargo, a diferencia de aquel, en el que todos los agentes eran iguales en cuanto a diseño y consumo de recursos, en este caso cada uno de los barcos posee una tasa de consumo distinta que dependerá de su capacidad, de su velocidad y de su autonomía. Esto añade un grado de libertad a mayores sobre el análisis de población, es decir, la población resultante para un tasa de entrada de recursos fija (nivel promedio de demandas) podría ser tanto una población numerosa de barcos de poco consumo como una población más reducida de barcos de consumo elevado como una población heterogénea que es lo que realmente ocurre, lo cual hace que el análisis energético no es abordable, al menos no para establecer los puntos de equilibrio del sistema.

En cuanto al análisis paramétrico definido en el capítulo 4, otro de los objetivos que perseguimos con este ejemplo es el de probar este tipo de análisis con un nuevo caso práctico y así comprobar la generalidad de este procedimiento, con este fin se realizará un análisis similar siguiendo el procedimiento definido.

Con el objeto de generar, ya no solo un problema con características similares a un problema real, sino que además presentase una aplicación directa, se ha incluido en el modelo el componente económico como el factor que va representar el nivel de calidad de los comportamientos, y por tanto, el objetivo de la evolución. El problema de más alto nivel, pasa a ser ahora un problema de optimización de recursos, y se formula de tal manera, que el estudio energético sea ahora un estudio económico y la calidad global sea el *nivel de recursos* económicos que posee la flota de barcos. Para ello, se ha definido una variable denominada nivel de recursos para cada uno de los barcos que va a ser un indicador de lo rentable o no que está siendo un barco en un momento dado. Esto se ha hecho de modo que el problema modelado represente a la flota de una empresa determinada de forma que, inicialmente, estos agentes poseen un nivel de recursos dado, y que con el tiempo, en función del consumo asociado a cada barco, dichos recursos vayan disminuyendo y aumentando en cuanto se realizan las entregas. De este modo, los barcos no rentables, o bien por su diseño o bien por su estrategia, son eliminados y los barcos que dan buen resultado, y por tanto generan superávit de recursos, se reproducen, ceden parte de su presupuesto y generan nuevos barcos basados en sus características. Desde el punto de vista de la evolución, ceden parte de su energía y tiempo de vida para generar descendencia. Desde el punto de vista económico reinvierten beneficios para generar una flota más rentable.

Por último, señalaremos también que se ha perseguido, en esta ocasión, que el trabajo del evolutivo se desarrolle a más alto nivel que en el caso del ejemplo anterior y en la mayoría de las aplicaciones de este tipo. Es decir, las acciones que van a realizar los agentes van a estar guiadas, en primer lugar, por una lógica de evaluación que ha sido desarrollada expresamente pero que representa la aplicación de algunas reglas de sentido común, y sobre esta primera evaluación el sistema de control afinará su estrategia para determinar cuál es el destino al que debe dirigirse

cada barco. Esto nos permite obtener resultados más eficientes, o al menos igual de eficientes pero con menos carga de trabajo para el evolutivo, que ahora precisará de un sistema de control más sencillo. En el apartado de diseño experimental se explicará en detalle cuáles son los parámetros sobre los que trabaja el sistema de control.

7.2. Configuración experimental

A continuación describiremos cuáles son los elementos que forman parte del escenario de fletes de buques y que interacciones se generan entre estos de acuerdo a la metodología de diseño propuesta en el capítulo 4.

Definición de los individuos

Para este escenario, la definición de los individuos más directa se extrae con facilidad de las condiciones del problema. Cada uno de los individuos será cada uno de los barcos de la flota con la que estamos trabajando, que serán los agentes que se van a encargar de transportar la mercancía.

En cuanto a los sensores de los que disponen los agentes, estos permiten analizar ciertas características de cada una de las rutas potenciales de un barco, a saber:

1. **Sensor de recompensa:** el nivel de recursos estimado que tendrá el barco cuando haga la entrega siguiendo una ruta concreta.
2. **Sensor de recursos mínimos:** indica el nivel mínimo de recursos que se alcanza durante el recorrido de una ruta concreta.
3. **Sensor de distancia:** la distancia a la que se encuentra el destino siguiendo una ruta concreta.
4. **Sensor de autonomía:** la autonomía que tendría el barco al llegar al punto de mantenimiento más cercano después de hacer la entrega siguiendo una ruta concreta.
5. **Sensor de carga:** la carga que tendría el barco tras hacer la entrega

En cuanto al criterio elegido para generar las rutas potenciales, los barcos definen sus rutas teniendo como destino final de la ruta siempre un punto de demanda. Es decir, en cuanto abandonan un punto de demanda, su sistema de control decide el próximo punto de demanda que van a visitar. La lógica que decide el itinerario se basa en lo que, para viajar desde el punto actual en el que se encuentra un barco hasta el próximo punto de demanda el sistema de control contempla cuatro posibles itinerarios (ver figura 7.1):

1. Un itinerario que pase por un punto de suministro antes de llegar al punto de demanda. El punto de suministro es aquel que minimice la distancia hasta el punto de demanda.

2. Un itinerario que pase por un punto de mantenimiento antes de llegar al punto de demanda. El punto de mantenimiento es también el que haga mínima la longitud del itinerario.

3. Un itinerario que pase por un punto de demanda y por un punto de mantenimiento antes de alcanzar el punto de demanda. Se seleccionan los dos puntos de los apartados anteriores.

4. Un itinerario que siga una ruta directa hacia el punto de demanda.

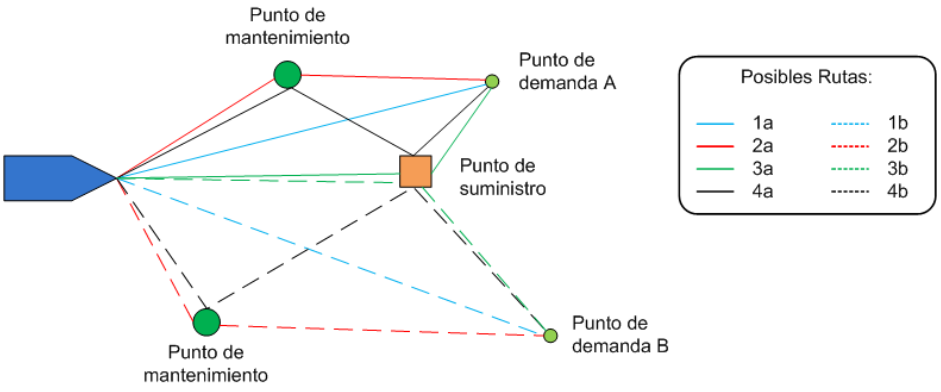


Figura 7.1: Generación de itinerarios para la evaluación mediante el sistema de control.

De esta manera, cualquier acción de un barco para elegir la dirección que debe tomar responderá a una de estas cuatro posibilidades para cada uno de los puntos de demanda sin pérdida de generalidad.

En el caso de los barcos, al igual que en el ejemplo de búsqueda de rutas en grafos, el sistema de control se utiliza como un evaluador, es decir, de entre las múltiples rutas que hemos comentado que se le presentan al barco, el evaluador asigna una puntuación a cada una de ellas y toma la dirección de la más valorada. Para realizar esta evaluación, el sistema de control recibe como entradas los valores que se obtienen de los sensores de autonomía, de carga y de recursos mínimos y como última entrada el resultado de dividir el valor del sensor de recompensa por el valor del sensor de distancia.

La primera entrada ('post-carga') nos indica con qué nivel de carga nos vamos a encontrar tras hacer la entrega. Esto permite que, en caso de que el nivel sea bajo pueda, por ejemplo, valorarse positivamente un itinerario incluya un punto de suministro para que el barco, tras hacer la entrega, tenga carga suficiente como para hacer otra nueva entrega sin tener que volver a cargar.

La segunda entrada ('post-autonomía') representa lo cerca que estamos de encontrarnos en una situación en la que ya no podamos alcanzar el punto de mantenimiento sin superar la autonomía del barco, lo que supondría la eliminación del mismo. Con el objeto de evitar tener que introducir la distancia hasta el punto de mantenimiento más cercano y el tiempo disponible, y forzar al sistema de control a realizar cálculos innecesarios, procesamos estos datos y generamos un parámetro de más alto nivel. Si este valor es menor que cero en alguna de las rutas disponibles significa que, siguiendo esa ruta, el barco llegaría a una situación de no retorno en la que sería eliminado con toda seguridad. Si ese es el caso, esta ruta ya no es una opción para el sistema de control.

La tercera entrada ('min-recursos') da una indicación del nivel de riesgo que posee una ruta teniendo en cuenta que la previsiones de entregas pueden fallar debido al alto dinamismo del entorno.

La última entrada ('tasa-recompensa') que utiliza el evaluador tiene un significado muy directo: teniendo en cuenta el tiempo estimado para alcanzar el punto de demanda y el consumo de recursos durante el trayecto (C_t), el coste de hacer la entrega (C_e) y la recompensa estimada por hacer la entrega (R_{est}), se estima también la recompensa por unidad de tiempo que se obtiene utilizando una ruta concreta. El coste de realizar la entrega que es proporcional a la capacidad de acuerdo a los costes reales de atraque de un barco en puerto. El consumo de recursos se obtiene multiplicando la tasa de consumo por el tiempo de duración del recorrido. El tiempo de duración del recorrido (t) se obtiene a partir de la distancia (D) y de la velocidad del barco (V_b). Esta es realmente una variable muy relevante, pues el objetivo de los barcos es recibir la mayor recompensa posible:

$$T_{rec} = \frac{R_{est} - C_t - C_e}{t}, t = \frac{D}{V_b} \tag{7.1}$$

$$C_e = Q \cdot c, C_e = C \cdot t \tag{7.2}$$

El sistema de control que se encargará de ser el evaluador, va a consistir en una función de evaluación que tendrá como valor base la *tasa-recompensa* que estará multiplicada por una serie de coeficientes (C_{pc}, C_{pa} y C_{mr}) que representan el efecto de los distintos valores obtenidos en el resto de los sensores. La valoración de cada trayectoria será por tanto:

$$V = T_{rec} C_{pc} C_{pa} C_{mr} \tag{7.3}$$

Los coeficientes de post-carga, post-autonomía y min-recursos se obtienen a partir de los valores de los sensores. Se normalizan y se obtiene el valor del sensor en función tres parámetros que definen una función de transformación. El valor base para estos coeficientes es 1. Cada una de las entradas tiene asociados tres parámetros que generan una variación con respecto al valor base. Los primeros 2 parámetros representan el punto de mayor desviación y la amplitud de esta (p_1 y p_2) y el tercer parámetro representa la anchura de la zona afectada (p_3).

La función de transformación se muestra en las figuras 7.2 y 7.3.

Por lo tanto, cada uno de los coeficientes se obtendrá a partir tres parámetros (p_1, p_2, p_3) y del valor de entrada normalizado (E_i^n).

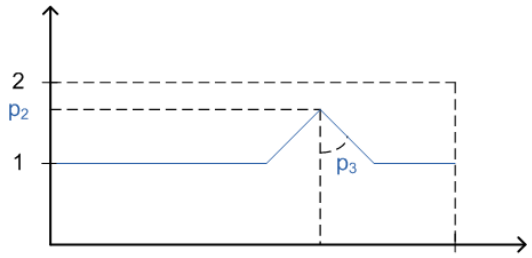


Figura 7.2: Función de transformación entrada-coeficiente. Ejemplo 1.

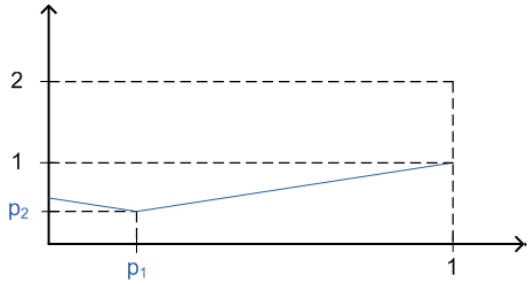


Figura 7.3: Función de transformación entrada-coeficiente. Ejemplo 2.

$$C_i = F_t\left(p_1, p_2, p_3\right) \tag{7.4}$$

Se ha utilizado este tipo de representación en lugar de alguna otra más compleja, como por ejemplo una red neuronal, con el objeto de reducir el espacio de búsqueda y de hacer más sencilla la interpretación del comportamiento de cada uno de los agentes por parte del diseñador.

Por tanto, cada barco va a poseer los siguientes 12 parámetros evolucionables que constituyen el genotipo de cada individuo de la población:

- La capacidad (Q), que indica la cantidad máxima de carga que puede transportar un barco.
- La autonomía (A), que nos indica el tiempo que pueden permanecer navegando hasta que necesitan realizar una revisión de mantenimiento.

- La velocidad (V), que representa el número de unidades que se desplaza en cada intervalo de tiempo.

- Finalmente los parámetros que determinan la función de transformación para cada una de los tres coeficientes de la función de evaluación: $(p_1^{pc}, p_2^{pc}, p_3^{pc}, p_1^{pa}, p_2^{pa}, p_3^{pa}, p_1^{mr}, p_2^{mr}, p_3^{mr})$

Utilidad global y utilidad individual

El uso de una población heterogénea y del CeAS nos conduce, como en los ejemplos anteriores, a definir una asociación entre la supervivencia del código genético de un individuo y el éxito en la tarea global. Debido al uso de un modelo económico, el objetivo global es obtener una flota eficiente y esto se consigue obteniendo individuos que sean lo más eficientes posible, es decir que generen más recursos.

Utilizando la misma metodología que en los escenarios anteriores, la medida de utilidad global que se extrae de este planteamiento se obtendría a partir del sumatorio del nivel recursos de cada uno de los barcos (r_b).

$$G(z) = \sum_{barcos} r_b \tag{7.5}$$

El objetivo será maximizar este valor de utilidad global. Para calcular la utilidad diferencial propuesta en el capítulo del CeAS debemos definir un segundo término que recordemos representaba el valor de utilidad del entorno si eliminamos los efectos un agente concreto. Para nuestro caso este valor se obtiene eliminando del sumatorio el valor de recursos de un barco (r_i). Por lo tanto, la utilidad diferencial será:

$$G(z_{-i} + c_i) = \sum_{barcos_{-i}} r_b \tag{7.6}$$

$$D_i = r_i \tag{7.7}$$

Es decir, la utilidad diferencial de cada barco corresponde a su nivel de recursos.

Modelado distribuido y descentralizado

El modelado del entorno se realiza mediante la creación tres tipos de elementos (ver figura 7.4).

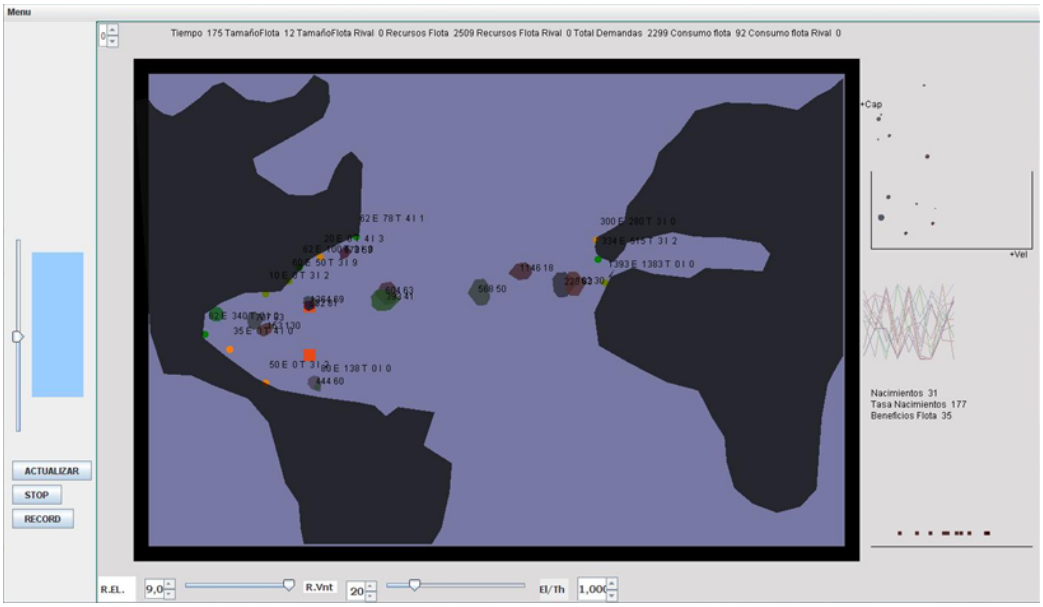


Figura 7.4: Escenario de simulación para el problema de fletes. Se muestran los puntos de suministro (cuadros naranjas), puntos de demanda (círculos verdes y naranjas), puntos de mantenimiento (cuadros verdes) y los barcos de la flota cuyo color y tamaño indican un código genético y capacidad respectivamente.

Los puntos de demanda: son elementos encargados de generar las demandas de material y recibir las entregas. Poseen tres funciones principales:

1. Por un lado son los puntos en los que se introducen los recursos en el sistema, y de este modo, son en realidad el motor de la simulación.

2. Sirven para gestionar la generación de nuevos agentes mediante el torneo que se realiza con el código genético que depositan los barcos que hacen las entregas.

3. Almacenan las planificaciones de los agentes de una forma distribuida y descentralizada.

Los puntos de suministro: estos puntos representan los almacenes o productores de material a transportar y no realizan ninguna acción propia. A efectos de la flota de barcos en la simulación, representan solo un emplazamiento por el que hay que pasar para aumentar el valor de la carga hasta el valor de capacidad. Por tanto, una vez que un barco pasa por un punto de suministro, este llena completamente su capacidad.

Los puntos de mantenimiento: en ellos se realiza una acción que es imprescindible para la supervivencia de los barcos, una revisión de mantenimiento. Los barcos tienen asignado lo que se denomina un valor de tiempo de funcionamiento hasta que requieren una revisión. Si dicha revisión no se produce dentro del tiempo requerido, el barco es eliminado. Esto representa una restricción fuerte para el comportamiento de los barcos, y existe un rango para el tiempo de revisión que varía para cada uno de los barcos pero que si aumenta, aumenta el consumo del barco, con lo que la flota buscará un punto de equilibrio consecuente.

La realización de las interacciones entre los elementos y con los individuos en el escenario se define a partir de los cuatro niveles especificados en la metodología:

- **Información de entorno**

  La información de entorno estará totalmente distribuida entre los distintos puntos de demanda, suministro y mantenimiento y estará accesible para cualquier barco bajo petición. Esta información puede ser o la posición de cualquiera de los tres distintos tipos de puntos o bien, para el caso de los puntos de demanda, la información relativa a las demandas emitidas por cada punto y no satisfechas hasta el momento actual.

- **Información de grupo**

  En cuanto a la comunicación entre individuos, esta se produce de manera implícita y descentralizada a través de los puntos de demanda. Concretamente, se realiza a partir de la gestión de las entregas estimadas que se realizarán en cada punto de demanda y tiene como objeto permitir cierto nivel de planificación en las acciones de los barcos.

  El modo de funcionamiento es el siguiente: cuando el sistema de control de un barco decide cuál es su itinerario, se comunica con el punto de demanda y le envía información relativa a la hora estimada de llegada y la carga que tendrá en el momento de la entrega. El punto de demanda almacena esta información y la hace disponible para el resto de los barcos, de manera que cuando otro barco está comparando las distintas posibilidades de destinos que tiene, el punto de demanda le indica, en función de la hora de entrega y de la carga que podría entregar, cuál es la recompensa esperada en caso de seleccionar esa ruta. Para ello, se deben tener en cuenta las demandas que han sido generadas, el tiempo que llevarán generadas cuando se haga la entrega y las entregas que se producen antes de que llegue este barco.

  Para poder proporcionar esta información, los puntos de demanda almacenan dos listas, una corresponde una historial de demandas y otra a una planificación de entregas. A medida que un punto de demanda va generando peticiones de demanda, estas se almacenan en el historial de demandas, tanto el tiempo en el que fueron generadas como la cantidad solicitada. De igual manera, cuando uno de los agentes decide que va a realizar una entrega en un punto de demanda, este almacena la carga entregada estimada y la hora estimada de llegada.

  Esta pequeña planificación evita, de manera relativamente sencilla, algunos comportamientos altamente ineficientes, como por ejemplo, que dos barcos estén viajando hacia el mismo punto de demanda

con el fin de hacer una entrega cuando el que llegue más tarde no realizará ninguna entrega o esta será más reducida de lo que previsto.

- **Actuación sobre el entorno**

Las interacciones que se realizan entre los individuos y el entorno son cuatro y todas ellas se realizan de forma descentralizada: el suministro de carga, la generación de demandas, la entrega de material y la realización del mantenimiento.

1. El suministro de carga se produce cuando un barco se sitúa en un punto de suministro, en ese momento la carga del barco se aumenta hasta el valor de su capacidad.
2. La entrega de material se produce cuando un barco alcanza su destino en un punto de demanda, el barco descarga tanto material como sea necesario para satisfacer las necesidades de demanda hasta agotar toda la carga si fuese necesario y recibe la recompensa correspondiente. Este valor de recompensa se genera a partir de un valor de recompensa por unidad entregada que se obtiene con una función que depende del tiempo de espera que ha tenido la demanda que se está cubriendo y es la misma función que se utiliza para calcular la recompensa estimada. La configuración por defecto es que el valor de la recompensa es de una unidad por cada unidad de carga entregada. Es decir, es independiente del tiempo de espera.
3. La maniobra de mantenimiento se ejecuta también de forma automática cuando se alcanza un punto de mantenimiento y reinicia el tiempo de revisión al valor de autonomía.
4. Por último, para realizar la generación de las demandas, cada uno de estos puntos posee dos parámetros que caracterizan la distribución de probabilidad que siguen estas peticiones de demanda: el tiempo entre demandas y la cantidad solicitada. Cada uno de esos parámetros sigue una distribución constante entre el valor mínimo y el máximo, y por tanto, el promedio de

demandas por unidad de tiempo vendrá dado por la ecuación:

$$\overline{D} = \frac{\overline{C}}{\overline{\overline{T}}} = \frac{C_{max} - C_{min}}{T_{max} - T_{min}} \tag{7.8}$$

- **Actuación en el grupo**

  La única interacción que existe entre miembros de la población es la de reproducción. Esta se explicará en el apartado de selección reproductiva.

Gestión energética

Este escenario se centra en la gestión de los recursos de una flota de barcos y por tanto, el diseño se ha realizado de manera que el nivel de energía equivaldrá al nivel de recursos. Estos recursos representan el dinero o presupuesto disponible en cada uno de los barcos en cada instante, y por tanto, la suma de todos estos será el presupuesto actual de la flota. Todos los comportamientos que se producen en el entorno giran alrededor de los recursos y son equivalentes a la energía o la recompensa de los anteriores ejemplos. El objeto de modelizar el entorno en base a una variable económica persigue la obtención de soluciones más realistas y con una aplicación más práctica, ya que en el caso de una empresa real que gestione una flota, este va a ser siempre el objetivo último: mantener o aumentar el nivel económico de la flota.

- **Entrada de energía:** la entrada de energía se produce a través de los puntos de demanda. Cuando un barco realiza una entrega recibe un valor de recompensa asociado a la cantidad de carga que deposita. Este valor de recompensa representa la única entrada de recursos de la que se dispone en el escenario tras la inversión inicial que representan los recursos iniciales de cada barco. Esta entrada se realiza distribuida y descentralizadamente a través de todos los puntos de demanda y está supeditada a la acción de entrega. Por tanto, se realiza de manera asíncrona.

  En cuanto al valor de recompensa, este lo definen los puntos de demanda y representa el precio que pagan los puntos de demanda por

cada unidad de carga. El precio puede variar en función del tiempo transcurrido desde la generación de la demanda, aumentar porque resulte más apremiante recibir la entrega o disminuir si se quiere establecer algún tipo de penalización al retraso.

- **Salida de energía:** la salida de energía se produce de dos maneras, de forma que represente el consumo real de recursos de una flota:

  1. El consumo de recursos por unidad de tiempo, que se produce durante la vida útil de cada barco y que representa dos factores económicos: los gastos variables de un barco, que serán el consumo de combustible, y los costes de mantenimiento.
  2. Los gastos fijos asociados a la amortización del coste total del barco. El cálculo de esta tasa de consumo (t_c) se realiza mediante un consumo base que es función de la capacidad del barco (Q) y unos coeficientes que representan el coste asociado a un sistema de propulsión, y que será función de la velocidad, el coste asociado al rendimiento del sistema de propulsión y al consumo de combustible, que será también función de la velocidad, y el coste asociado a las horas de operación entre cada revisión de mantenimiento, y que será función de la autonomía. Las distintas curvas de costes (ecuación 7.9) se han estimado y luego se han ajustado en función de unos datos obtenidos de un estudio sobre economías de escala en el transporte marítimo (Stopford, 2002) y que han permitido obtener un alto grado de verosimilitud de los modelos, en las figuras 7.5 y 7.6 se muestran las curvas de consumo real y unitario en función del tamaño del barco.

$$t_c = \frac{Q + 100}{150} C_{propulsion} C_{consumo} C_{autonomia} \tag{7.9}$$

$$C_{propulsion} = 0{,}8 + 0{,}2 \frac{V}{V_{min}} \tag{7.10}$$

$$C_{consumo} = 0{,}6 + 0{,}4 \left(\frac{V}{V_{min}} \right) \left(\frac{1}{2} + \frac{V}{2 \cdot V_{min}} \right) \tag{7.11}$$

$$C_{autonomia} = 0{,}9 + 01 \left(\frac{A - A_{min}}{A_{max} - A_{min}} \right) \tag{7.12}$$

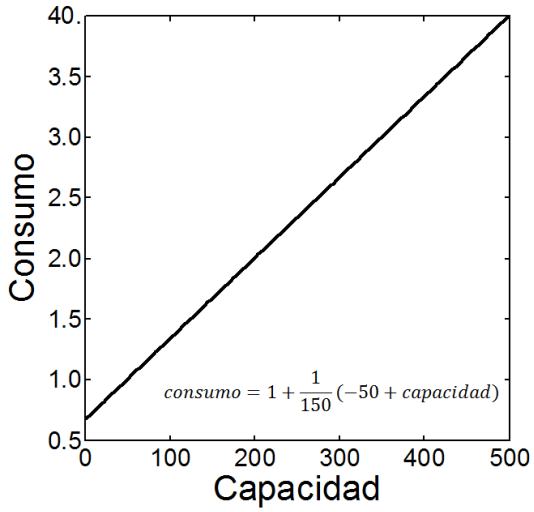


Figura 7.5: Relación entre la capacidad y el consumo por paso de tiempo de un barco.

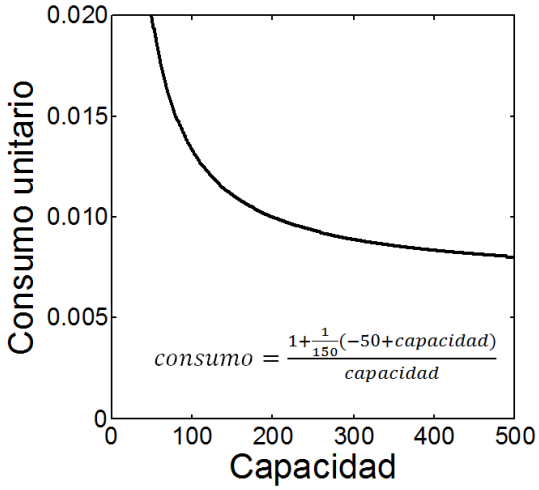


Figura 7.6: Relación entre la capacidad y el consumo unitario (consumo por unidad de carga) por paso de tiempo de un barco.

En realidad no es relevante que los costes sean exactos a la hora de modelar los costes del mercado actual, lo que sí es re-

levante es la relación entre el coste del transporte por unidad de carga y el tamaño del barco, que ha sido lo que hemos ajustado según la bibliografía. De todos modos, no es el objeto de este caso práctico el modelado fidedigno de los modelos de mercado del sector de transporte marítimo, sino el establecimiento de un procedimiento de modelado y evolución que nos permitan obtener una solución adaptada a unas condiciones concretas, sean las que sean.

A partir de este valor de tasa de consumo, el barco inicia su simulación con un nivel de recursos (presupuesto) determinado, este disminuye según la tasa de consumo, y si agota su presupuesto, el barco es eliminado. Existe otro gasto asociado a los barcos y es el que se debe realizar cuando se hace una entrega en un punto de demanda. En los puertos, esta comisión es función del tamaño del barco que hace la entrega, y por tanto, la hemos modelado como proporcional al tamaño del barco.

- **Intercambio energético:** el intercambio energético entre agentes se produce sólo durante la reproducción. En ella se le proporciona por parte de cada individuo base una cantidad de energía al nuevo agente. Esta cantidad de energía se obtiene a partir de un valor denominado *distancia de madurez*, y representa un valor de energía que debe proporcionársele a cada nuevo agente para que pueda sobrevivir durante un tiempo mínimo que le permita autoabastecerse de recursos aún si las circunstancias iniciales son adversas. Esta cantidad de energía mínima hace más robusta la evaluación de códigos genéticos evitando eliminaciones prematuras. En definitiva, se extrae cierta cantidad de energía a los individuos base a cambio de perpetuar partes o variaciones de su código genético.

- **Asociación energía-utilidad:** con respecto a las consideraciones para la asociación entre la gestión de la energía y la utilidad de los comportamientos de cada agente para la realización de las tareas deseadas, debemos destacar que, en primer lugar, el tratamiento de la energía como recursos económicos dirige de forma directa esta asociación hacia el campo que nos interesa, el de la optimización económica.

En primer lugar debemos considerar de forma independiente la utilidad global y la individual, de manera que las metas de cada uno de los agentes les guíen hacia el objetivo global. En este modelo, el objetivo base que hemos definido será el de realizar las entregas de la manera más rentable, es decir, maximizar la recompensa obtenida con una entrega frente al coste que supone esa entrega. Esto genera dos objetivos globales, en primer lugar, satisfacer las demandas lo antes posible, y por tanto, optimizar la eficiencia en el reparto, y en segundo lugar, se busca la supervivencia de un mercado de competencia, es decir, que cada uno de los barcos intente maximizar sus beneficios de cara a permitir eliminar competidores dentro del mercado de reparto.

Para ello debemos considerar cuál será la recompensa que se generará al hacer una entrega. Con respecto a eso, hemos barajado tres opciones de generación de recompensa. La idea base de la primera es la de penalizar las entregas que tengan un valor de espera muy alto y así forzar a los barcos a realizar las entregas cuanto antes. Sin embargo, esto provoca un efecto que en el modelo planteado no nos interesa: cuando una entrega pasa de un cierto tiempo sin haber sido satisfecha, el valor de recompensa baja de modo que no resulta rentable y no se sirve en ningún caso.

Otra opción es la de generar recompensas mayores cuando la entrega se hace más tarde, es decir, la recompensa aumenta con el tiempo de espera con el objeto de hacer más atractivos los puntos que llevan tiempo sin servir y así evitar tiempos de espera demasiado dilatados. Sin embargo, esta aproximación presenta dos inconvenientes, uno es que la población posee un punto de equilibrio rentable con altas recompensas realizando entregas lentas con barcos lentos y de poco consumo, y el otro es que realmente no refleja al mercado real ya que aumentar la recompensa cuanto peor sea el servicio no parece lo más adecuado.

Finalmente, la opción más interesante resulta de mantener constante la recompensa por cantidad entregada independientemente del tiempo de espera. Esto crea una competencia entre barcos cuyo re-

sultado es la disminución de los tiempos de entrega.

Tal y como hemos mencionado, y para que los balances del nivel de recursos que posee la flota tras la generación de un barco nuevo sean correctos, parte de los recursos de cada uno de los individuos base se le suministran al nuevo barco y representan su nivel de energía inicial.

Finalmente, este valor de recursos estará asociado a la supervivencia del código genético de un individuo mediante dos mecanismos. En primer lugar, aquellos individuos que no generen un balance de recursos positivos serán eliminados cuando su valor de recursos sea negativo. Estos individuos (y por tanto su código genético) permanece menos tiempo en el escenario y además tienen menos tiempo para reproducirse, y por tanto, el número medio de nuevos individuos que les sobreviven y que portan fracciones o modificaciones de su código genético, es también más reducido. En segundo lugar, este valor de recursos, tal y como veremos en el siguiente apartado, sirve como valor de calidad en la comparación que se realiza para elegir los individuos base, y por tanto, está relacionado directamente con la probabilidad de generar descendencia.

Selección reproductiva

En cuanto al modo de reproducirse, se requiere que el número medio de cruces realizados, así como la probabilidad de cruzarse estén asociados a la proximidad del comportamiento de cada barco con el comportamiento objetivo, que es la realización de entregas eficientes. Para poder llevar esto a cabo, debemos asociar tanto el proceso de selección como la probabilidad de ser elegidos como individuos base al nivel de utilidad y a la realización de tareas asociadas al comportamiento objetivo.

- **Proceso de selección:** la selección debe ser descentralizada y asociada a la tarea objetivo, tal y como corresponde al procedimiento propuesto por CeAS, que es la entrega de carga en los puntos de demanda. Podríamos llevar a cabo la reproducción en estos puntos

de demanda al igual que en el caso del problema de rutas en grafos, donde se realizaba cuando los agentes alcanzaban el nodo origen tras haber recorrido el grafo con éxito. Sin embargo, debido a que los barcos en este modelo no coinciden espacialmente, y de hecho cuanto más eficientes son menos tienden a juntarse para realizar entregas más grandes y así no interferirse, hemos tenido que generar otro tipo de procedimiento para cruzarse:

Cuando un barco alcanza un punto de entrega, si este posee el nivel mínimo de energía requerido para el cruce, deja una copia de su código genético en el punto de demanda. Los barcos que han dejado su código genético en un punto de demanda componen el torneo de la selección reproductiva. Cuando el punto de demanda posee un número de códigos genéticos correspondiente al tamaño de torneo definido por el diseñador, se seleccionan los agentes que han dejado su código genético, se evalúan y se seleccionan los individuos base que luego se cruzan para generar otro barco.

- **Asociación selección/nivel de utilidad:** la asociación entre la selección reproductiva y el nivel de utilidad, tal y como comentamos, se va a basar en la realización de la tarea objetivo. En primer lugar el tiempo de vida estará relacionado con el nivel de calidad de un agente, y así con el número promedio de descendientes que va a generar. En segundo lugar el criterio de selección se basa en el nivel de utilidad y por tanto vamos a asociar a la probabilidad de reproducirse el número de entregas realizadas y la disponibilidad de recursos suficientes para generar un nuevo barco. De este modo conseguimos guiar el comportamiento hacia un mayor número de entregas, pero al mismo tiempo, hacia individuos que generen recursos.

- **Parametrización de la selección reproductiva:** hay dos parámetros que debemos considerar en cuanto al diseño del procedimiento de selección reproductiva. En primer lugar, el tamaño del torneo seleccionado, y en segundo lugar, el límite mínimo de recursos necesarios para la reproducción. Ambos parámetros nos darán una indicación del equilibrio exploración/explotación de la evolución de la población. Adicionalmente, el límite mínimo de recursos influye

en la fluctuación en la evaluación de individuos, generando mayor o menor dependencia sobre las condiciones externas.

Evaluación

Con respecto a la evaluación que haremos de los barcos para seleccionar los individuos base de entre los individuos de la selección reproductiva, esta se basará en una medida de calidad obtenida a partir del nivel de recursos y de la tasa de consumo, y que representa el margen de ganancia de cada barco, es decir, la cantidad de recursos que genera cada barco por cada unidad de consumo (ecuación 7.13):

$$E_f = \frac{Recursos}{Consumo} \tag{7.13}$$

- **Estudio de los criterios de evaluación/asignación de utilidad.**

  El criterio elegido guía la evolución hacia individuos con un nivel de eficiencia económica más alto. Tal y como hemos comentado esto se obtiene optimizando dos variables: maximizando los recursos obtenidos y minimizando el consumo asociado a conseguir esos recursos.

- **Estabilidad de la utilidad.**

  Teniendo en cuenta que la utilidad decrece de manera constante y que se incrementa al realizar una entrega, en este escenario el nivel de utilidad presenta un nivel de fluctuación del orden de las entregas realizadas. Teniendo en cuenta que el valor de energía no se eleva muy por encima del valor de energía de madurez, el valor de la oscilación relativa será aproximadamente el valor de la entrega media entre el valor de energía de madurez. Es, por tanto, mayor para los barcos grandes y menor para los barcos de menor consumo.

Combinación genética

Tal y como hemos explicado en el capítulo de definición del CeAS, se utilizará el cruce bipolar con los parámetros de ajuste por defecto. El nivel

de exploración y de explotación vendrán definidos por estos parámetros que representan la mayor o menor amplitud de las funciones de probabilidad que generan los nuevos genes a partir de los genes de los individuos base.

Reemplazo

El control del reemplazo lo vamos a llevar a cabo mediante el uso del tiempo de vida máximo de cada individuo o tiempo de permanencia, y mediante el uso del valor mínimo de recursos para la reproducción, que representa el coste energético para los individuos base del evento de reproducción. En este sentido, en caso de utilizar coste elevados, estamos disminuyendo el elitismo, y por tanto, aumentando la explotación. Estos parámetros se ajustarán con el objeto de obtener el comportamiento evolutivo deseado. Tal y como hemos comentado, dentro del modelo diferenciamos cuatro tipos de elementos. Los barcos, los puntos de demanda, los puntos de suministro y los puntos de mantenimiento.

7.3. Planteamiento en el WaspBed

Para llevar a cabo la implementación en el WaspBed del modelo que hemos expuesto y que representa un problema de fletes de barcos, se han generado un conjunto de eventos y elementos que expondremos a continuación. El diagrama de clases de la implementación de este ejemplo se muestra en la figura 7.7.

La definición de los tipos de elemento para este caso se genera a partir de los archivos XML y se muestran en la tabla de la figura 7.8

Para los elementos *punto de demanda* generamos la siguiente configuración de parámetros que aparece en la figura 7.9.

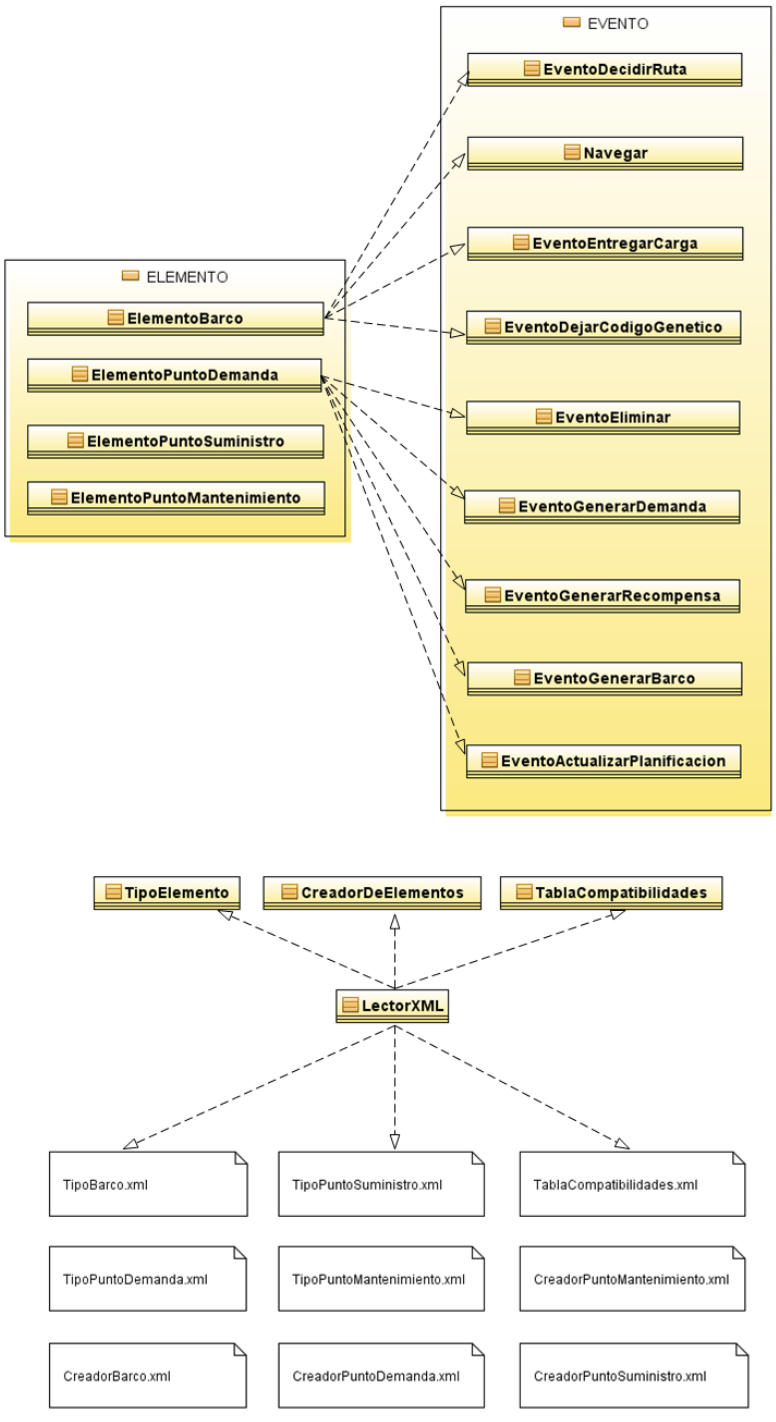


Figura 7.7: Diagrama de clases del escenario.

Elemento Barco	
Parámetros de tipo de elemento	**Valor**
Edad Límite	20000
Parámetros descriptivos de elemento	**Valor**
Coeficientes del sistema de control	Evol [-1,1]
Autonomía	Evol [1,10000]
Velocidad	Evol [5,15]
Capacidad	Evol [50,500]
Tasa Consumo	[0.5,5]
Parámetros de estado	**Valor**
Posición	[0,1000][0,700]
Carga	[0,500]
Tiempo Revisión	[1,10000]
Edad	[0,20000]
Recursos	[0, ∞]
Punto de destino	NA
Hora llegada estimada	[0, ∞]
Carga entrega estimada	[0,500]

Figura 7.8: Parámetros de: Elemento Barco.

Elemento Punto Demanda	
Parámetros de tipo de elemento	**Valor**
Tamaño torneo	4
Pasos madurez	400
Parámetros descriptivos de elemento	**Valor**
Posición	[0,1000][0,700]
Cantidad demanda	[1,500]
Intervalo demanda	[1,500]
Parámetros de estado	**Valor**
Historial Demandas	NA
Planificación entregas	NA
Lista Genes Torneo	NA
Número de barcos en torneo	[0,20]

Figura 7.9: Parámetros de: Elemento Punto Demanda.

Vemos que tenemos unos parámetros que son fijos para todos los elementos *Punto Demanda* que constituyen un tipo de parámetros de estado del escenario. Estos son el tamaño de torneo y los pasos madurez. Esta última variable indica la cantidad de recursos mínima que debe tener un barco para poder generar un barco descendiente mediante el cruce. Esta cantidad de recursos mínima corresponde a la cantidad de recursos que un barco concreto consume durante un intervalo de tiempo igual al tiempo madurez. De esa energía, la mitad se cede al nuevo barco, y por tanto se le extrae a cada uno de los barcos que son individuos base.

El siguiente elemento es el *punto de suministro* y sólo posee un parámetro descriptivo de elemento que indica su posición en el escenario (ver figura 7.15).

Elemento Punto de suministro	
Parámetros descriptivos de elemento	**Valor**
Posición	[0,1000][0,700]

Figura 7.10: Parámetros de: Elemento Punto Suministro.

Elemento Punto de mantenimiento	
Parámetros descriptivos de elemento	**Valor**
Posición	[0,1000][0,700]

Figura 7.11: Parámetros de: Elemento Punto Mantenimiento.

Por último, el *punto de mantenimiento* que posee al igual que el punto de suministro solo un parámetro de posición (figura 7.11).

```
- <TablaCompatibilidad>
-   <Asociacion>
        <Elemento nombre="barco" />
        <Evento nombre="decidirRuta" />
        <Evento nombre="navegar" />
        <Evento nombre="entregarCarga" />
        <Evento nombre="dejarCodigoGenetico"/>
        <Evento nombre="eliminar"/>
    </Asociacion>
-   <Asociacion>
      <Elemento nombre="puntoDemanda"/>
        <Evento nombre="generarDemanda"/>
        <Evento nombre="generarBarco"/>
        <Evento nombre="actualizarPlanificacion"/>
    </Asociacion>
  </TablaCompatibilidad>
```

Figura 7.12: Asociaciones Elemento-Evento.

Evento	Descripción
DecidirRuta	Genera los itinerarios posibles y los evalúa mediante el sistema de control
Navegar	Avanza en dirección al destino fijado por el sistema de control, si se encuentra un punto de suministro o de mantenimiento actualiza sus valores de carga y de tiempo de mantenimiento
entregarCarga	Se ejecuta al alcanzar un punto de demanda, se entrega la carga hasta que se satisfaga la demanda o se acabe la carga que se transporta y se obtiene el valor de recompensa correspondiente.
dejarCodigoGenético	Si se tienen suficientes recursos y se acaba de realizar una entrega se deja una copia del código genético en el punto de carga
Eliminar	Si se alcanzan los valores límites de tiempo de vida o de energía o si se llega a una situación que el sistema de control evalúa como irresoluble se elimina al barco
generarDemanda	Se genera un valor aleatorio en función de la distribución que posee ese punto de demanda para decidir si se genera o no una demanda. Si se genera esa demanda la cantidad se elige también aleatoriamente dentro del rango de la variable cantidad demanda
generarBarco	Si se alcanza un número de barcos en el torneo igual al tamaño de torneo, se seleccionan los dos códigos genéticos cuyos barcos posean mayor valor de recursos y se cruzan. Se les substrae una cantidad de recursos equivalente a la mitad de la energía para sobrevivir un tiempo igual a pasos madurez y se le suministran al nuevo barco
Actualizar	Se actualizan tanto el historial de demandas como

Figura 7.13: Descripción de eventos.

A continuación se crean las clases *Evento* correspondientes a las acciones que hemos definido para los barcos y para los puntos de demanda, y se configura la *tabla de compatibilidades* para crear las asociaciones entre los eventos y los elementos (ver figura 7.12). La figura 7.13 muestra una descripción breve de cada uno de los eventos generados.

7.4. Análisis de los parámetros de evolución del CeAS

Por último, tal y como comentamos en los objetivos de este apartado, realizaremos un estudio de los parámetros de configuración de este escenario y su influencia en los parámetros genéricos de análisis definidos por la metodología del CeAS. El principal objetivo de este estudio es identificar los parámetros que son más relevantes en este problema y poder generar unos determinados cambios dirigidos en el evolutivo, mejorar el control que tenemos sobre él, y además, permitir extrapolar estos resultados a otros casos de estudio.

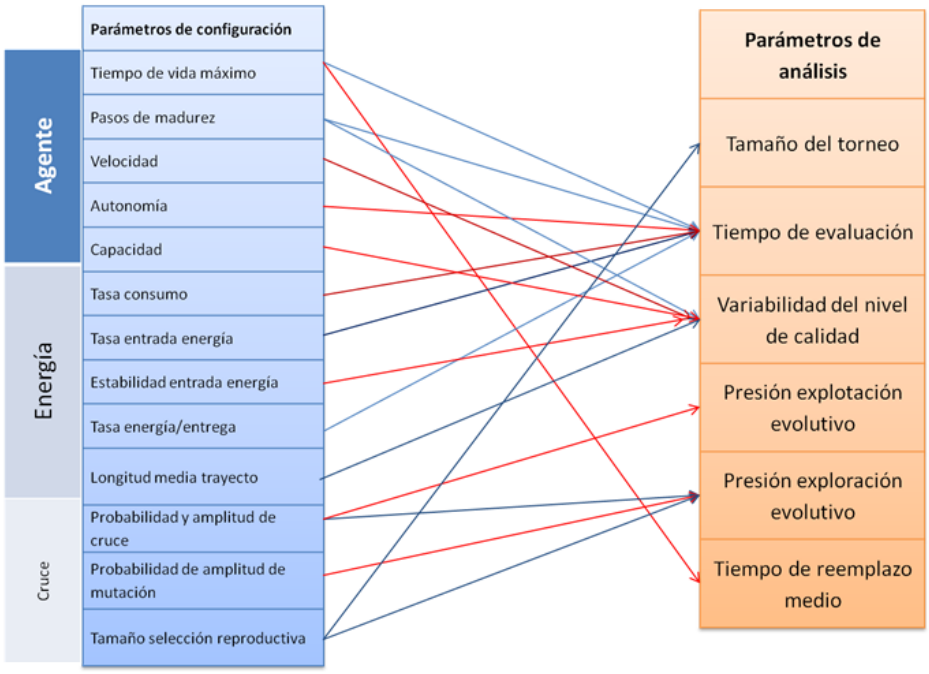


Figura 7.14: Tabla de análisis de los parámetros del escenario.

A continuación se muestra el efecto de la modificación de los parámetros de configuración en los parámetros de análisis y las asociaciones entre ellos. En la figura 7.14 mostramos estas relaciones y si la dirección es directa o inversa (para claridad en la interpretación se han utilizado flechas azules en el caso de relaciones directas y rojas en caso de inversas). El procedimiento para llevar a cabo el análisis a partir de esta tabla el mismo que hemos explicado en otros escenarios, y nos permite identificar que por ejemplo en este caso el tamaño del torneo que genera para la reproducción está controlado por un sólo parámetro de configuración, que es el tamaño de la selección reproductiva, a diferencia del ejemplo anterior en el que influían varios parámetros de configuración.

7.5. Resultados

7.5.1. Configuración básica. Mejora en la eficiencia para la satisfacción de la demanda

El primer experimento que llevaremos a cabo tendrá como objetivo comprobar de nuevo la capacidad del algoritmo CeAS para generar un comportamiento dirigido a optimizar al criterio que estamos imponiendo en el modelo que hemos diseñado. En este caso será mejorar la eficiencia en la entrega de material en los puntos de demanda.

Figura 7.15: Detalle del escenario de fletes para la configuración base. Se ven representados los puntos de suministro (cuadros naranjas), puntos de demanda (círculos verdes y naranjas), puntos de mantenimiento (cuadros verdes) y los barcos de la flota cuyo color y tamaño indican un código genético y capacidad respectivamente.

Para ello, se ha configurado el escenario de simulación para que se generen 20 puntos de demanda en los cuales existe una distribución de demandas promedio de 2 unidades por paso de tiempo. Se han creado

también un punto de mantenimiento y dos puntos de suministro. En la figura 7.15 se muestra un detalle del escenario. Los puntos de suministro se representan como cuadrados de color naranja, el punto de mantenimiento se representa con un círculo verde grande y los puntos de demanda son los círculos pequeños que cambian su color entre verde y naranja en función de si ya ha sido o no programado el abastecimiento de su demanda. Los barcos poseen distintos colores en función de su estrategia de comportamiento, su superficie indica su capacidad y su anchura su velocidad. Se muestran tambien para cada elemento valores que representan determinados parámetros como el nivel de recursos, nivel de demanda insatisfecha, tamaño del torneo, etcétera que sirven para estudiar la evolución del escenario y de la población.

La población inicial será de 50 barcos y el nivel energético inicial de cada uno de ellos será el equivalente al consumo que representa la supervivencia durante el tiempo de madurez establecido. De este modo se establecen unas condiciones iniciales equitativas para cualquier diseño de barco. Esto resultará en que, en función del tipo de barco, se requerirá una inversión inicial mayor o menor, pero en promedio, la configuración de la flota inicial generada aleatoriamente supondrá un coste inicial aproximadamente constante para todas las simulaciones. Se situarán también un punto de suministro y un punto de mantenimiento. La tabla de los elementos de creación se ven en la figura 7.16.

El comportamiento inicial de los barcos es el de intentar aumentar los recursos individuales. Una vez que la flota alcanza un nivel de recursos este no varía significativamente, la entrada de recursos está controlada por la producción de demandas y la salida por el nivel de consumo de la flota. Por lo tanto, una flota adapta su consumo a la producción de demanda. Si el consumo fuese menor, se generarían beneficios en la flota pero estos serían redistribuidos al generar otros nuevos barcos hasta que este consumo se igualase. De forma equivalente, si el consumo fuese mayor que la producción de recursos a través de las demandas la energía de los barcos menos eficientes disminuye hasta que son eliminados.

Esta dinámica conduce a que, aunque el nivel de recursos en la flota se mantenga en aproximadamente constante, lo que si va a disminuir es

la cantidad de demanda sin satisfacer, a medida que mejora la eficiencia de la flota.

Creador Elemento Barco **NEl = 50**	
Parámetros de estado	
Carga	(0)
Tiempo Revisión	(autonomía)
Edad	(0)
Recursos	(0)

Creador Punto Demanda **NEl=20**	
Parámetros descriptivos de elemento	
Cantidad demanda	(10)
Intervalo demanda	(5)

Elemento Punto de suministro **NEl=2**

Elemento Punto de mantenimiento **NEl = 1**

Figura 7.16: Tabla de creadores para el escenario de la configuración base.

La figura 7.17 nos muestra la evolución de una flota en las condiciones mencionadas. En ella se puede apreciar que una vez que la población se estabiliza en un número de barcos determinado, aproximadamente tras 4000 pasos de evolución, el nivel de demanda comienza a decrecer. Por tanto, como resultado de generar esa competición entre agentes en el nivel de eficiencia vemos que lo que obtenemos es un proceso que tiende a una mejora en la satisfacción de demandas.

Para obtener un análisis más en profundidad de lo que está ocurriendo, en la gráfica 7.18 hemos estudiado también como variaron durante este proceso de simulación los valores de nivel de recursos medio por barco y consumo medio por barco. Aquí se aprecia la estrategia seguida durante la evolución, la flota tiende a utilizar barcos con un nivel de consumo cada vez más reducido y con un margen de beneficios por barco cada vez más ajustado.

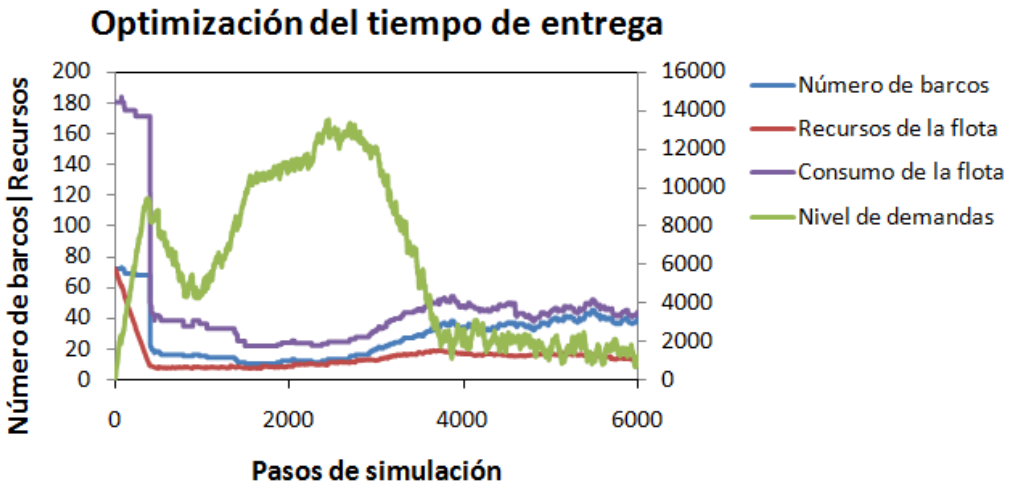


Figura 7.17: Evolución para la optimización del tiempo de entrega.

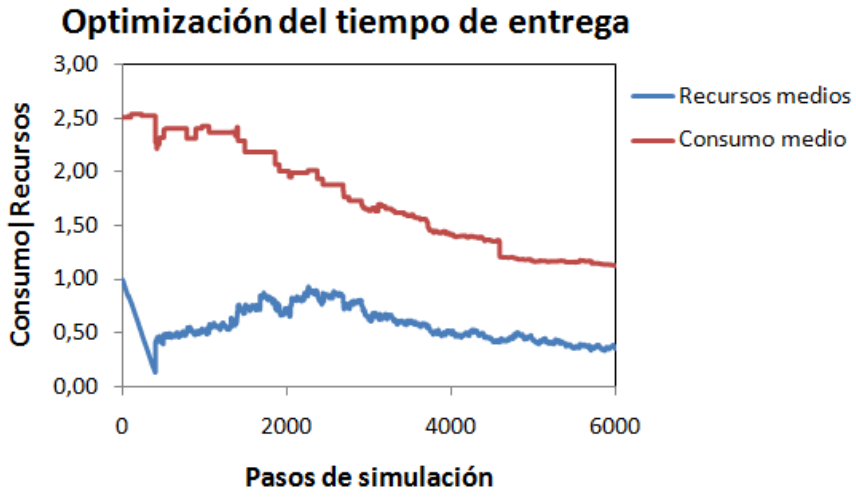


Figura 7.18: Evolución para la optimización del tiempo de entrega: recursos y consumo medio.

De nuevo, en este caso podemos concluir que, para la configuración base, hemos comprobado la capacidad del CeAS para generar soluciones a este tipo de problemas, ahora con una formulación centrada en un enfoque económico. El sistema evoluciona hacia una población que permite una satisfacción de la demanda más eficiente.

7.5.2. Variación del mercado de demanda: generación de especies

Una vez que hemos obtenido un resultado satisfactorio a la hora de optimizar la eficiencia en las demandas, el siguiente caso experimental va a estudiar la capacidad del algoritmo para generar distintas soluciones para distintos tipos de mercado. El objetivo ahora es comprobar que la configuración de los parámetros del CeAS no está ajustada de manera que su funcionamiento esté supeditado a unas condiciones del problema fijadas, si no que puede generar comportamientos que satisfagan distintos criterios objetivo.

En esta configuración experimental se van a situar 17 puntos de demanda en las cercanías (distancia media de 40 unidades) de un punto de suministro y de otro punto de mantenimiento. Estos puntos generarán una demanda promedio de 2 unidades por paso de tiempo. Se van a generar también 3 puntos de demanda a una distancia mayor (distancia media de 500 unidades) del punto de suministro, que generarán un promedio de 5 unidades por paso de tiempo (ver figura 7.19).

A continuación se definen los archivos XML de creación (ver figura 7.20).

Se pretende estudiar así la generación una flota heterogénea para satisfacer dos tipos de mercados que generan diferentes estrategias óptimas en cuanto al diseño y sistema de control de cada barco. Para ello se crea una flota inicial, con parámetros de diseño y del sistema de control aleatorios y se deja evolucionar hasta que el sistema alcanza un estado estable durante un cierto tiempo.

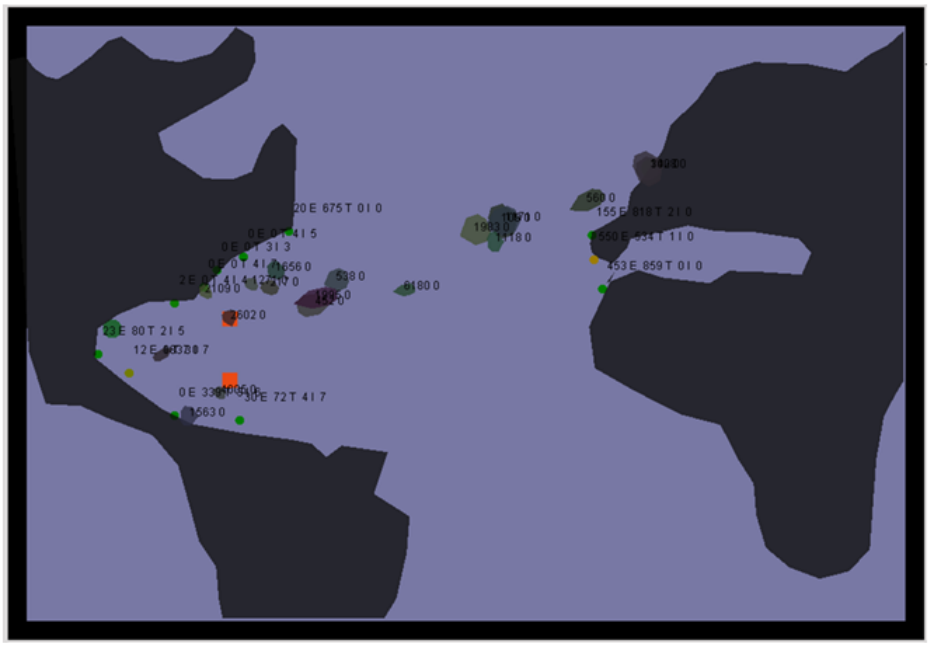


Figura 7.19: Escenario de fletes con dos tipos de puntos de demanda (uno con puntos próximos a la zona de suministro y otro con puntos alejados de la zona de suministro y con una demanda más grande): Generación de especies.

En la figuras 7.21, 7.22 y 7.23 se muestra una la estructura de la población en tres diferentes instantes de esta evolución que representa como se ha ido produciendo la adaptación. Los parámetros de diseño que estamos utilizando para analizar la población son la velocidad y la capacidad. Hemos seleccionado estos dos parámetros pues son los más relevantes en cuanto al comportamiento de un barco, y a su contribución a la tasa de consumo. En dicha figura se muestra, para cada instante fijo de evolución, en el cuadro de la izquierda una representación de todos los barcos de la población asignando a cada barco una posición cuyas coordenadas horizontales y verticales corresponden al valor de sus dos parámetros de velocidad y capacidad. En las gráficas central y derecha se muestran todos los valores de velocidad y capacidad para toda la población ordenados, de modo que resulta sencillo apreciar la distribución de la población en función de estos dos parámetros.

Creador Elemento Barco NEI = 20	
Parámetros de estado	
Carga	(0)
Tiempo Revisión	(autonomía)
Edad	(0)
Recursos	(0)

Creador Punto Demanda NEI=17	
Parámetros descriptivos de elemento	
Cantidad demanda	(10)
Intervalo demanda	(5)

Creador Punto Demanda NEI=3	
Parámetros descriptivos de elemento	
Cantidad demanda	(25)
Intervalo demanda	(5)

Elemento Punto de suministro NEI=2

Elemento Punto de mantenimiento NEI = 2

Figura 7.20: Tabla de creadores para el escenario de fletes. Variación del mercado de demanda.

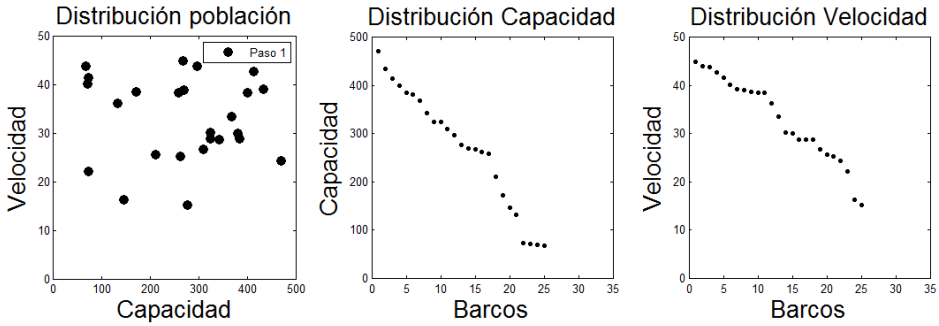


Figura 7.21: Evolución para el mercado de demanda 1. Población inicial.

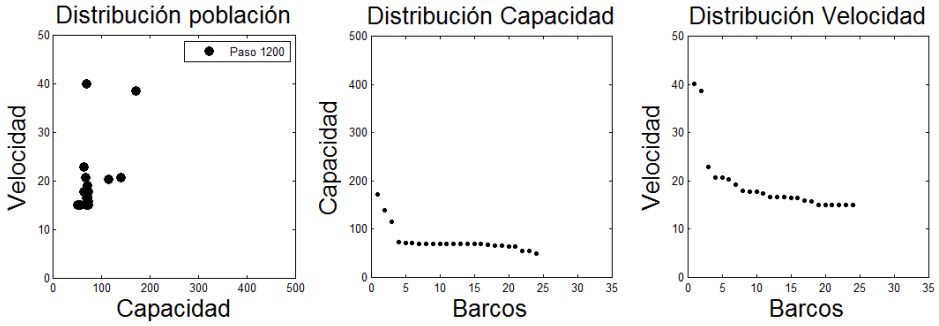


Figura 7.22: Evolución para el mercado de demanda 1. Población intermedia.

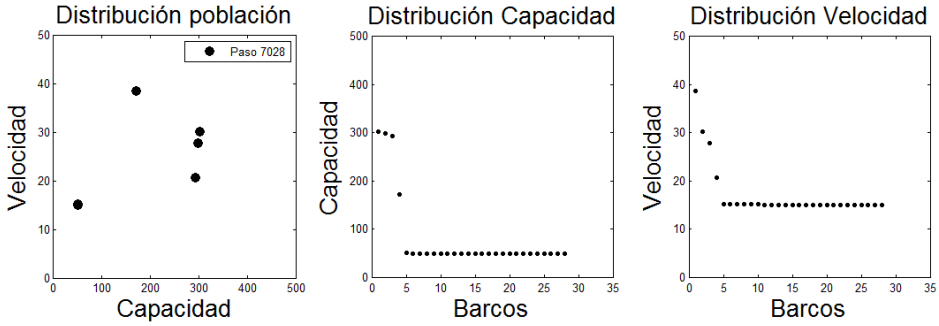


Figura 7.23: Evolución para el mercado de demanda 1. Población final.

Los tres instantes mostrados corresponden al primer, a un paso intermedio y al último paso de evolución. Puede apreciarse en el primer paso de evolución la distribución aleatoria inicial en ambos parámetros, la distribución final con una clara diferenciación entre los barcos que realizan el tráfico entre puntos próximos al punto de suministro y los que realizan largos recorridos, y una distribución transitoria intermedia a mitad del proceso evolutivo. Huelga decir que los barcos especializados en realizar grandes trayectos son aquellos que poseen una mayor capacidad, pues esto permite que la recompensa obtenida pueda compensar el elevado gasto en el que se incurre en un trayecto tan largo. Adicionalmente, el nivel de velocidad entre todos los barcos que operan a poca distancia no es tan relevante como el coste del transporte, pues existe una producción constante y continua de demandas que permite que se reparta adecuadamente dicha demanda entre los barcos. Sin embargo, en el caso de largos recorridos, la demanda en los puntos está más discretizada, y una diferencia de velocidad hace que se pueda perder una entrega. En este caso cada entrega sí es altamente relevante.

En una segunda configuración experimental se han variado las condiciones de demanda para ver cómo responde la estructura de la población. La variación que hemos realizado ha sido la de incrementar de manera significativa la demanda en los puntos lejanos. Para ello se han generado 3 nuevos puntos de demanda alejados de los puntos de suministro. Esto hace que si antes teníamos una generación de demanda de 34 unidades (17 puntos x 2 unidades/punto) por paso de tiempo en la zona cercana y de 15 unidades (3 puntos x 5 unidades/punto) por paso de tiempo en la zona lejana, ahora tendremos una generación de 30 unidades de demanda por paso de tiempo en la zona lejana. Tras permitir al sistema evolucionar durante 5000 pasos de tiempo, la distribución de la población obtenida se muestra en las figuras 7.24, 7.25 y 7.26). El proceso es similar al anterior, pero en este caso, el equilibrio alcanzado se produce, como se puede apreciar, con un porcentaje de barcos de largo recorrido superior al anterior.

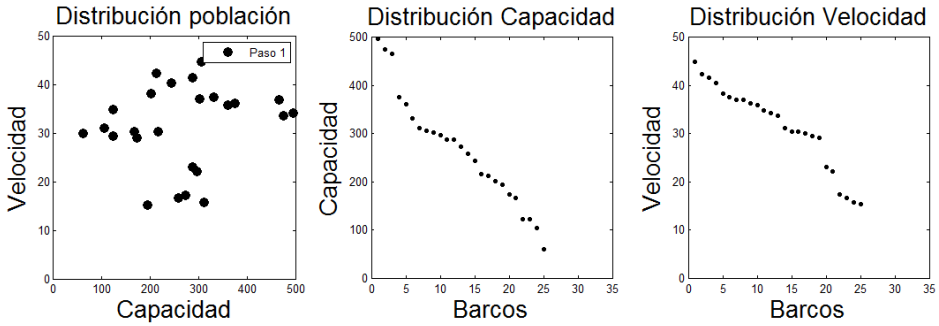


Figura 7.24: Evolución para el mercado de demanda 2. Población inicial.

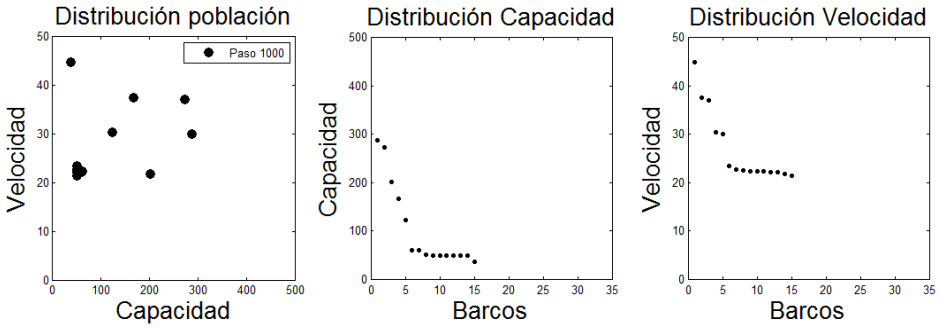


Figura 7.25: Evolución para el mercado de demanda 2. Población intermedia.

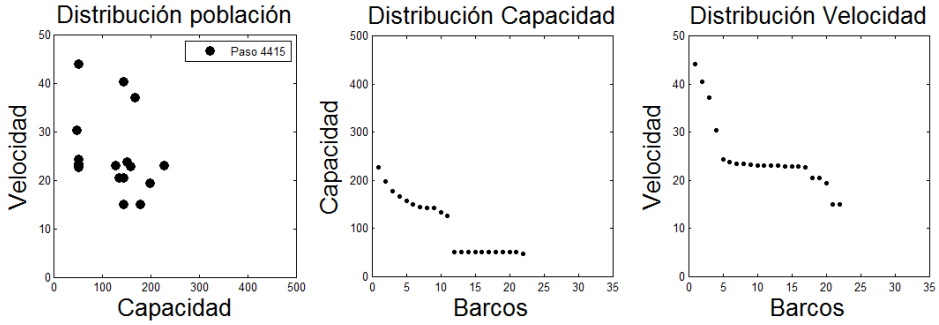


Figura 7.26: Evolución para el mercado de demanda 2. Población final.

Tal y como comprobamos con los resultados mostrados, al introducir en el modelo nuevas tareas bien diferenciadas CeAS es capaz de generar dos especies distintas que comparten los puntos de suministro pero que se encarga de realizar entregas a puntos de demanda cercanos o lejanos. En este experimento no ha sido necesario activar el coeficiente de afinidad para favorecer la especiación debido a la distancia genotípica que implica la realización de una tarea u otro y a la incompatibilidad entre tareas. Este resultado confirma de nuevo la capacidad del CeAS para crear especialización en la población de forma automática.

Tras haber estudiado las variaciones en las condiciones de la función objetivo, vamos a estudiar ahora el cambio en las condiciones del entorno, así como la capacidad de adaptación en tiempo de evolución a estos cambios.

7.5.3. Adaptación a cambios en las condiciones del modelo: variación de los costes asociados al transporte

En este apartado queremos estudiar la capacidad de adaptación de las poblaciones que genera el CeAS. Por un lado, vamos a generar cambios no en la función objetivo, sino en la configuración del modelo, para ver si podemos obtener una solución nueva y estable para estas nuevas condiciones. Pero vamos a realizar estos cambios durante el transcurso de la evolución, de modo que comprobemos que el sistema permite una adaptación en tiempo real a nuevas condiciones. En concreto, el cambio que vamos a realizar simulará el encarecimiento del gasto en combustible de los barcos, y por tanto, repercutirá en un mayor coste relativo del uso de velocidades más altas que suponen un nivel de eficiencia energética más bajo. Para ello, hemos substituido la ecuación 7.11 por la ecuación 7.14 en la que el efecto del aumento de velocidad ha sido duplicado. En la simulación realizaremos esta modificación en el coste del combustible en el paso de tiempo 10.000, y luego dejaremos que siga evolucionando hasta estabilizarse de nuevo.

$$C_{consumo} = 0{,}6 + 0{,}8 \left(\frac{V}{V_{min}} \right) \left(\frac{1}{2} + \frac{V}{2 \cdot V_{min}} \right) \tag{7.14}$$

En las figuras (7.27, 7.29, 7.28 y 7.30) mostramos la estructura de la población aleatoria inicial (figura 7.27), la estructura justo antes de producirse el cambio en el coste de combustible(figura 7.29), el estado transitorio que se genera después del cambio (figura 7.28), y finalmente, la nueva configuración que se alcanza para las nuevas condiciones (figura 7.30)

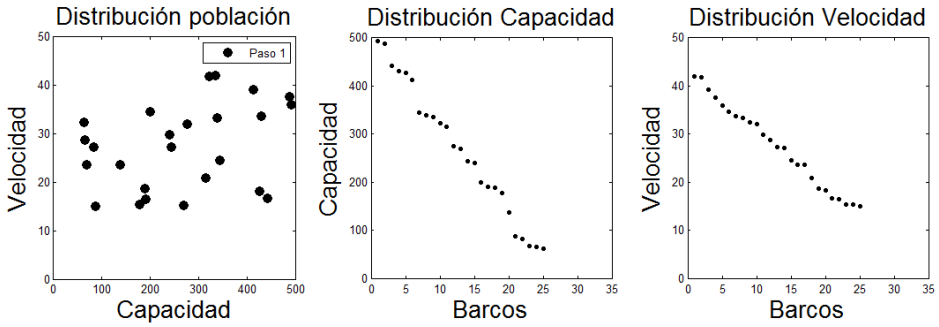


Figura 7.27: Evolución con variación de las condiciones del mercado en tiempo de simulación. Población inicial. Condiciones 1.

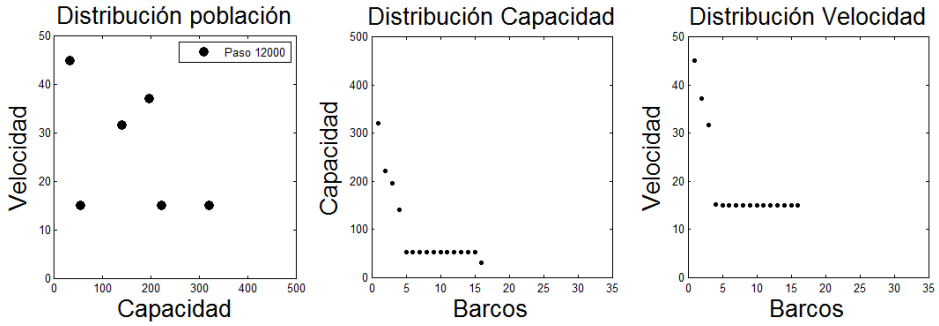


Figura 7.28: Evolución con variación de las condiciones del mercado en tiempo de simulación. Población estable. Condiciones 1.

Tal y como se aprecia en la evolución de la población, la situación en el equilibrio para las condiciones iniciales es equivalente a la que vimos en el apartado anterior, pues las condiciones son las mismas. Sin embargo, tras el cambio de condiciones vemos que se genera un estado

transitorio, tal y como se muestra en la figura 7.28 la población cambia su configuración para adaptarse en tiempo real a los nuevos requerimientos del mercado. Este resultado es muy significativo pues demuestra una de las características perseguidas en el campo de la evolución de poblaciones en sistemas dinámicos que es la adaptabilidad en tiempo real. Esta capacidad de obtener soluciones en problemas dinámicos era una de las más importantes cuando se definió el CeAS, y con este resultado vemos que el resultado obtenido es adecuado.

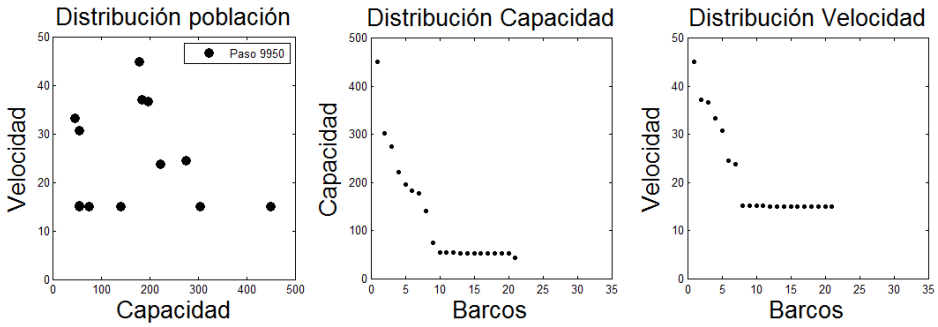


Figura 7.29: Evolución con variación de las condiciones del mercado en tiempo de simulación. Población transitoria. Condiciones 2.

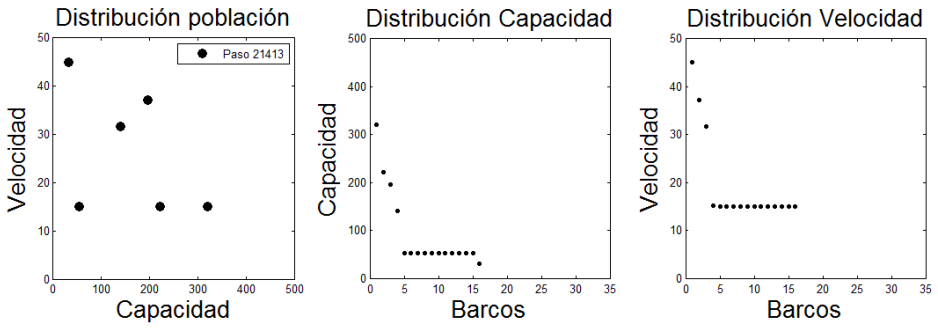


Figura 7.30: Evolución con variación de las condiciones del mercado en tiempo de simulación. Población estable. Condiciones 2.

7.5.4. Estudio de la coevolución en un problema de competencia: interacción entre dos flotas

Finalmente, hemos decidido explotar otra de las características que nos parecen muy interesantes y que es inherente a un algoritmo de coevolución como el CeAS: el tratamiento de un problema de competencia entre especies. Para ello, vamos a introducir en el escenario una flota de barcos 'rival' que poseerá una composición y comportamientos fijos y que competirá con la flota de barcos gestionada por el CeAS.

Lo que vamos a estudiar con este planteamiento es si el algoritmo CeAS es capaz de generar una flota que consiga evolucionar de manera que expulse del mercado a la flota rival, incluso a expensas de una pérdida de eficiencia. Es decir, que consiga hacer a los barcos de la flota rival no rentables y que por tanto estos, siguiendo las reglas que hemos definido en el entorno, cuando su balance económico pasa a ser negativo son eliminados.

Para ello, se genera en el CeAS una flota aleatoria inicial y se simula durante 500 pasos de tiempo. Esta flota puede eliminar a elementos que se queden sin recursos, pero no puede cruzarse y generar nuevos elementos. Este ajuste inicial representa el 'modus operandi' habitual para generar una flota real, el mecanismo más sencillo pero efectivo y que se basa en la experiencia. Es decir, se prueba el funcionamiento en el entorno de un abanico inicial numeroso de barcos con distintos comportamientos y la flota se queda con aquellos que resultan más rentables. En este tiempo, por tanto, la flota elimina a aquellos barcos menos eficientes. Una vez pasada esta selección inicial, la flota obtenida se mantiene fija. En el paso de tiempo 900, es decir 500 pasos después de que desaparezcan la mayoría de los barcos (recordamos que el tiempo de madurez y por tanto de supervivencia para barcos no rentables es de 400 pasos de evolución) se introduce la flota de barcos que serán evolucionados con el CeAS, al igual que en las anteriores configuraciones 20 barcos, y se continúa con la simulación.

Los resultados obtenidos se muestran en las figuras 7.31, 7.32 y 7.33 de las que extraemos los siguientes análisis:

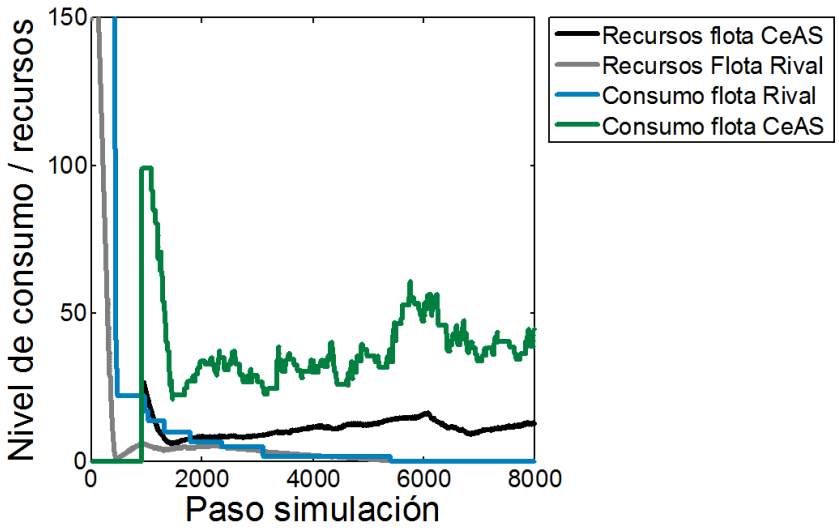


Figura 7.31: Evolución del nivel de consumo y de recursos.

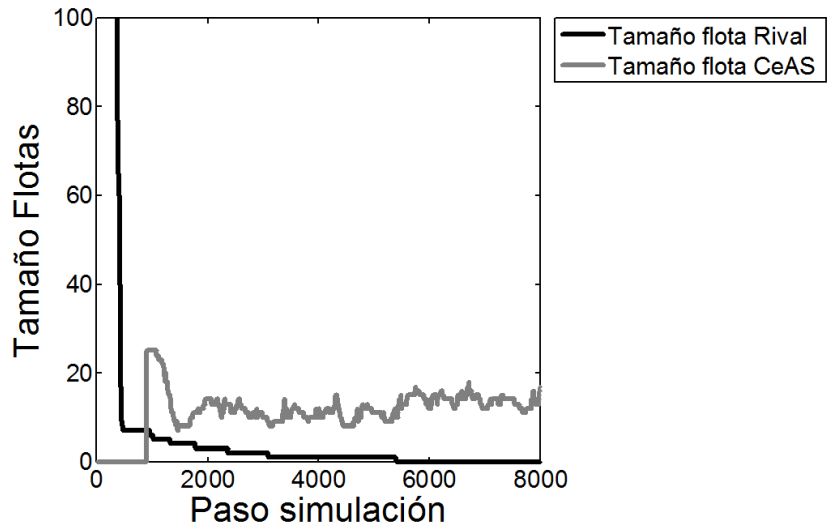


Figura 7.32: Evolución del tamaño de cada una de las flotas.

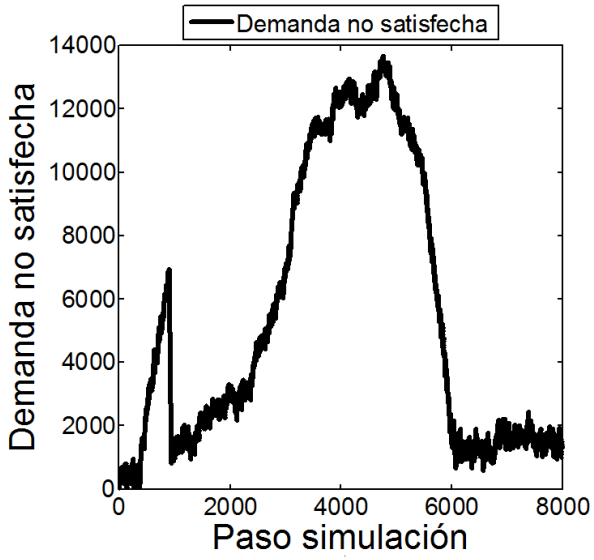


Figura 7.33: Evolución de la demanda no satisfecha.

- Inicialmente vemos que, hasta el paso 400, la población de la flota rival es de 150 barcos, el consumo es muy elevado, y por tanto, la demanda está altamente satisfecha pero a costa de una disminución rápida de los recursos de la flota rival.

- Cuando se alcanza el paso 400, los barcos ineficientes que no han desaparecido ya (pueden morir antes de su madurez si consumen recursos en hacer entregas que les proporcionan menos recompensas que el coste de la descarga) desaparecen. El tamaño de la flota rival y su consumo alcanzan un valor que se mantiene fijo hasta la entrada de la flota CeAS. Durante este período, la flota incrementa gradualmente su nivel de recursos, es decir, la flota es capaz de obtener más recompensas que el nivel de consumo que posee. Este ritmo de crecimiento de los recursos nos da una indicación de la eficiencia de la flota rival. Aún a pesar de generar recursos, no son capaces de satisfacer por completo la demanda, que crece también de forma rápida en este periodo.

- En el paso de tiempo 900 introducimos la flota CeAS. Inicialmente, es una población de 25 elementos que caen rápidamente hasta 10

barcos en el paso de tiempo 1500. Durante este período se produce la coevolución de la flota CeAS para conseguir un mayor nivel de recursos a costa de los recursos de la otra población. Se observa un descenso en el consumo y el tamaño de la flota rival.

- A partir del instante 1500, la población CeAS ha eliminado a los barcos ineficientes y continúa el proceso de coevolución. Gradualmente, los recursos de la flota rival van bajando, y en consecuencia también el número de agentes y su consumo hasta alcanzar el paso 5500 aproximadamente en el que la flota rival desaparece. Durante este tiempo, el criterio objetivo para la flota CeAS fue el de conseguir recursos a costa de mejorar la eficiencia con respecto a la flota rival, y por tanto, la satisfacción de demanda descendió considerablemente, alcanzando un nivel muy alto de requerimientos de demanda.

- Finalmente, tras haber eliminado a la flota rival, el tamaño de población medio de la flota CeAS aumenta ligeramente y se produce una disminución rápida de la demanda insatisfecha.

En resumen, hemos introducido la competición mediante la introducción de una nueva flota rival seleccionada en base a un criterio de eficiencia durante una primera fase inicial de 900 pasos de tiempo. Hemos dejado al algoritmo CeAS que evolucionase la población para quitar del mercado a la flota rival, aún a costa de perder eficiencia absoluta respecto al entorno. Finalmente, y tras haber utilizado una estrategia competitiva, el CeAS cambia su estrategia y optimiza el criterio de partida que es el de satisfacción de la demanda.

Tras este último experimento hemos comprobado la capacidad de esta técnica para llevar a cabo procesos de evolución que involucran cambios complejos en el objetivo de la población como ha sido la introducción de una flota rival dentro del escenario. Por otro lado, esta flota rival va modificándose gradualmente al perder elementos hasta que desaparece por completo, lo cual supone de nuevo una modificación de las condiciones del entorno en tiempo de ejecución. Tal y como nos muestra la gráfica de satisfacción de la demanda, durante el proceso de competencia el objetivo de satisfacción de demanda queda descuidado y una vez que este proceso finaliza esta satisfacción vuelve a descender hasta un valor mínimo.

7.6. Conclusiones

En este último caso de estudio del CeAS en la resolución de un problema dinámico descentralizado, se ha generado un escenario que representa un problema de abastecimiento. En concreto, se modelado un problema de fletes de barcos, donde el objetivo es, además de encontrar la ruta óptima, encontrar la configuración de barcos óptima.

Este ejemplo ha servido para comprobar el funcionamiento del CeAS en un problema típico de ingeniería donde la formulación se realiza en base a criterios económicos. El resultado obtenido resuelve el problema de fletes de forma eficiente, incluso permitiendo la formación automática de especies cuando el planteamiento así lo requiere.

La principal aportación de este ejemplo al estudio del CeAS reside en las pruebas relativas a cambios en el problema, es decir, al dinamismo. Se ha comprobado que esta técnica es capaz de proporcionar soluciones que se adaptan a cambios en las condiciones, como por ejemplo, un cambio en el precio del combustible. Este es un resultado muy relevante para la aplicación en tiempo real del CeAS.

Finalmente, se ha utilizado el CeAS en un problema competitivo, es decir, de nuevo estudiamos la adaptación pero ahora los elementos que cambian son otros agentes. El resultado obtenido ha sido, de nuevo, muy satisfactorio llegando a eliminar a la flota competidora del mercado.

Capítulo 8

Conclusiones y principales aportaciones

En el desarrollo de esta tesis se ha propuesto y se ha estudiado el alcance de la Co-evolución Asíncrona Situada (CeAS) como alternativa a las técnicas de optimización existentes para el tratamiento de problemas dinámicos descentralizados presentes en diversos ámbitos de la ingeniería.

Con este objetivo, se ha realizado un estudio de los tipos de problemas de optimización en ingeniería, sus características más importantes y las deficiencias de otras técnicas de optimización para el tratamiento de algunos de estos problemas. En concreto, para muchos de los problemas que se engloban dentro del campo de la Investigación Operativa. En ellos, existen una serie de características como el dinamismo, el gran número de interacciones entre sus elementos, la ausencia de información global, y otras, que los hacen especialmente complejos. Esta revisión aparece en la primera parte del capítulo 3 de esta memoria y nos ha llevado a concluir la necesidad de afrontar el desarrollo de una nueva técnica de optimización computacional que mejore a las existentes en un tipo particular de problemas de ingeniería: aquellos que son dinámicos y descentralizados.

Como base para diseñar la técnica que se plantea, se utiliza un algoritmo evolutivo open-ended propio de los Sistemas Complejos de Vida Artificial. Esta elección se justifica en el segundo apartado del capítulo 3 de la memoria a partir de una revisión detallada de los principales logros ob-

tenidos en el campo de la Vida Artificial aplicada. Como mecanismo para guiar la evolución del sistema open-ended hacia un objetivo de diseño, se ha analizado con detalle el campo de la Inteligencia Artificial Distribuida, y tras revisar las principales aproximaciones para la obtención automática de comportamientos coordinados en Sistemas Multiagente, nos hemos decidido por la teoría de utilidades privada y global, tal y como se describe en el segundo apartado del capítulo 3 de la memoria.

A partir de este estudio se ha diseñado e implementado computacionalmente la técnica CeAS. En este sentido, se han definido las características y el esquema de funcionamiento del algoritmo y se ha estructurado una metodología de diseño para la transformación de los problemas reales a un escenario evolucionable mediante el CeAS. También se han abordado una serie de consideraciones de análisis para el control y la extracción de soluciones. Todo ello se explica con profundidad en el capítulo 4 de la memoria.

Con el objeto de resolver un problema dinámico descentralizado en CeAS, se requiere simular los escenarios que modelan dicho problema, por lo que se ha desarrollado una herramienta de simulación computacional denominada WaspBed (World-Agent Simulation Platform for Behavior Design), que permite la realización de las tareas de diseño, ejecución y análisis de un modo cómodo y flexible. El diseño del WaspBed se ha basado en una serie plantillas de configuración para la implementación de los modelos de simulación y la adaptación del CeAS para cada problema. El uso de estas plantillas sirve también como procedimiento para el diseñador tanto en la definición de los escenarios como para la adaptación del CeAS a cada uno de ellos. El diseño del Waspbed se ha presentado en el capítulo 4 de esta memoria, mientras que la implementación concreta que se ha realizado aparece detallada en el Apéndice I.

Durante el desarrollo específico del CeAS y en las pruebas realizadas en la resolución de problemas reales, se han ido creando y adaptando diversos operadores. Entre ellos, podemos destacar el operador de cruce bipolar, el mecanismo de control energético mediante la variación de la densidad aparente, la reproducción basada en embriones y la adaptación de las técnicas de selección por torneo a su operación en entornos asíncronos y descentralizados con información local, donde la probabili-

dad de coincidencia espacio-temporal entre agentes es muy baja.

Adicionalmente, y tras el estudio de los distintos ejemplos prácticos que se presentan en esta tesis, se han ido generando una serie de procedimientos de uso frecuente en el análisis e implementación de escenarios de simulación tales como acciones genéricas para la reproducción, optimización de procesos para búsqueda de vecinos, sistemas de control mediante redes neuronales o sistemas de reglas, funciones de evaluación, representaciones para la monitorización de la evolución energética, composición genética de la población, etcétera, que facilitan el uso del WaspBed.

En cuanto a las capacidades del CeAS, se han estudiado los aspectos básicos que se perseguían en el diseño del algoritmo. En primer lugar se ha constatado su validez para la generación de comportamientos orientados a un objetivo dentro de poblaciones de agentes que evolucionan en un sistema abierto. En el estudio de los tres modelos propuestos en los casos prácticos se ha comprobado la validez del algoritmo para resolver de forma distribuida problemas descentralizados, en los cuales existe un conocimiento parcial del entorno y que presentan como características más relevantes la existencia de múltiples interacciones entre los elementos que los definen, y por tanto, de un gran número de estados posibles. También se ha verificado su funcionamiento en problemas con un elevado número de restricciones en el modelo del problema y gran dinamismo del escenario consecuencia de la existencia de un número alto de agentes interactuando, el uso de parámetros estocásticos para la definición del entorno o la introducción de determinadas modificaciones durante la evolución.

En el caso de estudio relacionado con obtención de un algoritmo que proporcione la ruta óptima que se describe en el capítulo 5 de la memoria, cabe destacar de forma particular que ha servido para mostrar las capacidades del CeAS a la hora de modificar los parámetros de diseño de un problema real y poder estudiar de forma sencilla la nueva configuración de soluciones que se alcanza. Esta propiedad analítica del CeAS proviene de su origen en los Sistemas Complejos, donde este tipo de estudios de sensibilidad y estabilidad ante modificaciones de los parámetros son muy comunes. En este caso, además, la evolución en el CeAS está guia-

da por un objetivo mediante las funciones de utilidad, por lo que estas modificaciones siempre llevan a soluciones que cumplen el objetivo en la medida de lo posible.

En el segundo caso de estudio, centrado en un problema de exploración autónomo del entorno, en concreto, en un problema de vigilancia, se ha probado el CeAS en un caso más realista desde el punto de vista de la definición de los agentes, con el objetivo de que las soluciones alcanzadas pudiesen ser directamente trasladables a una flota de robots. Esto implica una mayor complejidad en la definición de los elementos en el Waspbed y un mayor cuidado en el planteamiento de las funciones de utilidad, ya que ahora la eficiencia en la solución es el objetivo básico. Los resultados obtenidos aparecen en el capítulo 6 de la memoria y han sido muy satisfactorios, probando así la idoneidad de la teoría de funciones de utilidad privada y global, y la capacidad del CeAS para resolver problemas reales de ingeniería.

En este caso de estudio también se ha llevado a cabo una comparación entre el CeAS y otras técnicas convencionales para la evolución de una población homogénea y de tamaño fijo en la tarea de vigilancia de un recinto cerrado. Esta comparación ha mostrado que el uso del CeAS, además de sus capacidades inherentes para tratar problemas que no son abordables mediante estas técnicas convencionales como es, por ejemplo, la evolución de poblaciones heterogéneas y/o de tamaño variable y/o en entornos altamente dinámicos, permite una mayor velocidad de convergencia hacia soluciones y unos resultados en términos de eficiencia iguales o mejores que los obtenidos con las otras aproximaciones en problemas en que éstas sí que pueden ser empleadas.

El capítulo 6 de la memoria recoge además el estudio realizado sobre la capacidad del CeAS para obtener soluciones heterogéneas, es decir, para permitir la coexistencia de especies diferentes a nivel genotípico dentro de la población, si esto conlleva una mejora en el objetivo.

En el último caso de estudio que se ha planteado, relativo a un problema de abastecimiento, en este caso con fletes de buques, se ha mostrado la adaptatividad del CeAS mediante el uso de entornos cambiantes, con modificaciones tanto en las condiciones de operación del entorno como

en la función objetivo. Esta propiedad está presente en los tres casos de estudio, pero especialmente en el problema de fletes de buques que se detalla en el capítulo 7 de esta memoria.

Otra de las aportaciones de esta tesis ha sido el desarrollo del método de reproducción basada en embriones, con el objeto de permitir la utilización del CeAS como técnica evolutiva en línea y actuando en tiempo real sobre un número fijo de agentes (como es el caso de robots reales cooperando). Este método permite la realización del proceso evolutivo en entornos asíncronos y distribuidos en términos de la transmisión de información genética con un sesgo hacia individuos mejor adaptados sin implicar la desaparición de los padres en el momento de la procreación. En otras palabras, un individuo puede tener múltiples contactos con transmisión de información genética a lo largo de su vida, pero el resultado no se manifiesta hasta que el individuo desaparece, que es el momento en el que el embrión resultante pasa a estar activo.

Se han generado también, a lo largo del estudio de cada una de las aplicaciones prácticas, una serie de consideraciones metodológicas para el análisis energético y paramétrico relativo al diseño, control y extracción de soluciones de las simulaciones. Consideraciones que van desde el proceso de modelizado descentralizado, el estudio de la utilidad privada y los balances energéticos, hasta los criterios de selección para la reproducción. Estas son de gran utilidad, ponen orden y dan sentido al gran número de parámetros y reglas de interacción que intervienen, primero en los modelos que tratan de representar problemas reales de gran complejidad, y luego en el propio algoritmo CeAS para guiar el proceso evolutivo.

En conclusión, inspirados en las reglas de interacción, generación de dinámicas colectivas, procedimientos de control y técnicas de análisis de los Sistemas Complejos, en el procedimiento de evolución basado en la selección natural estudiado y utilizado por las simulaciones de Vida Artificial, y finalmente, en las técnicas de generación de procedimientos de interacción y la asignación de funciones de utilidad privada con el objetivo de satisfacer una función objetivo propias de la Inteligencia Artificial Distribuida, se ha diseñado el algoritmo de Co-evolución Asíncrona Situada, CeAS. Este algoritmo ha mostrado su eficiencia e idoneidad para la

resolución de ciertos tipos de problemas dinámicos y descentralizados en ingeniería de una manera distribuida eficaz que hemos visto que no permite otro tipo de técnicas evolutivas.

Queda ahora, a partir de este trabajo, una línea de investigación bien estructurada que seguir en cuanto a la validación de esta aproximación en nuevos ámbitos, la creación de operadores diversos que permitan adaptarla a nuevos problemas de una forma más eficiente y la realización de estudios teóricos pormenorizados para determinar la correspondencia entre los parámetros de configuración y los resultados obtenidos, y el ajuste para la obtención de los equilibrios dinámicos deseados.

Con el objeto de realizar un estudio más exhaustivo, formal y controlado, se pretende llevar a cabo una serie de evoluciones sobre problemas modelizados mediante ecuaciones sintéticas que permitan estudiar los efectos de cambios en el tipo de tareas objetivo en el proceso evolutivo. Se estudiará también, mediante el uso de estas pruebas, la capacidad para generar especies en función de la separabilidad de las tareas objetivo o de la existencia de más de dos tareas, de hecho, un número indeterminado.

Se llevarán a cabo estudios con respecto a los efectos de variar el nivel de complejidad de los agentes integrantes de la población para llevar a cabo ciertas tareas. Se estudiará la relación entre nivel de complejidad, tamaño de la población y eficiencia en la resolución del problema.

Se pretende estudiar la aplicación de operadores de cruce basados en técnicas más actuales y que han demostrado ser muy eficaces en un gran número de problemas como el ‘Differential evolution’ (Storn y Price, 1997) o los algoritmos macroevolutivos (Marín y Solé, 1999) para comprobar si existen mejoras en la evolución. También se estudiará la modificación para la creación de nuevos cruces bipolares a partir de estas técnicas con el objeto de permitir la creación de especies en la población.

En cuanto al desarrollo del WaspBed, se pretende seguir generando herramientas que permitan una mejor gestión y análisis de las simulaciones dinámicas, así como implementar nuevos operadores para la evolución que faciliten realizar estudios comparativos y mejorar así los resulta-

dos obtenidos. Además, y con el objeto de ampliar el ámbito de aplicación del sistema, cabría establecer procedimientos para la inclusión de librerías u otras componentes tales como simuladores dinámicos 3D.

Se realizará también un estudio detallado de la transferencia de los resultados obtenidos en simulación a los problemas reales e incluso a la aplicación del CeAS en tiempo real como sistema de control permanente de un Sistema Multiagente.

Por último, se estudiará la aplicación del CeAS en nuevos problemas y en versiones más complejas de los problemas actuales con el objeto de definir el ámbito de aplicación así como para generar nuevas herramientas o procedimientos para tratar otros tipos de problemas.

Apéndice A

Implementación del WaspBed

A continuación pasaremos a explicar en detalle cada una de las clases y las relaciones que hay entre ellas.

A.1. La clase Centralita

Esta clase, junto con la clase Actualizador, representan la centralita del diseño que hemos presentado. Posee las asociaciones con otras clases del sistema Waspbed:

- Con la **clase Actualizador**, representada por la variable *actualizador*, la centralita se encarga de gestionar las funciones de actualización e inicialización de los elementos y eventos. La clase actualizador contiene métodos para realizar el acceso a los ficheros de creación y configuración de Elementos y Eventos y a la *Tabla de Compatibilidades* y para la comprobación de la existencia de nuevos ficheros o de la actualización de las asociaciones de la *Tabla de Compatibilidades*.

- Contiene una lista de rutas a los archivos de configuración existentes en cada momento representada por la variable **vectorArchivosXML**. Su gestión se realiza a través del *Actualizador*.

- Una **HashMap** representada por la variable *tablaTiposElementos* contiene los tipos de elementos y sus identificadores. Todos los elementos del entorno están asociados a un objeto *TipoElemento*. Por

lo tanto, mediante esta lista podemos acceder a todos los elementos del entorno.

- La variable **vectorHilosElemento** contiene un vector con las referencias a las instancias de *HiloElemento* existentes.

- La variable **tablaEventos** contiene otro objeto de tipo *HashMap* con las referencias a los distintos eventos activos durante la ejecución y que extienden la clase *Evento*.

- La relación entre la clase *Centralita* y la clase *ItermediarioInterfaz* se representa mediante la variable **intermediarioInterfaz**. Esta relación sirve de conexión entre la estructura de parámetros gráficos de los elementos y el sistema de representación del Panel del Waspbed.

- La variable de tipo entero **ratioElementosHilo** indica el número de elementos que se asignan como máximo a un hilo y que puede ser modificada por el usuario durante la simulación para mejorar la eficiencia.

Además de las variables anteriores la clase Centralita contiene los siguientes métodos:

- El método **setEntradasPanel** recibe las modificaciones desde el panel de representación de los parámetros que pueden ser controlados en tiempo real por parte del usuario. Estas pueden ser, variaciones en el retardo de los hilos de los elementos o en el hilo del interfaz gráfico, activar la creación de un log del estado actual, activar el inicio de la simulación, etcétera.

- El método **buscarActualizacionesEventos** comprueba mediante el actualizador si existen nuevos eventos para activar. Este método se ejecuta cuando el usuario lo indica a través del panel de representación.

- El método **lanzarActualizacionesEventos** permite que a través del actualizador se activen los nuevos eventos que han sido detectados.

- El método **cambiarRetardoHilos** se ejecuta bajo petición del usuario durante la simulación y en la inicialización del escenario. Este método modifica el parámetro de retardo de cada uno de los hilos que están referenciados en el *vectorHilosElemento*.

- El método **cambiarRatioET** modifica el valor de la variable *ratioElementosHilo*.

- El método **buscarActualizacionesTablaCompatibilidad** se encarga de comprobar mediante el actualizador si existen nuevos archivos de configuración para tablas de compatibilidad para activar.

- El método **lanzarTablaCompatibilidad:** genera una instancia de la clase tabla de compatibilidad basada en el archivo XML de configuración que contiene las asociaciones entre elementos y eventos.

- El método **buscarActualizacionesElementos** comprueba, mediante el actualizador, si existen nuevos elementos para activar o bien nuevas definiciones de tipos de elementos.

- El método **lanzarActualizacionesElementos** permite a la centralita través del actualizador que se activen los nuevos elementos o tipos de elementos.

- El método **lanzarElementosOnline** se utiliza para crear nuevos elementos de un tipo ya existente basados en un creador que se genera durante la simulación a partir de parámetros de otros elementos. Se ha introducido para mejorar la eficiencia de eventos como los de reproducción, que generan variaciones de los elementos existentes. Se gestiona a través del actualizador.

- El método **eliminarElementoOnline** se encarga de eliminar elementos en tiempo de simulación. Esta eliminación se gestiona a través del actualizador.

- El método **lanzarGestores** es el encargado de enviar las referencias de los eventos y elementos existentes a dos clases estáticas cuya función es permitir el acceso desde cualquier evento a las listas de elementos, de otros eventos o de eventos herramienta (*EvTool*).

A.2. La clase Actualizador

Dentro del módulo centralita, esta clase se encarga de ejecutar las tareas de actualización. En ella encontramos las siguientes variables:

- La variable **centralitaOrigen** es una referencia a la centralita para poder acceder a sus métodos y a las variables públicas. El actualizador la recibe en su creador.
- El vector **almacenIntercambioElementos** representa una variable en la que se almacenan referencias a los elementos que han sido creados pero aún no han sido asignados a un hilo de ejecución. Funciona como una cola FIFO para el proceso de creación y activación de los elementos.

Los métodos más relevantes de esta clase son:

- El método **buscarActualizacionesElementos** es el encargado de buscar, en la ruta que recibe como parámetro, si existen nuevos archivos *XML* de creación de elementos o Tipos de Elementos y devuelve un vector con las rutas de los nuevos archivos encontrados. Es la centralita la que se encarga de llamar a este método.
- El método **buscarActualizacionesTablaComp** se encarga de buscar en la ruta correspondiente el archivo *XML* de la tabla de compatibilidad, o bien se sustituye la tabla de compatibilidad existente por la que genera un nuevo archivo que haya sustituido al anterior. Es la centralita la que llama a este método.
- El método **addElementoAlmacenIntercambio** añade un elemento al almacén de intercambio de elementos. Se llama tras crear un elemento.
- El método **sacarElementoAlmacenIntercambio** elimina un elemento del almacén de intercambio. Este método y el anterior están sincronizados para evitar conflictos de acceso entre hilos.
- En el método **lanzarTablaCompatibilidad** se lee la tabla de compatibilidad cargada mediante la clase *LectorXML* y se almacenan las asociaciones en el objeto Tabla de Compatibilidad.

- En el método **cambiarRatioThreadsElemento** se reasignan los elementos dentro de los hilos para adaptarse al nuevo número de elementos por hilo. Si es necesario se crean hilos nuevos. Se ejecuta bajo petición de la centralita.
- El método **buscarActualizacionesEventos** recorre la carpeta donde se almacenan los eventos y busca los archivos que contengan la cadena de caracteres 'Evento'. Devuelve una lista con las rutas de los nuevos eventos.
- En el método **lanzarActualizacionesEventos** se genera una nueva clase que extiende a la clase evento con el nombre del fichero '.class' que la define. Se añaden también los nuevos eventos a la tabla de eventos de la centralita.
- El método **creacionElementosOnline** se encarga de crear nuevos elementos de algún Tipo de Elemento ya existente una vez que se ha generado un creador en la simulación. Estos elementos se añaden al almacén de intercambio, se reasignan entre los hilos existentes teniendo en cuenta el ratio de Elementos por hilo actual y finalmente, se envían al **IntermediarioInterfaz** para que extraiga los parámetros gráficos y los incorpore a la tabla de representación del panel.
- En el método **destruccionElementoOnline** se elimina la referencia a ese elemento del vector de elementos del Tipo de Elemento correspondiente. Además, se elimina del hilo al que está asociado y se reasignan los hilos si es necesario y, finalmente, se elimina de la tabla de parámetros gráficos del panel de representación.
- El método **buscarActualizacionesEventos** recorre la carpeta donde se almacenan los creadores de elementos y los tipos de elementos y si no están en la lista de archivos XML, se añaden.
- El método **lanzarActualizacionesElementos** selecciona, de entre los nuevos archivos de configuración *XML*, aquellos que contienen la cadena de caracteres 'tipo' y los que contienen la cadena de caracteres 'creador'. En el caso de la generación de un nuevo tipo, la clase *lectorXML* genera un objeto *TipoElemento* a partir de la definición del archivo *XML* y este *TipoElemento* se añade en la *tablaTiposElementos* de la centralita. En el caso de un archivo *XML* correspondiente a

un creador, el *lectorXML* genera un objeto de la clase *creadorDeElementos* a partir del archivo de configuración. Este creador se le pasa al *TipoDeElemento* correspondiente y este genera un grupo de elementos mediante el método *crearGrupoElementos*. Este grupo de elementos se pasa al *almacenIntercambioElementos* y se reasignan los elementos antiguos y los nuevos en los distintos hilos de ejecución. Finalmente se pasa al *intermediarioInterfaz* para agregar nuevos parámetros gráficos al panel de representación.

A.3. La clase TipoElemento

La clase **TipoElemento** es la representación de un conjunto de elementos que cumplen una serie de características comunes. Estas características comunes son los parámetros descriptivos de tipo de elemento y la representación gráfica. En esta clase se almacenan estas características comunes a todos los elementos de un mismo tipo y un vector con todos los elementos existentes que pertenecen e ese tipo. Sus métodos gestionan la lista con las referencias a los elementos activos además de crear o destruir elementos. Sus variables más representativas son las siguientes:

- La variable **centralitaOrigen** es una referencia a la centralita para poder acceder a sus métodos y a las variables públicas. Esta variable se recibe en el constructor de la clase.
- La variable **nombreTipo** es una cadena de texto o variable de tipo String que contiene el nombre del Tipo de Elemento.
- La variable **tablaPDT** es una tabla que contiene los valores de los parámetros descriptivos del tipo de elemento.
- La variable **tablaPDEdef** es una tabla que contiene los valores por defecto de la definición de los parámetros descriptivos de elemento para todos los elementos de este tipo: rango de valores, rango inicialización, etc.
- La variable **tablaPEdef** es una tabla que contiene los valores por defecto de la definición de los parámetros de estado para los ele-

mentos de este tipo. Si el creador no especifica otra cosa, se toman estas tablas para generar un grupo de elementos.

- La variable **tablaPG** es la tabla en la que se almacenan los parámetros gráficos. Estos parámetros son el nombre de la figura que los representa (punto, línea, polígono, círculo, grafo, etcétera) y una lista con cadenas de caracteres que especifican una expresión matemática con el valor de las coordenadas de la figura en función de los parámetros del elemento.
- La variable **grupoElementos** es un vector de elementos donde se almacenan las referencias a los elementos en activo pertenecientes a ese grupo.

Además de las variables anteriores, el comportamiento de esta clase está representado en los siguientes métodos:

- En primer lugar una serie de métodos para enviar o recibir los valores asociados al *TipoDeElemento*, definiciones de parámetros o elementos, que no entraremos a describir en detalle.
- El método **crearElemento** es el encargado de generar una nueva instancia de un elemento y se ocupa de ejecutar los métodos de inicialización de parámetros de un elemento utilizando un *CreadorDeElementos* que se recibe como parámetro de entrada del método. Se llama desde el método del actualizador *CreacionElementosOnline* y desde *lanzarActualizacionesElementos*.
- El método **destruirElemento** se encarga de eliminar la referencia del vector *grupoElementos*.

A.4. La clase Elemento

Es la clase encargada de almacenar los parámetros asociados a cada uno de los elementos, los inicializa y modifica, y posee los métodos para llamar a los eventos asociados a cada elemento. Las variables de esta clase son las siguientes:

- La variable **tablaPE** es una tabla Hash en la que se almacenan los nombre y valores de los parámetros de estado.

- La **variabletablaPDE** es una tabla Hash en la que se almacenan los nombres y valores de los parámetros descriptivos de elemento.
- La variable **tablaPG** es un vector que contiene pares indentificador-coordenadas que definen los elementos gráficos que representan al elemento y sus coordenadas. El identificador es una cadena de texto que indica qué tipo de figura que debe mostrar el interfaz gráfico y las coordenadas son cadenas de texto que contienen expresiones matemáticas en las que intervienen valores constantes, variables asociadas a los parámetros del elemento y funciones matemáticas convencionales.
- La variable **tipoOrigen** es una referencia al *TipoElemento* al que pertenece.
- La variable **hiloAsociado** es una referencia al *hiloElemento* en el que se ejecuta el elemento.

Los principales métodos de esta clase son los siguientes:

- El método **iniciarParametros** que se encarga, a partir de las definiciones del *Tipo de Elemento* y del creador que haya generado este elemento, de crear los parámetros asociados al elemento teniendo en cuenta el rango permitido para cada valor y el rango de valores de inicialización. Si el diseñador requiere un valor fijo para un parámetro, debe fijar en el creador tanto el máximo como el mínimo del rango de inicialización de un parámetro con ese valor.
- El método **iniciarParametrosGraficos** a partir de la *tablaPG*, que contiene el tipo de elemento, genera un array de cadenas de texto que contienen las expresiones matemáticas que definen cada coordenada para cada figura de las que contiene la tabla.
- El método **modificarParametro** recibe el valor nuevo propuesto para un parámetro y si está dentro del rango permitido para ese parámetro modifica el valor actual.
- El método reaccionar llama a la clase **TablaCompatibilidades** y obtiene una lista con las reacciones asociadas a un Elemento. A continuación, recorre esa lista y llama al método **lanzarEvento** de la clase estática *GestorEventos* para cada una de las reacciones.

A.5. La clase Parámetro

Es la clase que representa las variables de cada uno de los elementos. Cada uno de los parámetros contiene uno o más valores además de los rangos de funcionamiento y los rangos de inicialización. El número de valores de cada parámetro se especifican en la definición del Tipo de Elemento. Cada uno de los valores se almacena en un objeto del tipo *DatosParametro* y estos a su vez se almacenan en un vector. Posee los métodos para modificar los valores de los parámetros.

A.6. La clase DatosParametro

Es la clase que representa cada uno de los valores que posee un parámetro, además del valor real del parámetro contiene el máximo valor permitido, el mínimo valor permitido, el máximo valor de inicialización y el mínimo valor de inicialización. Contiene también el paso de discretización, que indica si los valores son continuos o si, por el contrario, están discretizados ($valor\_real = k * paso$).

A.7. La clase Evento

Esta clase es de la que heredan todos los Eventos. Solamente contiene un método llamado *ejecucion* que recibe como variables de entrada: el paso de tiempo actual, los parámetros de llamada (parámetros que genera el sistema de control para modificar al evento) y una referencia al elemento que lo ha llamado.

A.8. La clase Evento Herramienta (EvTool)

Es una clase abstracta que posee un único método estático para su ejecución. sus parámetros de entrada no están predefinidos, serán todos aquellos necesarios para el uso del Evento Herramienta. Al igual que en el caso de la clase evento, *EvTool* es extendida por todos los eventos herramienta que se generen en Waspbed.

A.9. La clase TablaCompatibilidades

Es otra clase estática, pues requiere poder ser llamada desde cualquier elemento y estos pueden generarse en tiempo de ejecución. Se encarga de almacenar la lista de asociaciones entre eventos y tipos de elementos, y devolver una lista de referencias a eventos cuando se le solicite la compatibilidad de un tipo de elemento determinado. La tabla se llama en la acción reaccionar de los elementos y la genera la clase *LectorXML* a partir del archivo de configuración correspondiente.

A.10. La clase CreadorDeElementos

Esta clase la crea el *LectorXML* a partir de un archivo de configuración de creación de un grupo de elementos nuevos de un tipo ya existente. La clase contiene todos objetos necesarios para crear un nuevo grupo de elementos. Para crear elementos son necesarias las siguientes variables:

- La variable **nombretipo** que contiene el nombre del tipo de los objetos que se van a crear.
- La variable **numerodeelementos** que indica cuántos elementos se van a crear con la configuración especificada en el creador.
- La variable **tablaPDEcreador** representa una tabla en la que se almacenan el nombre de cada parámetro descriptivo de elemento que requiere una inicialización diferente a la inicialización por defecto que genera el tipo de elemento correspondiente, y una variable *parametro* que contiene el nuevo rango de inicialización.
- La variable **tablaPEcreador** que representa una tabla en la que se almacenan el nombre de cada parámetro de estado que requiere una inicialización diferente a la inicialización por defecto que genera el tipo de elemento correspondiente, y una variable *parametro* que contiene el nuevo rango de inicialización.

Los métodos que implementa son únicamente los necesarios para permitir el acceso a las variables privadas por lo que no serán expuestos en este trabajo.

A.11. La clase HiloElemento

Esta es otra clase fundamental para el funcionamiento de la aplicación. Implementa el interfaz Runnable de JAVA para crear un objeto de la clase Thread, que se activa y ejecuta el método *run*, que requiere ser redefinido, después de haber activado el hilo (Thread.start) y mientras este no sea detenido (Thread.stop). En cada uno de los hilos de ejecución se gestionan las llamadas a los eventos de los elementos asociados y las asociaciones entre hilos y elementos. La activación del Thread asociado se realiza ya en el creador de la clase *hiloElemento*. Esta clase contiene las siguientes variables:

- La variable **retardo** representa el número de milisegundos de retardo que se aplican durante la ejecución de cada iteración del hilo, para regular el ratio entre tiempo de cálculo de simulación y tiempo dedicado a la representación gráfica.
- La variable **ratioHiloElemento** es el número de elementos que se asocian a cada hilo. En función de este valor, los hilos accederán al almacén de intercambio de elementos para añadir o eliminar elementos asociados.
- La variable **elementosAsociados** es un vector que contiene las referencias a los elementos que están actualmente asociados a este hilo.
- La variable **actualizadorOrigen:** referencia al actualizador que se requiere para acceder al almacén de intercambio de elementos.

Esta clase implementa los siguientes métodos:

- El método **añadirElemento** añade el elemento recibido a la lista de elementos asociados para que sea incluido en los procesos de ejecución.
- En el método **eliminarElemento** se elimina un elemento de la lista de elementos asociados. La forma de realizarlo requiere de ciertas consideraciones debido a la sincronización entre hilos. Pueden surgir conflictos al añadir o eliminar elementos durante la simulación

cuando se realizan accesos desde otros hilos de ejecución o desde el hilo de la ventana gráfica. Para solucionar esto se crean una serie de vectores auxiliares para gestionar los elementos eliminados o propuestos para incluir en los elementos asociados.

- El método **run** redefine el run del interfaz Runnable y es ejecutado por un hilo distinto al hilo general de la aplicación. Con esto conseguimos lanzar los eventos de un grupo de elementos en un hilo de ejecución separado. Dentro del método creamos un bucle que se repite indefinidamente, hasta que sea interrumpido por el usuario, y que recorre el vector de elementos asociados realizando las siguientes tareas: si el número de elementos fijados por *ratiohiloelemento* es mayor que el número de elementos asociados, se accede al almacén de intercambio de elementos del actualizador y si hay algún elemento disponible se asocia a el hilo actual. Si el número de elementos asociados es mayor que el requerido se mueven estos elementos al almacén de intercambio de elementos. Finalmente se ejecuta el método *reaccionar* del elemento actual si no ha sido eliminado.

A.12. La clase LectorXML

Esta clase es la que se encarga de hacer la transformación entre los archivos *XML* de configuración y los objetos *TipoElemento*, *CreadorDeElementos* y *TablaDeCompatibilidad*. No posee ninguna variable asociada a la clase y los tres métodos más representativos son los siguientes:

- El método **leerDefinicionTipo** recibe la ruta del archivo de configuración de la definición de un Tipo de Elemento y analiza la estructura del archivo *XML* para extraer los parámetros descriptivos del tipo de elemento, y la definición por defecto de los parámetros descriptivos de elemento y los parámetros de estado. Devuelve un objeto *TipoDeElemento*.

- El método **leerCreadorTipo** recibe la ruta del archivo de creación de un grupo de elementos y analiza la estructura del archivo *XML* para extraer el tipo de elemento asociado, el número de elementos que se crearán y los valores de inicialización de los parámetros de estado y

parámetros descriptivos de elemento, si estos difieren de los valores por defecto. Este método devuelve un objeto *CreadorDeElementos*.

- El método **leerTablaCompatibilidad** recibe la ruta del archivo de configuración de la *Tabla de Compatibilidad* y analiza la estructura del archivo *XML* para extraer las asociaciones *Elemento-Eventos*. Devuelve un vector con los identificadores de los elementos y sus eventos asociados que luego el actualizador utilizará para añadir asociaciones a la *Tabla de Compatibilidad* actual.

A.13. La clase IntermediarioInterfaz

Esta clase es la que se encarga de realizar la conexión entre la ventana de la representación gráfica y la ejecución de eventos y actualización de los elementos e independiza estas dos partes, representación y actualización. Con el uso de un intermediario permitimos la incorporación de otros módulos de representación sin la modificación de la estructura de tratamiento y gestión de modelos del Waspbed. Sí sería necesario generar un nuevo intermediario para cada nuevo interfaz gráfico. La ventana de representación es la encargada de generar el intermediario en su creador. Esta clase contiene las siguientes variables:

- La variable **ventanaOrigen** que es una referencia a la ventana de representación a la que está asignado el intermediario.

- La variable **centralita** que es una referencia a la centralita con la que conecta al interfaz gráfico.

Además de las variables anteriores, esta clase implementa el comportamiento de los siguientes métodos:

- El método **elementosToTablaGrafica** recibe un array de elementos y genera una tabla gráfica intepretable por el interfaz gráfico con los parámetros gráficos de cada elemento. La tabla gráfica contiene un identificador de cada figura y un *ArrayCoordenada* que contiene cada una de las coordenadas necesarias para representar una figura concreta mediante un objeto de la clase *ExpresionVariable*.

- El método **recibirElementosNuevos** recibe un vector con un conjunto de nuevos elementos para añadir a la representación. Estos elementos se transforman en una tabla gráfica y se envían a la ventana asociada para añadir a la tabla gráfica existente.
- En el método **eleminarElemento** se recibe un elemento cuya representación va a ser eliminada del panel de representación. Se identifican las referencias de *ArrayCoordenada* que corresponden a ese objeto y se notifica a la ventana que las elimine de la tabla gráfica.

A.14. La clase ArrayCoordenadas y ExpresionVariable

La clase *ArrayCoordenadas* contiene una lista con objetos ExpresiónVariable que se corresponden con las coordenadas de una figura de la tabla de parámetros gráficos de un elemento.

La clase ExpresiónVariable se encarga de interpretar la expresión matemática que el diseñador ha introducido para cada coordenada de una figura y devolver un valor en función de los valores actuales de los parámetros de los que depende. Para ello almacena las referencias a los DatosParametro que intervienen en cada coordenada y la cadena de texto correspondiente a la expresión matemática y utiliza la librería *MathEvaluator* para evaluar la expresión una vez que se han obtenido los valores reales de cada parámetro.

A.15. La clase VentanaWB y Ventana

La clase VentanaWB es una clase abstracta que define los métodos para introducir o eliminar nuevas figuras en la representación (ver figura A.1) gráfica y para pintar el panel de representación y, además, define la referencia al intermediario con la centralita y al hilo asociado. En Waspbed hemos generado la clase Ventana que extiende la clase VentanaWB.

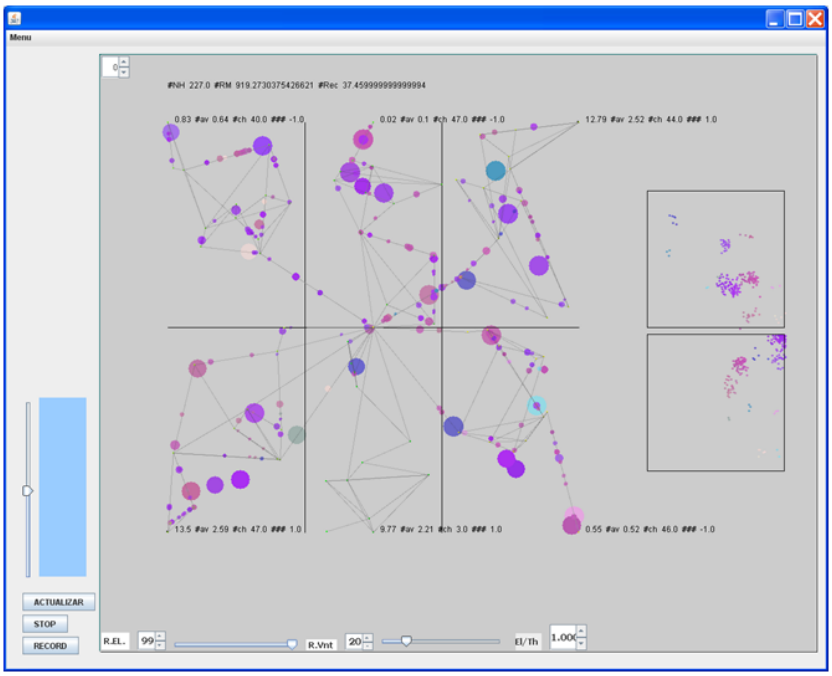


Figura A.1: Ventana Wasped 1.

La clase Ventana contiene las siguientes variables:

- La variable **intermediarioInterfaz** contiene la referencia al intermediario que instancia Ventana.
- La variable **hiloVentana** contiene la referencia al hilo que ejecuta los métodos de ventana. En el creador se instancia el hilo y se le pasa la referencia a la ventana que lo instancia.
- La variable **panelRepresentacion** contiene una referencia a la clase *PanelRepresentacion* que extiende un *JPanel* y representa la zona del interfaz gráfico en la que se dibujan las figuras de la simulación.

La clase Ventana contiene, además, los siguientes métodos:

- El método **pintarPanel** se encarga de llamar al método *repaint* del panel de representación para que se actualice.

- El método **actualizarTablaGrafica** se encarga de enviar al panel de representación una tabla con nuevas figuras para representar.

- El método **reducirTablaGrafica** envía al panel de representación una tabla con las figuras que se van a eliminar porque han desaparecido los elementos a los que estaban asociados.

A.16. La clase PanelRepresentacion

Esta clase se encarga de representar los elementos en un JPanel contenido en la ventana del interfaz. Para el correcto funcionamiento de esta clase son necesarias las siguientes variables:

- La variable **tablagrafica** que contiene una tabla con elementos *ArrayCoordenadas*, que están asociados a los *DatosParametro* correspondientes en cada elemento.

Además, también son necesarios los siguientes métodos:

- El método **pintarTabla** que recorre la *tablagrafica* y utiliza la clase *Graphics* de JAVA para dibujar las figuras en el panel. Una vez que ha recorrido la tabla, elimina las figuras desactivadas.

- El método **aumentarTabla** inserta nuevas entradas de *ArrayCoordenadas* en la tabla.

- El método **reducirTabla** elimina entradas de la tabla. Para evitar problemas de sincronización, los *ArrayCoordenadas* poseen una variable de activación que indica que una figura está activa o inactiva, y cuando se finaliza el proceso de pintado, se realiza esta eliminación. Este método y el de aumentar la tabla son ejecutados por el hilo de la centralita, el hilo principal de la aplicación, y es por eso que pueden surgir conflictos con el hilo de la ventana que solo se encarga de pintar.

A.17. La clase hiloVentana

Esta clase representa un hilo que se activa cuando se construye una ventana y se encarga de ejecutar repetidamente el método pintarPanel de la ventana. Contiene la siguiente variable:

- La variable **retardo** indica los milisegundos de espera entre cada actualización de la representación.

Posee, además del método *run* asociado al hilo, un único método:

- El método **run** posee un bucle infinito que ejecuta el método *pintarPanel* y se pone en espera durante un tiempo fijado por el retardo.
- El método **cambiarRetardo** varía el retardo a petición de la centralita.

A.18. GestorEventos y GestorElementos

Son dos clases estáticas, que se lanzan al comienzo de la aplicación y que poseen un listado de los Elementos y Eventos disponbiles para ser llamados en la inicialización o durante la simulación por cualquier Evento. Su único método devuelve una referencia a un Elemento o Evento asignado a un identificador.

A.19. Los archivos de configuración XML

Estos archivos, contienen la definición de la Tabla de Compatibilidad, Tipos de elementos o Creadores de grupos de elementos para Tipos ya existentes. Ver ejemplo en la figura A.2.

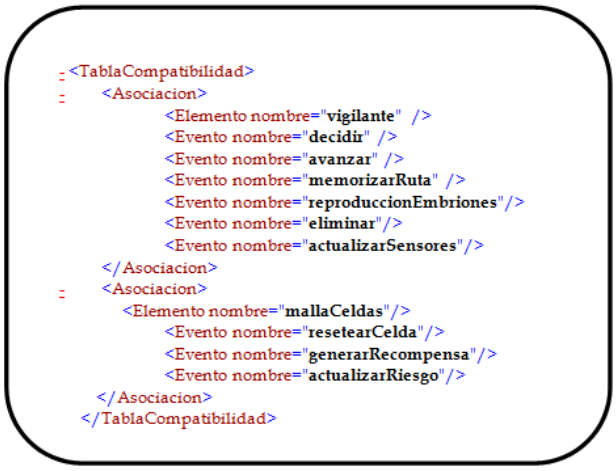


Figura A.2: Asociaciones Elemento-Evento.

Bibliografía

Abdelrazik, M. (1993). Graph-based model for implementing sequential circuits. volume 4, pages 2725–2728.

Adami, C. and Brown, C. T. (1994a). Evolutionary learning in the 2d artificial life system 'avida'. *'Artificial Life IV' Proceedings*.

Adami, C. and Brown, T. C. (1994b). Evolutionary learning in the 2d artificial life system.

Agogino, A. and Tumer, K. (2008). Efficient evaluation functions for evolving coordination. *Evolutionary Computation*, **16**(2), 257–288.

Ahn, Y. K., Song, J. D., and Yang, B. . (2003). Optimal design of engine mount using an artificial life algorithm. *Journal of Sound and Vibration*, **261**(2), 309–328.

Anderson, J., Tanner, B., and Wegner, R. (2002). Peer reinforcement in homogeneous and heterogeneous multiagent learning. *Proceedings of the IASTED International Conference on Artificial Intelligence and Soft Computing (ASC2002)*.

Antsaklis, P., Koutsoukos, X., and Zaytoon, J. (1998). On hybrid control of complex systems: A survey. *Journal Europeen des Systemes Automatises*, **32**(9-10), 1023–1045.

Arthur, W. B. (1994). On the evolution of complexity. *Complexity: Metaphors, Models, and Reality*, **19**, 65–81.

Atanasova, T., Nern, H.-J., Hamalainen, M., and Eldin, H.Ñ. (2000). Distributed heterogeneous knowledge data base for control system design: multiagent development and support. pages 363–368.

Baker, A. D., Parunak, H. V. D., and Erol, E. (1997a). Manufacturing over the internet and into your living room: Perspectives from the aaria project. *IEEE Internet Computing*.

Baker, A. D., Parunak, H. V. D., and Erol, E. (1997b). Manufacturing over the internet and into your living room: Perspectives from the aaria project. *IEEE Internet Computing*.

Batalin, M. A. and Sukhatme, G. S. (2004). Spreading out: A local approach to multi-robot coverage. *Proc. of 6th International Symposium on Distributed Autonomous Robotic Systems, Fukuoka, Japan, 2002*, pages 373–382.

Baum, E. B. (1996). Toward a model of mind as a laissez-faire economy of idiots, extended abstract. *Proc. 13th ICML '96*, pages 28–36.

Baum, E. B. and Durdanovic, I. (1999). Toward a model of mind as an economy of agent. *Machine Learning*, **35**(2), 155–185.

Bedau, M. A. (1998). Four puzzles about life. *Artificial Life*, **4**(2), 125–140.

Bedau, M. A. (2003). Artificial life: Organization, adaptation and complexity from the bottom up. *Trends in Cognitive Sciences*, **7**(11), 505–512.

Bellas, F., Becerra, J. A., and Duro, R. J. (2008). *Internal and external memory in neuroevolution for learning in non-stationary problems*, volume 5040 LNAI.

Beni, G. and Wang, J. (1989). Swarm intelligence. *Proceedings of the Seventh Annual Meeting of the Robotics Society of Japan*, pages 425–428.

Bernardine Dias, M., Zlot, R., Kalra, N., and Stentz, A. (2006). Market-based multirobot coordination: A survey and analysis. *Proceedings of the IEEE*, **94**(7), 1257–1270.

Berry, A. and Vamplew, P. (2003). A simplified artificial life model for multi-objective optimisation: A preliminary report. *The Congress on Evolutionary Computation (CEC)*.

Bianco, R. and Nolfi, S. (2004). Toward open-ended evolutionary robotics: Evolving elementary robotic units able to self-assemble and self-reproduce. *Connection Science*, **16**(4), 227–248.

Birge, J. R. (1997). Stochastic programming computation and applications. *INFORMS Journal on Computing*, **9**(2), 111–133.

Blume, L. E. and Easley, D. (2002). Optimality and natural selection in markets. *Journal of Economic Theory*, **107**(1), 95–135.

Boccara, N. (2003). *Modeling Complex Systems (Graduate Texts in Contemporary Physics)*. Springer.

Bonabeau, E. W. and Theraulaz, G. (1991). Why do we need artificial life? *Artificial Life, an Overview*, pages 303–335.

Borkar, V. S., Jain, S., and Rangarajan, G. (1998a). Collective behaviour and diversity in economic communities. some insights from an evolutionary game. *Econophysics*.

Borkar, V. S., Jain, S., and Rangarajan, G. (1998b). Dynamics of individual specialization and global diversification in communities. *Complexity*, **3**, 50–56.

Boutilier, C. (1996a). Learning conventions in multiagent stochastic domains using likelihood estimates. *Proc. 12th Conf. on Uncertainty in AI.*, pages 106–114.

Boutilier, C. (1996b). Planning, learning and coordination in multiagent decision processes. *Proceedings of the Sixth Conference on Theoretical Aspects of Rationality and Knowledge*, pages 195–210.

Bowles, S. and Gintis, H. (2004). The evolution of strong reciprocity: Cooperation in heterogeneous populations. *Theoretical population biology*, **65**(1), 17–28.

Box, G. E. P. and Chanmugam, J. (1962). Adaptive optimization of continuous processes. *Industria & Engineering Chemistry Fundamentals*, **1**(1), 2–16.

Bradshaw, J. (1997). Software agents. *AAAI Press/MIT Press*, pages 3–49.

Branke, J. (2001). Evolutionary optimization in dynamic environments. *Genetic Algorithms and Evolutionary Computation*, **3**.

Bronson, R. and Naadimuthu, G. (1997). *Schaum's Outline of Operations Research*. 8th edition.

Brown, A. (2002). Web-based plant design tools are ready, but are engineers? *CEP*, **98**(6), 12–13.

Burmeister, B., Haddadi, A., and Matylis, G. (1997). Application of multiagent systems in traffic and transportation. *IEE Proceedings: Software*, **144**(1), 51–60.

Buyya, R., Abramson, D., Giddy, J., and Stockinger, H. (2002). Economic models for resource management and scheduling in grid computing. *Concurrency Computation Practice and Experience*, **14**(13-15), 1507–1542.

Chades, I., Scherrer, B., and Charpillet, F. (2003). Planning cooperative homogeneous multiagent systems using markov decision processes. *Proceedings of the 5th International Conference on Enterprise Information Systems (ICEIS 2003)*, pages 426–429.

Chantemargue, F. and Hirsbrunner, B. (1999). A collective robotics application based on emergence and self-organization. *Proc. of Fifth Int. Conference for Young Computer Scientists (ICYCS'99), 1999*, pages 1–8.

Chapelle, J., Simonin, O., and Ferber, J. (2002). How situated agents can learn to cooperate by monitoring their neighbors' satisfaction. *ECAIO?2*, pages 68–72.

Chen, X. S., Lim, M. H., and Ong, Y. S. (2007). An ant colony system algorithm for path planning in sparse graphs. pages 31–36.

Chung, K. and Wu, C. (1997). Dynamic scheduling with intelligent agents: An application note. *Metra Application Note*, **105**.

Clark, A. (1998). *Being There: Putting Brain, Body, and World Together Again*. The MIT Press.

Claus, C. and Boutilier, C. (1998). The dynamics of reinforcement learning cooperative multiagent systems. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, pages 746–752.

Cole, B. J. (1991). Short-term activity cycles in ants: generation of periodicity by worker interaction. *American Naturalist*, **137**(2), 244–259.

Colorni, A., Dorigo, M., and Maniezzo, V. (1991). Distributed optimization by ant colonies. *Proceedings of the First European Conference on Artificial Life*, pages 134–142.

Comis, C. (2003). www.darwinbots.com.

Conrad, M. and Pattee, H. H. (1970). Evolution experiments with an artificial ecosystem. *Journal of Theoretical Biology*, **28**(3), 393–409.

Corne, K. P. D., Dorigo, M., Glover, F., Dasgupta, D., Moscato, P., and Poli, R., editors (1999). *New ideas in optimization*. McGraw-Hill Ltd., UK, Maidenhead, UK, England.

Costa, D. and Hertz, A. (1997). Ants can colour graphs. *Journal of the Operational Research Society*, **48**(3), 295–305.

Cotsaftis, M. (2006). From trajectory control to task control - emergence of self-organization in complex systems. *Understanding Complex Systems Series N° 10*.

Crites, E. H. and Barto, E. G. (1996). Improving elevator performance using reinforcement learning. *Proc. Conf. Neural Information Processing Systems 8, D. Touretzky, M. Mozer, and M. Hasselmo, Eds.*

Darr, T. P., Darr, T. P., Birmingham, W. P., and Birmingham, W. P. (1996). An attribute-space representation and algorithm for concurrent engineering. *AI EDAM*, **10**, 21–35.

Dash, R. K. (2005). Engineering distributed protocols for multi-agent interactions using game theory. page 1475.

Dash, R. K., Jennings, N. R., and Parkes, D. C. (2003). Computational-mechanism design: A call to arms. *IEEE Intelligent Systems*, **18**(6), 40–47.

De Jong, K. A. (1975). *An analysis of the behavior of a class of genetic adaptive systems*. Ph.D. thesis, University of Michigan, Ann Arbor.

Decker, K. and Lesser, V. (1995). Designing a family of coordination algorithms. *Proceedings of the First International Conference on Multi-Agent Systems*, pages 73–80.

Deloach, S. A., Wood, M. F., and Sparkman, C. H. (2001). Multiagent systems engineering. *International Journal of Software Engineering and Knowledge Engineering*, **11**(3), 231–258.

Dodgson, C. and Lewis, C. (1871). Alice through the looking glass. *The Annotated Alice*.

Dorigo, M., Maniezzo, V., and Colorni, A. (1996). Ant system: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, **26**(1), 29–41.

Durfee, E. and Montgomery, T. (1989). Mice: A flexible test bed for intelligent coordination experiments. *Proceedings of the 9th Workshop on Distributed AI*.

Eberhart, R. C. and Shi, Y. (2001). Particle swarm optimization: Developments, applications and resources. volume 1, pages 81–86.

Ellison, C. (1975). The utah tenex scheduler. *Proceedings of the IEEE*, **63**(6), 940–945.

Farinelli, A., Iocchi, L., and Nardi, D. (2004). Multirobot systems: A classification focused on coordination. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, **34**(5), 2015–2028.

Farmer, J. D. and Belin, A. (1992). Artificial life: The coming evolution. *Artificial Life II*, pages 815–840.

fei Yu, H. and wei Wang, D. (2006). Design and analysis of food-chain algorithm. volume 1, pages 453–456.

Feo, T. A. and Resende, M. G. C. (1995). Greedy randomized adaptive search procedures. *Journal of Global Optimization*, **6**(2), 109–133.

Ferber, J. (1999). *Multi-Agent Systems: An Introduction To Distributed Artificial Intelligence*. Addison-Wesley.

Ferguson, D., Yemini, Y., and Nikolaou, C. (1988). Microeconomic algorithms for load balancing in distributed computer systems. volume 8, pages 491–499.

Ferguson, D., Nikolaou, C., and Yemini, Y. (1989). An economy for flow control in computer networks. *INFOCOM '89. Proceedings of the Eighth Annual Joint Conference of the IEEE Computer and Communications Societies. Technology: Emerging or Converging, IEEE*, **1**, 110–118.

Ficici, S. G., Watson, R. A., and Pollack, J. B. (1999). Embodied evolution: A response to challenges in evolutionary robotics. *Proceedings of the Eighth European Workshop on Learning Robots*, pages 14–22.

Floreano, D., Nolfi, S., and Mondada, F. (1998). Competitive co-evolutionary robotics: From theory to practice. *Proceedings of the Fifth International Conference on Simulation of Adaptive Behavior (SAB'1998)*, pages 515–524.

Fogel, L. J., Owens, A. J., and Walsh, M. J. (1966). *Artificial Intelligence through Simulated Evolution*. John Wiley, New York, USA.

Friedman, G. J. (1959). Digital simulation of an evolutionary process. *General Systems Yearbook*, **4**, 171–184.

Fudenberg, D. and Levine, D. K. (1993). Steady state learning and nash equilibrium. *Econometrica*, **61**(3), 547–573.

Gao, J., Gen, M., and Sun, L. (2009). Modelling and scheduling preventative maintenance in semiconductor manufacturing industry with mas. *International Journal of Manufacturing Technology and Management*, **16**(1-2), 101–126.

Gardner, M. (1970). The fantastic combinations of john conway's new solitaire game 'life'. *Scientific American*, **223**(4), 120–123.

Gill, A. E. (1980). Some simple solutions for heat-induced tropical circulation. *Quarterly Journal, Royal Meteorological Society*, **106**(449), 447–462.

Glover, F. (1977). Heuristic for integer programming using surrogate constraints. *Decision sciences*, **8**, 156–166.

Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. *Computers and Operations Research*, **13**(5), 533–549.

Glover, F. (1994). Genetic algorithms and scatter search: unsuspected potentials. *Statistics and Computing*, **4**(2), 131–140.

Gmytrasiewicz, P. J. and Durfee, E. H. (2001). Rational communication in multi-agent environments. *Autonomous Agents and Multi-Agent Systems*, **4**(3), 233–272.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

Goldberg, D. E. and Smith, R. E. (1987). Nonstationary function optimization using genetic algorithms with dominance and diploidy. In *ICGA*, pages 59–68.

Goldman, V., Jeffrey, D., and Rosenschein, S. (1994). Emergent coordination through the use of cooperative state-changing rules. In *In Proceedings of the Twelfth National Conference on Artificial Intelligence*, pages 408–413.

Goldspink, C. (2002). Methodological implications of complex systems approaches to sociality: Simulation as a foundation for knowledge. *JASSS*, **5**(1).

Groβ, R. and Dorigo, M. (2008). Evolution of solitary and group transport behaviors for autonomous robots capable of self-assembling. *Adaptive Behavior - Animals, Animats, Software Agents, Robots, Adaptive Systems*, **16**(5), 285–305.

Guenther, O., Hogg, T., and Huberman, B. (1997). Market organizations for controlling smart matter. *Proc. Internat. Conference on Computer Simulation and Social Sciences*.

Guo, D.-W. and Kong, C.-L. (2004). A new artificial life algorithm to solve time-varying optimization problem. volume 4, pages 2146–2148.

Gutierrez-Maldonado, J., Alsina-Jurnet, I., Rangel-Gomez, M. V., Aguilar-Alonso, A., Jarne-Esparcia, A. J., Andres-Pueyo, A., and Talarn-Caparros, A. (2008). *Virtual intelligent agents to train abilities of diagnosis in psychology and psychiatry*, volume 142. Studies in Computational Intelligence.

Haken, H. (1986). *Fórmulas de Éxito en la Naturaleza*. Salvat.

Harty, K. and Cheriton, D. (1996). A market approach to operating system memory allocation. *Market-Based Control: A Paradigm for Distributed Resource Allocation*.

Harvey, I. (1997). Artificial evolution and real robots. *Artificial Life and Robotics*, **1**(1).

Hay, A. (1993). Equity and welfare in the geography of public transport provision. *Journal of Transport Geography*, **1**(2), 95–101.

He, Q. and Wang, L. (2007). An effective co-evolutionary particle swarm optimization for constrained engineering design problems. *Engineering Applications of Artificial Intelligence*, **20**(1), 89–99.

Hillis, W. D. (1990). Co-evolving parasites improve simulated evolution as an optimization procedure. *Physica D: Nonlinear Phenomena*, **42**(1-3), 228–234.

Holland, J. (1995). Hidden order: How adaptation builds complexity. *Reading, MA: Perseus*.

Holland, J. H. (1962). Outline for a logical theory of adaptive systems. *jacm*, **9**(3), 297–314.

Holland, J. H. (1975). *Adaptation in natural and artificial systems : an introductory analysis with applications to biology, control, and artificial intelligence*. University of Michigan Press, Ann Arbor.

Hu, J. and Wellman, M. P. (1996). Self-fulfilling bias in multiagent learning. *Proceedings 2nd International Conference on Multi-Agent Systems (ICMAS-96)*, pages 118–125.

Huberman, B. and Clearwater, S. H. (1995). A multi-agent system for controlling building environments. *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95)*, pages 171–176.

Husbands, P., Harvey, I., and Cliff, D. (1992). *Central issues in evolutionary robotics*.

Ihara, H. and Mori, K. (1984). Autonomous decentralized computer control systems. *Computer*, **17**(8), 57–66.

Ishiwaka, Y., Sato, T., and Kakazu, Y. (2003). An approach to the pursuit problem on a heterogeneous multiagent system using reinforcement learning. *Robotics and Autonomous Systems*, **43**(4), 245–256.

J., A. K. and Debreu, G. (1954). The existence of an equilibrium for a competitive economy. *Econometrica*, **22**, 265–90.

Jennings, N. (1995). The archon system and its applications. *Project Report, International Working Conference on Co-operating Knowledge Based Systems*.

Jennings, N. R., Mamdani, E. H., Laresgoiti, I., Perez, J., and Corera, J. (1992). Grate: A general framework for cooperative problem solving. *IEE-BCS Journal of Intelligent Systems Engineering*, **1**(2), 102–114.

Jennings, N. R., Sycara, K., and Wooldridge, M. (1998). A roadmap of agent research and development. *Autonomous Agents and Multi-agent Systems*, **1**, 275–306.

Journal, A. L. (Consultado en 2009). Mit press journals.

Julka, N., Karimi, I., and Srinivasan, R. (2002). Agent-based supply chain management-2: a refinery application. *Computers and Chemical Engineering*, **26**(11), 1771–1781.

Kennedy, J. and Eberhart, R. (1995). Particle swarm optimization. *Neural Networks, 1995. Proceedings., IEEE International Conference on*, **4**, 1942–1948 vol.4.

Kephart, J. O., Hanson, J. E., and Sairamesh, J. (1998). Price and niche wars in a free-market economy of software agents. *Artificial Life*, **4**(1).

Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, **220**, 671–680.

Kitano, H., Tambe, M., Stone, P., Veloso, M., Coradeschi, S., Osawa, E., Matsubara, H., Noda, I., and Asada, M. (1997). The robocup synthetic agent challenge 97. *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence*, pages 24–29.

Klein, J. (2002). Breve: A 3d simulation environment for the simulation of decentralized systems and artificial life. *Proceedings of Artificial Life VIII*.

Kolmogorov, A.Ñ. (1965). Three approaches to the quantitative definition of information. *Problems on information transmission*, **1**(1), 1–7.

Konidaris, G. D. and Hayes, G. M. (2005). An architecture for behavior-based reinforcement learning. *Adaptive Behavior*, **13**(1), 5–32.

Kosloff, D. and Kosloff, R. (1983). A fourier method solution for the time dependent schrödinger equation as a tool in molecular dynamics. *Journal of Computational Physics*, **52**(1), 35–53.

Koza, J. R. (1990). Genetic programming: A paradigm for genetically breeding populations of computer programs to solve problems. Technical Report STAN-CS-90-1314, Dept. of Computer Science, Stanford University.

Kraus, S. and Plotkin, T. (2000). Algorithms of distributed task allocation for cooperative agents. *Theoretical Computer Science*, **242**(1-2), 1–27.

Kraus, S., Sycara, K., and Evenchik, A. (1998). Reaching agreements through argumentation: A logical model and implementation. *Artificial Intelligence*, **104**(1-2), 1–69.

Kube, C. R. and Bonabeau, E. (2000). Cooperative transport by ants and robots. *Robotics and Autonomous Systems*, **30**(1), 85–101.

Kuhn, H. W. and Tucker, A. W. (1951). Nonlinear programming. *Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability, Ed.*

Kuo, J. Y. and Hsieh, F. (2008). Adaptive and cooperative learning for robocup agents. volume 6, pages 3125–3131.

Kuo, T. and Hwang, S.-Y. (1997). Using disruptive selection to maintain diversity in geneticalgorithms. *Applied Intelligence*, **7**(3), 257–267.

Kurose, J. and Simha, R. (1989). A microeconomic approach to optimal resource allocation in distributed computer systems. *IEEE Trans. Comput.*, **38**, 705–717.

Langle, T. and Worn, H. (2001). Human-robot cooperation using multi-agent-systems. *Journal of Intelligent and Robotic Systems: Theory and Applications*, **32**(2), 143–159.

Langton, C. G. (1997). *Artificial life : an overview*. MIT Press, Cambridge, Mass.

Laumanns, M., Rudolph, G., and Schwefel, H. P. (1998). A spatial predator-prey approach to multi-objective optimization: A preliminary study. *Parallel Problem Solving From Nature - PPSN V*, pages 241–249.

Lee, K. S. and Geem, Z. W. (2005). A new meta-heuristic algorithm for continuous engineering optimization: harmony search theory and practice. *Comput. Methods Appl. Mech. Engrg.*, **194**, 3902–3933.

Lesser, V. R. and Erman, L. D. (1980). Distributed interpretation: A model and experiment. *IEEE Transactions on Computers*, **C-29**(12), 1144–1163.

Luke, S., Cioffi-Revilla, C., Panait, L., Sullivan, K., and Balan, G. (2005). Mason: A multiagent simulation environment. *Simulation*, **81**(7), 517–527.

Luo, R. C. and Su, K. L. (2003). A multiagent multisensor based real-time sensory control system for intelligent security robot. volume 2, pages 2394–2399.

Luss, H. and Rosenwein, M. B. (1997). Operations research applications: Opportunities and accomplishments. *European Journal of Operational Research*, **97**(2), 220–244.

Maes, P. (1990). Designing autonomous agents. theory and practice from biology to engineering and back. *Robotics and Autonomous Systems*, **6**(1-2).

Magyari, E. and Keller, B. (2000). Exact solutions for self-similar boundary-layer flows induced by permeable stretching walls. *European Journal of Mechanics, B/Fluids*, **19**(1), 109–122.

Marín, J. and Solé, R. V. (1999). Macroevolutionary algorithms: A new optimization method on fitness landscapes. *IEEE Transactions on Evolutionary Computation*, **3**(4), 272–287.

Marocco, D. and Nolfi, S. (2007). Emergence of communication in embodied agents evolved for the ability to solve a collective navigation problem. *Connection Science*, **19**(1), 53–74.

Mataric, M. J. (1997). Reinforcement learning in the multi-robot domain. *Autonomous Robots*, **4**(1), 73–83.

McPartland, M., Nolfi, S., and Abbass, H. A. (2005). Emergence of communication in competitive multi-agent systems: A pareto multi-objective approach. pages 51–58.

Melián, B., Moreno Pérez, J. A., and Moreno Vega, J. M. (2003). Metaheurísticas: una visión global. *Inteligeencia Artificial, Revista Iberoamericana de Inteligencia Artificial*, **19**, 7–28.

Metropolis, N., Rosenbluth, A., Rosenbluth, M., Teller, A., and Teller, E. (1953). Equations of state calculations by fast computing machines. *Journal of Chemical Physics*, **21**, 1087–1092.

Michalewicz, Z. and Schoenauer, M. (1996). Evolutionary algorithms for constrained parameter optimization problems. *Evolutionary Computation*, **4**, 1–32.

Molina López, J. M., Herrero, J. G., Jiménez Rodríguez, F. J., and Casar Corredera, J. R. (2003). Cooperative management of a net of intelligent surveillance agent sensors. *International Journal of Intelligent Systems*, **18**(3), 279–307.

Mota, L. and Reis, L. P. (2008). *A common framework for co-operative robotics: An open, fault tolerant architecture for multi-league RoboCup teams*, volume 5325 LNAI.

Nash, J. F. (1950). Equilibrium points in n-person games. In *Proceedings of the National Academy of Sciences of the United States of America*.

Ng, K. P. and Wong, K. C. (1995). A new diploid scheme and dominance change mechanism for non-stationary function optimization. In *Proceedings of the 6th International Conference on Genetic Algorithms*, pages 159–166, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Nocedal, J. and Wright, S. J. (1999). *Numerical Optimization*. Springer.

Nolfi, S. (2005). Emergence of communication in embodied agents: Co-adapting communicative and non-communicative behaviours. *Connection Science*, **17**(3-4), 231–248.

Noriega, P. and Sierra, C. (1998). Agent mediated electronic commerce. *Proceedings of the 1st International Workshop on Agent Mediated Electronic Trading (AMET'98)*, page 1571.

O'Reilly, U. ., Ross, I., and Testa, P. (2000). Emergent design: Artificial life for architecture design. *Artificial Life VII*, pages 454–463.

Osman, I. H. and Kelly, J. P. (1996). *Meta-Heuristics: Theory and Applications*. Kluwer Academic Publishers, Boston.

O'Toole, D. V., Robinson, P. A., and Myerscough, M. R. (1999). Self-organized criticality in termite architecture: A role for crowding in ensuring ordered nest expansion. *Journal of theoretical biology*, **198**(3), 305–327.

Overgaard, L., Petersen, H. G., and Perram, J. W. (1994). Motion planning for an articulated robot: A multi-agent approach. *Distributed Software Agents and Applications: 6th European Workshop on Modelling Autonomous Agents in a Multi-Agent World*, pages 206–219.

Packard, N. H. (1989). Intrinsic adaptation in a simple model for evolution. *Artificial Life*, pages 141–155.

Pargellis, A.Ñ. and Yurke, B. (2000). Spontaneous generation and colony formation in the amoeba world. *Artificial Life VII Workshop Proceedings*, pages 71–73.

Parsons, S. and Wooldridge, M. (2002). Game theory and decision theory in multi-agent systems. *Autonomous Agents and Multi-Agent Systems*, **5**(3), 243–254.

Perez-Uribe, A., Floreano, D., and Keller, L. (2003). Effects of group composition and level of selection in the evolution of cooperation in artificial ants. volume 2801, pages 128–137.

Pincus, M. (1970). A monte carlo method for the approximate solution of certain types of constrained optimization problems. *Operations Research*, **18**, 1225–1228.

Potters, M., Cont, R., and Bouchaud, J. . (1998). Financial markets as adaptive systems. *Europhysics Letters*, **41**(3), 239–244.

Powell, D. and Skolnick, M. M. (1993). Using genetic algorithms in engineering design optimization with non-linear constraints. *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 424–431.

Praça, I., Ramos, C., Vale, Z., and Cordeiro, M. (2003). Mascem: A multiagent system that simulates competitive electricity markets. *IEEE Intelligent Systems*, **18**(6), 54–60.

Preist, C., Byde, A., and Bartolini, C. (2001). Economic dynamics of agents in multiple auctions. pages 545–551.

Prusinkiewicz, P. (2004). Modeling plant growth and development. *Current Opinion in Plant Biology*, **7**(1), 79–83.

Rao, S. S. (1996). *Engineering Optimization-Theory and Practice*. Wiley-IEEE, third edition.

Ray, T. S. (1991). An approach to the synthesis of life. *Artificial life II*, page 10.

Ray, T. S. (1992). Evolution, ecology and optimization of digital organisms. *Santa Fe Institute Working Paper*, **92**, 8–42.

Rechenberg, I. (1973). *Evolutionsstrategie: optimierung technischer systeme nach prinzipien der biologischen evolution*. Frommann-Holzboog.

Rekleitis, I., Lee-Shue, V., Peng, A., and Choset, H. (2004). Limited communication, multi-robot team based coverage. volume 2004, pages 3462–3468.

Reynolds, C. W. (1987). Flocks, herds, and schools: A distributed behavioral model. *Computer Graphics (ACM)*, **21**(4), 25–34.

Roumeliotis, S. I. and Bekey, G. A. (2002). Distributed multi-robot localization.

Ruiz-Mirazo, K., Pereto, J., and Moreno, A. (2004). A universal definition of life: Autonomy and open-ended evolution. *Origins of Life and Evolution of the Biosphere*, **34**(3), 323–346.

Russell, S. J. and Norvig, P. (2003). *Artificial Intelligence: A Modern Approach*. Pearson Education.

Sallez, Y., Berger, T., and Tahon, C. (2004). Simulating intelligent routing in flexible manufacturing systems using netlogo. volume 2, pages 1072–1077.

Samejima, K. and Omori, T. (1999). Adaptive internal state space construction method for reinforcement learning of a real-world agent. *Neural Networks*, **12**(7-8).

Sandholm, T. W. (1999). Distributed rational decision making. *Multiagent Systems: A Modern Introduction to Distributed Artificial Intelligence*, pages 201–258.

Sarma, J. and Jong, K. D. (1999). The behavior of spatially distributed evolutionary algorithms in non-stationary environments. In W. B. et al., editor, *GECCO*, volume 1, pages 572 –578. Morgan Kaufmann, San Francisco, California.

Satoh, T., Uchibori, A., and Tanaka, K. (1999). Artificial life system for optimization of nonconvex functions. volume 4, pages 2390–2393.

Schoonderwoerd, R., Bruten, J. L., Holland, O. E., and Rothkrantz, L. J. M. (1996). Ant-based load balancing in telecommunications networks. *Adaptive Behavior*, **5**(2), 169–207.

Schwabacher, M. and Gelsey, A. (1998). Multilevel simulation and numerical optimization of complex engineering designs. *Journal of Aircraft*, **35**(3), 387–397.

Schwefel, H. P. (1977). Numerische optimierung von computer-modellen mittels der evolutionsstrategie. *Basel: Birkhäuser*.

Semret, N. and Lazar, A. A. (1998). Design, analysis and simulation of the progressive second price auction for network bandwidth sharing. *CTR Technical Report*.

Sims, K. (1994). Evolving 3d morphology and behavior by competition. *Proceedings of Fourth Conference on Artificial Life*.

Smith, J. M. (1982). *Evolution and the Theory of Games*. Cambridge University Press.

Socha, K. and Kisiel-Dorohinicki, M. (2002). Agent-based evolutionary multiobjective optimisation. *Proc.of the 2002 Congress on Evolutionary Computation*, **1**, 109–114.

Solomonoff, R. J. (1960). A preliminary report on a general theory of inductive inference. In *Revision of Report V-131), Contract AF 49(639)-376, Report ZTB-138, Zator Co*.

Steels, L. (1998). The origins of ontologies and communication conventions in multi-agent systems. *Autonomous Agents and Multi-Agent Systems*, **1**(2), 169–194.

Stephanopoulos, G. (1989). Artificial intelligence in process engineering: current state and future trends. *Symp. on computer application in the chemical industry*, **116**, 3–25.

Stopford, M. (2002). Is the drive for ever bigger containerships irresistible? *Lloyds List Shipping Forecasting Conference*.

Storn, R. and Price, K. V. (1997). Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, **11**, 341 – 359.

Stützle, T. (1998). Applying iterated local search to the permutation flow shop problem. Technical report.

Stützle, T. and Hoos, H. H. (2000). Max-min ant system. *Future Generation Computer Systems*, **16**(8), 889–914.

Surry, P. D., Radcliffe, N. J., and Boyd, I. D. (1995). A multi-objective approach to constrained optimisation of gas supply networks: the comoga method. In *Selected Papers from AISB Workshop on Evolutionary Computing*, pages 166–180, London, UK. Springer-Verlag.

Symeonidis, A. L., Valtos, E., Seroglou, S., and Mitkas, P. A. (2005). Biotope: An integrated framework for simulating distributed multiagent computational systems. *IEEE Transactions on Systems, Man, and Cybernetics Part A:Systems and Humans.*, **35**(3), 420–432.

Taha, H. A. (2006). *Operations Research: An Introduction (8th Edition)*. Prentice-Hall, Inc.

Takeda, H., Hirata, Y., Wang, Z. D., and Kosuge, K. (2002). Collision avoidance algorithm for two tracked mobile robots transporting a single object in coordination based on function allocation concept. *Distrib Auton Robot Syst 5*, **5**, 155–164.

Tarapore, D., Floreano, D., and Keller, L. (2006). Influence of the level of polyandry and genetic architecture on division of labour. *The Tenth International Conference on the Simulation and Synthesis of Living Systems*.

Taylor, C. (1991). 'fleshing out' artificial life ii. *Artificial Life II*, pages 25–38.

Taylor, T. (1997). The cosmos artificial life system.

Taylor, T. (2001). Creativity in evolution: Individuals, interactions and environments. *Creative Evolutionary Systems. San Francisco: Morgan Kaufman*.

Terzopoulos, D. (1999). Artificial life for computer graphics. *Communications of the ACM*, **42**(8), 33–42.

Thalman, N. M. and Thalmann, D. (1994). *Artificial Life and Virtual Reality*. John Wiley & Sons, Inc., New York, NY, USA.

Theraulaz, G. and Bonabeau, E. (1993). A brief history of stigmergy. *Artificial Life*, **2**.

Theraulaz, G., Goss, S., Gervet, J., and Deneubourg, J. L. (1990). Swarm intelligence in wasps colonies: An example of task assignment in multiagents systems. *Proceedings of the 1990 IEEE International Symposium on Intelligent Control*, pages 135–143.

Thrun, S., Burgard, W., and Fox, D. (2000). Real-time algorithm for mobile robot mapping with applications to multi-robot and 3d mapping. volume 1, pages 321–328.

Tomlin, C., Member, S., Member, S., and Sastry, S. (1998). Conflict resolution for air traffic management: A study in multi-agent hybrid systems. *IEEE Transactions on Automatic Control*, **43**, 509–521.

Tu, X. and Terzopoulos, D. (1994). Artificial fishes: Physics, locomotion, perception, behavior. *Proceedings of SIGGRAPH*, pages 43–50.

Tucker, P. and Berman, F. (1996). On market mechanisms as a sofware techniques. *Technical Report CS96-513*.

Tuyls, K. and Nowé, A. (2005). Evolutionary game theory and multi-agent reinforcement learning. *Knowledge Engineering Review*, **20**(1), 63–90.

Urzelai, J. and Floreano, D. (2001). Evolution of adaptive synapses: robots with fast adaptive behavior in new environments. *Evolutionary computation*, **9**(4), 495–524.

Usui, Y. and Arita, T. (2003). Situated and embodied evolution in collective evolutionary robotics. *Proc.of the 8th International Symposium on Artificial Life and Robotics*, pages 212–215.

Van Valen, L. (1973). A new evolutionary law. *Evol. Theory*, **1**, 1–30.

Vlassis, N. (2003). A concise introduction to multiagent systems and distributed ai. *Fac. Sci.*

Waldspurger, C., Hogg, T., Huberman, B., Kephart, J., and Stornetta, W. (1992). Spawn: a distributed computational economy. *Software Engineering, IEEE Transactions on*, **18**(2), 103–117.

Wang, Y. . and Usher, J. M. (2005). Application of reinforcement learning for agent-based production scheduling. *Engineering Applications of Artificial Intelligence*, **18**(1), 73–82.

Wang, Y. and de Silva, C. W. (2008). A machine-learning approach to multi-robot coordination. *Engineering Applications of Artificial Intelligence*, **21**(3), 470–484.

Watson, R. A., Ficici, S. G., and Pollack, J. B. (1999). Embodied evolution: Embodying an evolutionary algorithm in a population of robots. *Proceedings of the Congress on Evolutionary Computation*, pages 335–342.

Watson, R. A., Ficici, S. G., and Pollack, J. B. (2002). Embodied evolution: Distributing an evolutionary algorithm in a population of robots. *Robotics and Autonomous Systems*, **39**(1), 1–18.

Weiss, G. (2000). *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*. MIT Press, Cambridge, MA, USA.

Wellman, M. (1994). A computational market model for distributed configuration design. *12th National Conference on Artificial Intelligence*.

Wellman, M. P. (1993). A market-oriented programming environment and its application to distributed multicommodity flow problems. *Journal of Artificial Intelligence Research*, **1**, 1–23.

Wellman, M. P. (1995). Computational market model for distributed configuration design. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing: AIEDAM*, **9**(2), 125–133.

Wischmann, S., Stamm, K., and Wörgötter, F. (2007). *Embodied evolution and learning: The neglected timing of maturation*, volume 4648 LNAI.

Wittig, T. (1992). *ARCHON: An Architecture for Multi-Agent Systems*. Ellis Horwood.

Wolfram, S. (1984). Cellular automata as models of complexity. *Nature*, **311**(5985), 419–424.

Wolpert, D. and Tumer, K. (2000). An illustration of the coin approach to design of multi-agent systems. In *Proceedings of the Agents 00 and ECML 00 Workshop on Learning in Agents*.

Wooldridge, M. (2002). *An Introduction to MultiAgent Systems*. Wiley.

Wooldridge, M. and Jennings, N. R. (1999). Cooperative problem-solving process. *Journal of Logic and Computation*, **9**(4), 563–592.

Yaeger, L. (1993). Computational genetics, physiology, metabolism, neural systems, learning, vision, and behavior or polyworld: Life in a new context. *Proc.of the 3rd Int.Conf on Artificial Life*.

Yamaguchi, T., Masubuchi, M., Tanaka, Y., and Yachida, M. (1996). Propagating learned behaviors from a virtual agent to a physical robot in reinforcement learning. pages 855–860.

Yang, B., Lee, Y., Choi, B., and Kim, H. (2001). Optimum design of short journal bearings by artificial life algorithm. *Tribology International*, **34**(7), 427–435.

Yang, B. S. and Lee, Y. H. (2000). Artificial life algorithm for function optimization. *Proceedings of 2000 ASME Design Engineering Technical Conferences and Computers and Information in Engineering Conference*.

Yang, B. S. and Lee, Y. H. (2004). Artificial life algorithm for function optimization. *2000 ASME Design Engineering Technical Conferences & Computer and Information in Engineering, 2000*.

Yanovski, V., Wagner, I. A., and Bruckstein, A. M. (2001). Vertex-ant-walk - a robust method for efficient exploration of faulty graphs. *Annals of Mathematics and Artificial Intelligence*, **31**(1-4), 99–112.

Zhao, B., Guo, C. X. C., and Cao, Y. J. (2005). A multiagent-based particle swarm optimization approach for optimal reactive power dispatch. *IEEE Transactions on Power Systems*, **20**(2), 1070–1078.

Zhong, W., Liu, J., Xue, M., and Jiao, L. (2004). A multiagent genetic algorithm for global numerical optimization. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, **34**(2), 1128–1141.

Zlot, R., Stentz, A. T., Dias, M. B., and Thayer, S. (2002). Multi-robot exploration controlled by a market economy. volume 3, pages 3016–3023.

Zundong, Z., Limin, J., and Yuanyuan, C. (2008). Research on elementary principals of complex system control.