UNIVERSIDADE DA CORUÑA

Escola Politécnica Superior

Trabajo Fin de Grado

CURSO 2019/20

APLICACIÓN DE ALGORITMOS DE OPTIMIZCIÓN EN MODELOS DE SIMULACIÓN

Grado en Ingeniería en Tecnologías Industriales

ALUMNA/O

Carlos Veiga González

TUTORAS/ES

Alejandro García del Valle

Diego Crespo Pereira

FECHA

DICIEMBRE 2019

1 TÍTULO Y RESUMEN

Aplicación de algoritmos de optimización en modelos de simulación

En el presente proyecto se desarrollará la optimización de una serie de modelos de simulación que representan diferentes situaciones en el ámbito industrial.

La simulación de dichos modelos se realizará con el software de simulación de eventos discretos FlexSim.

En cuanto a la optimización, será realizada una comparativa entre el optimizador de software cerrado utilizado por FlexSim: OptQuest y el software libre HeuristicLab, el cual dispone de un conjunto de algoritmos metaheurísticos.

El objetivo principal del presente proyecto consistirá en aplicar una serie de algoritmos presentes en las librerías de HeuristicLab y compararlos con los resultados obtenidos mediante OptQuest. Para alcanzar dicho objetivo será necesaria la comunicación entre ambos sotwares, lo cual será posible mediante el uso de sockets.

Aplicación de algoritmos de optimización en modelos de simulación

No presente proxecto desenvolverase a optimización dunha serie de modelos de simulación que representan diferentes situacións no ámbito industrial.

A simulación dos devanditos modelos realizarase co software de silumación de eventos discretos FlexSim.

En canto á optimización, será realizada unha comparativa entre o optimizador de software pechado utilizado por FlexSim: OptQuest e o software libre HeuristicLab, o cal dispón dun conxunto de algoritmos metaheurísticos.

O obxectivo principal do presente proxecto consistirá en aplicar unha serie de algoritmos presentes nas librerías de HeuristicLab e comparalos cos resultados obtidos mediante OptQuest. Para alcanzar o devandito obxectivo será necesaria a comunicación entre ambos os sotwares, o cal será posible mediante o uso de sockets.

Application of optimization algorithms in simulation models

In this project, the optimization of different simulation models that represent different situations in the industrial field will be developed.

The simulation of these models will be carried out with FlexSim, which is a discrete event simulation software.

As for the optimization, a comparison will be made between the closed software optimizer used by FlexSim: OptQuest and the free HeuristicLab software, which has a set of metaheuristic algorithms.

The main objective of this project will be to apply a series of algorithms present in HeuristicLab libraries and compare them with the results obtained through OptQuest. To achieve this goal, communication between both software will be necessary, which will be possible through the use of sockets.

UNIVERSIDADE DA CORUÑA

Escola Politécnica Superior

TRABAJO FIN DE GRADO

CURSO 2019/2020

APLICACIÓN DE ALGORITMOS DE OPTIMIZCIÓN EN MODELOS DE SIMULACIÓN

Grado en Ingeniería en Tecnologías Industriales

Documento I

MEMORIA

Índice de contenido

Índice de figuras

Índice de tablas

2 INTRODUCCIÓN

Realizar cambios en un proceso como, por ejemplo, una línea de producción puede ser muy costoso a la par que complejo. Por ende, es muy importante definir correctamente los cambios a realizar, haciendo un análisis exhaustivo de cada uno de los elementos que lo componen para evaluar si es conveniente llevarlo a cabo o no, y de ser el caso, hacerlo de la mejor forma posible.

Es en este contexto en el que cobra especial importancia la simulación de procesos, la cual es una herramienta muy útil para evaluar los cambios que se consideren oportunos en cada momento sin tener la necesidad de hacerlos en el sistema real. Esto permite probar numerosas configuraciones sin los elevados costes que supondría realizarlos por métodos tradicionales.

Dentro del ámbito de la simulación, en el presente proyecto, realizaremos una serie de optimizaciones haciendo uso de métodos metaheurísticos, centrándonos especialmente en los algoritmos evolutivos. Dichas optimizaciones se realizarán mediante evaluación externa, haciendo uso de sockets. Se comenzará por un caso básico para introducir el método, para posteriormente aplicarlo a un segundo modelo, de mayor complejidad. Por último, se aplicará el método a un caso real proporcionado por el GII, el cual simula el proceso de un taller de soldadura de nudos para celosías.

Una vez obtenidos los resultados de las optimizaciones, se realizará una comparativa entre la optimización mediante algoritmos metaheurísticos y la presente internamente en FlexSim: OptQuest.

3 OBJETIVOS

El presente Trabajo de Fin de Grado tiene como objetivo principal ***aplicar algoritmos de optimización metaheurísticos mediante comunicación externa para una serie de procesos y compararlo con las diferentes alternativas existentes***. Para la implementación de dichos algoritmos debemos cumplir una serie de subobjetivos consistentes en:

- Desarrollar el modelo de simulación de cada caso estudiado mediante el software de simulación 3D de eventos discretos FlexSim.
- Efectuar la optimización de la función objetivo para cada caso en un software especializado, en nuestro caso HeuristicLab. Dicho software nos permite escoger algoritmos metaheurísticos para posteriormente personalizar sus parámetros y adaptarlos a nuestros casos concretos.
- Comunicar ambos programas para que se transfieran información mediante sockets.
- Aplicar resultados al modelo 3D.
- Recoger los resultados obtenidos a través de informes de simulación.

Una vez realizados todos los subobjetivos expuestos anteriormente, se procederá al análisis de dichos resultados y la comparación con los obtenidos por otros métodos, sacando las conclusiones oportunas sobre qué método es mejor aplicar en cada caso.

4 ANTECEDENTES

4.1 Industria 4.0 y Simulación

A lo largo de la historia hemos vivido varias etapas de cambio tecnológico. Desde la Primera Revolución Industrial, marcada por la invención de la máquina de vapor, pasando por la producción en serie de la mano de Henry Ford y la automatización de los procesos industriales a finales del siglo XX. Pero es en la actualidad cuando estamos presentes en una nueva etapa de la industria, debida a los numerosos avances existentes en las nuevas tecnologías, denominada ***Cuarta Revolución Industrial*** o ***Industria 4.0.***

Figura 1: Pilares de la Industria 4.0. Fuente: poderindustrial.com

Este cambio se basa en ayudarse de las nuevas tecnologías para la automatización y optimización del proceso productivo, y para ello se genera un entorno virtual que analiza el proceso real y valora cuáles son las opciones que valorar para mejorarlo.

Figura 2: Trabajadores empleando modelos de simulación. Fuente: izaro.com

Dentro de esta etapa de cambio, la simulación es uno de sus pilares fundamentales, ya que es la encargada de valorar los resultados de un supuesto cambio en el proceso sin la necesidad de ponerlo en práctica previamente en el Sistema Real. La digitalización de este Sistema Real da lugar al Sistema Virtual, y es comúnmente denominado Digital Twin (en español, Gemelo Digital).

Es sencillo deducir que mediante el Sistema Virtual pueden ahorrarse costes de manera notable, ya que desaparece la necesidad de parar el proceso productivo para realizar los cambios en el mismo y también se vuelven nulos los posibles gastos asociados a un supuesto cambio en el Sistema Real. Una vez valoradas todas las opciones posibles para mejorar el sistema, se escoge la mejor y se implementa en el sistema real con todos los parámetros influyentes claramente definidos en el Gemelo Digital.

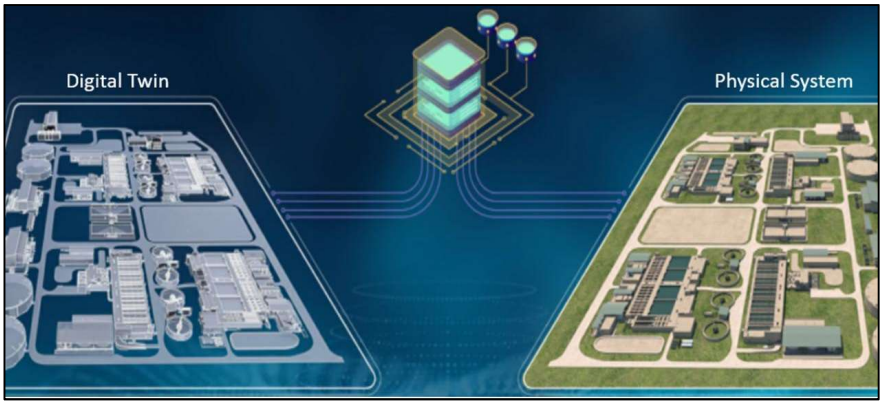

Figura 3: Ejemplo de sistema digital y sistema real. Fuente: jacobs.com

Dicho de otra forma, tal y como dicta la *guía práctica para la simulación de procesos industriales*, los modelos de sistemas que pretendemos simular pueden dividirse en dos grandes grupos: modelos físicos y modelos matemáticos

Los modelos físicos son una recreación del sistema real en tamaño y complejidad reducidas. En este grupo podrían incluirse maquetas, túneles de viento, etc. Este modelo tiene limitaciones evidentes, ya que, aunque tiene cierta flexibilidad con respecto al sistema real, sigue siendo mucho más complicado realizar cambios.

Por otro lado, los modelos matemáticos son los encargados de predecir de una forma muy precisa el comportamiento de un determinado sistema. Para ello hacen uso de distintas herramientas, tales como la programación lineal, la física o la geometría. Cabe destacar además que este tipo de modelos engloba otros dos: los modelos analíticos y los modelos de simulación.

Los modelos analíticos son los que llegan a sus resultados a partir de la resolución de una fórmula conocida o un conjunto de ellas. En muchas ocasiones (especialmente en este tipo de problemas) estos métodos son demasiado complejos, por lo que resolver un problema de optimización suele ser muy laborioso y requiere de la sincronización exhaustiva de personas de varios departamentos (físicos, informáticos, ingenieros) y durante períodos elevados de tiempo, por lo que no es la forma más aconsejable de resolver un problema de optimización. Además, puede ser muy complicado encontrar las ecuaciones que mejor representen el comportamiento del sistema real. Todo ello se resume en que la dificultad para resolver un problema por este método puede ser tan elevada que no tenga sentido su implementación.

Es por ello por lo que aparecen los modelos de simulación, los cuales proporcionarán una solución correcta a la optimización además de permitir ver de forma estimada

cómo se comportan todos los elementos que lo componen. Por lo tanto, en numerosas ocasiones, la simulación es el único método capaz de resolver problemas de optimización, tales como producción, distribuciones, logística, etc. Comparándolo directamente con los métodos analíticos, en este caso es mucho menor la necesidad y complejidad de coordinación entre distintos departamentos, ya que es una resolución que por lo general se hace de forma mucho más sencilla, además de abaratar costes. Pese a todo, los modelos analíticos no son inútiles, ya que sirven de base para desarrollar posteriormente los algoritmos de los modelos de simulación de forma eficiente.

Antes de profundizar más en el ámbito de la simulación, es imprescindible establecer la definición de **sistema**, el cual puede definirse como *un conjunto de objetos y entidades que interactúan entre sí para alcanzar un determinado objetivo*. Dichos elementos del sistema podrían pertenecer a su vez a un sistema de mayores dimensiones, por lo que se convertirían en un subsistema que tiene un fin de servir al sistema al que pertenecen para alcanzar su objetivo.

El **estado de un sistema** define y caracteriza todas las variables mínimas necesarias para que sea capaz de realizar el objetivo que se le ha establecido en un instante determinado del tiempo. Dichas variables son denominadas **variables de estado.**

Ahora bien, considerando exclusivamente los sistemas que evolucionan a lo largo del tiempo, éstos pueden clasificarse en: continuos, discretos, orientados a eventos discretos y combinados.

- **Sistemas continuos:** Las variables de estado evolucionan de forma continua con el tiempo. Un ejemplo de este tipo de variable es el control de la temperatura de un fluido con el tiempo.
- **Sistemas discretos:** Las características de las variables de estado permanecen constantes hasta que ocurre un cambio o secuencia de cambios en el sistema. Estos cambios son definidos como **eventos**, y en este tipo de sistemas obedecen normalmente a un patrón periódico.
- **Sistemas orientados a eventos discretos:** Al igual que los sistemas discretos, las características de las variables de estado permanecen constantes hasta la aparición de un evento, pero en este caso dichos eventos suelen obedecer a patrones aleatorios.
- **Sistemas combinados:** Como bien indica su nombre, es una combinación de subsistemas continuos y discretos. Como ejemplo lo que ocurre en un proceso de producción de botellas de refresco: en principio el fluido es tratado mediante sistemas continuos y pasa a ser discontinuo en cuanto es embotellado.

Debido a que los modelos de estudio de este proyecto presentan características puramente discretas, nos centraremos en este tipo de sistemas.

4.2 Modelos de simulación de eventos discretos

Los modelos de simulación de eventos discretos presentan características dinámicas, discretas y estocásticas en los que las variables de estado cambian debido a la aparición de eventos en el sistema.

Los elementos de interés dentro de este tipo de modelos son:

- **Eventos:** Acción instantánea que tiene la capacidad de realizar cambios en las variables de estado.
- **Actividades:** Son las acciones o tareas que tienen lugar en el sistema. Normalmente necesitan del uso de recursos y tienen una determinada duración, que puede ser constante o aleatoria.
- **Entidades:** Conjunto de elementos que componen el sistema (normalmente permanentes) o fluyen por el mismo (normalmente temporales). Podrían incluirse tanto las máquinas de un proceso como las piezas que son procesadas.
- **Entidades temporales:** Son los elementos que se procesan en el sistema, ya sean las piezas que se han puesto como ejemplo en las **Entidades**, clientes, etc. Las entidades pueden tener diferentes características entre ellas, las cuales serán denominadas **atributos.** Estos atributos pueden ser, prioridad dentro del proceso, precio, color, tamaño, etc.
- **Recursos o entidades permanentes:** Medios gracias a los cuales las actividades pueden desarrollar correctamente su función y realizar cambios en las entidades temporales. Ejemplos de ello son los operarios, maquinas, etc.

Para realizar la correcta modelización del sistema, es necesario el uso de un software que permita establecer todos y cada uno de estos elementos. Para el presente proyecto se ha decidido hacer uso del software de simulación de eventos discretos FlexSim, el cual pasaremos a detallar a continuación.

4.2.1 El software FlexSim y OptQuest

FlexSim es un software para la simulación de eventos discretos, que permite modelar, analizar, visualizar y optimizar cualquier proceso industrial, desde procesos de manufactura hasta cadenas de suministro.

Figura 4: Logo de FlexSim. Fuente: flexsim.com

Además, FlexSim permite trabajar en un entorno tridimensional desde el comienzo del desarrollo del proyecto, lo cual hace que la interfaz sea más amigable tanto para el equipo técnico que lo está desarrollando como para el cliente al que va dedicado, ya que de un simple vistazo puede hacerse una idea de lo que ocurre en el modelo. Debido a estos motivos, entre otros, cada vez son más las empresas que implementan este software en sus procesos, por lo que la comunidad de desarrolladores de FlexSim aumenta y permite mejorar las competencias del software. Dichas mejoras se ven reflejadas en la página web, a través de la cual se generan interacciones con usuarios mediante foros y en la cual se pueden descargar librerías, formas 3D, entre otras herramientas adicionales para el software.

En cuanto a la interfaz de usuario, en este software es claramente más intuitiva que la mayoría de su competencia, reduciendo la curva de aprendizaje drásticamente. Para iniciar un nuevo modelo de simulación, basta con iniciar el programa y seleccionar "Create New Model". Esto nos llevará a la ventana principal, la cual consta de partes bien diferenciadas:

- **La ventana de modelización 3D:** Donde puedes editar todos los elementos que conforman tu sistema y realizar modificaciones sobre ellos.
- **Ventana de Librerías:** a la izquierda de la pantalla, desde la cual podrás seleccionar los distintos elementos que conformen tu sistema y arrastrarlos hacia la ventana de modelización para introducirlos en el modelo. Existen numerosos elementos para escoger, desde procesadores, fuentes, sumideros, cintas de transporte, etc. Además, podremos introducir formas propias para dichos elementos, de forma que será posible realizar el sistema virtual de una forma mucho más fiel al real.
- **La barra de herramientas:** como en cualquier otro programa, será aquí desde donde podremos guardar el archivo, iniciar un nuevo proyecto o abrir otro existente, además de las características específicas de este software, entre las cuales se encontrará la posibilidad de elaborar scripts, los cuales se utilizarán para la implementación de algoritmos externos, tal y como se explicará en posteriores apartados con todo detalle.

El motivo por el que se ha seleccionado este entorno de simulación, además de lo mencionado anteriormente, es que cuenta con un sistema integrado de optimización basado en **OptQuest**, el cual nos permitirá realizar comparaciones con los resultados obtenidos mediante la aplicación de algoritmos de optimización realizada de forma externa y valorar su viabilidad.

Figura 5: Logos de OptTek y OptQuest, respectivamente. Fuente: opttek.com

Tal y como dice OptTek en su página web (www.opttek.com) OptQuest es un módulo de optimización diseñado para facilitar su integración en aplicaciones que requieren de la optimización de sistemas de alta complejidad, ya sea que estos utilicen simulación o no.

Este software emplea algoritmos metaheurísticos, optimización matemática y redes neurales para la búsqueda de las mejores soluciones a cualquier tipo de problema relacionado con la toma de decisiones y planificación.

Los ejemplos de usos que nos indica el fabricante son:

- Diseño de procesos de producción y servicio
- Planificación de la cadena de suministro
- Planificación de la producción
- Diseño de rutas de vehículos
- Optimización de la fuerza laboral
- Diseño de sistemas de distribución
- Diseño de sistemas de energía y medioambiente

Jorge Antonio Faundes Berkhoff (alumno de la Escuela de Ingeniería de la Pontificia Universidad Católica de Chile) en su tesis "*Análisis del rendimiento del algoritmo QptQuest para optimización en simulación*" de 2014 ha realizado un estudio detallado sobre el algoritmo comercial OptQuest aplicado a redes de optimización en simulación, con lo que ha encontrado las fortalezas y debilidades del mismo. Esto nos es de gran ayuda como punto de partida para conocer su comportamiento y poder así compararlo con el método que propondremos posteriormente, que será el de integrar los distintos algoritmos disponibles en las librerías del software HeuristicLab.

En dicha tesis, se llegan a una serie de conclusiones que expondré a continuación separadas según distintos puntos de vista: **velocidad de convergencia, diversidad y calidad de las soluciones** y **estructura del problema:**

- **Velocidad de convergencia:** En todos los casos que ha propuesto el algoritmo ha alcanzado la convergencia *de forma agresiva y muy rápida*, llegando a valores de mejora de un 94% en 10.000 iteraciones en uno de ellos. Cabe destacar también que dicha velocidad desciende con el tiempo, ya que en las últimas 1800 iteraciones de ese mismo algoritmo tan solo mejora un 14%. Además, el algoritmo converge más rápido si las soluciones iniciales son de buena calidad según el valor de la función objetivo.
- **Diversidad y calidad de las soluciones:** Es este apartado se detecta que OptQuest tiene *dificultades para ocupar la diversidad* de forma eficiente para encontrar buenas soluciones que sean diversas. Esto se traduce en que el algoritmo podría estancarse en una buena solución local, omitiendo una más lejana que podría ser de mayor calidad.
- **Estructura del problema:** Para problemas que no son altamente sensibles a las variaciones de las variables de decisión, los efectos de modificar dichas variables solo afectan a una pequeña parte del horizonte de simulación. En el caso de que afecten de forma notable, dicho horizonte es afectado por completo, por lo que es un factor a tener en cuenta de cara a la resolución del problema. Por último, se comprueba que OptQuest no trata de forma adecuada el problema cuando aparecen restricciones aleatorias.

Según este análisis, las recomendaciones finales que se tienen son las de resolver inicialmente una simulación de bajas iteraciones y obtener así una población de soluciones de una mayor calidad. Esta población será la inicial del algoritmo que nos proporcionará la solución definitiva, ya que así realizaremos la optimización de una forma más eficiente.

4.3 Algoritmos metaheurísticos

Un algoritmo se puede definir como una secuencia de instrucciones que representan un modelo de solución para determinado tipo de problemas. O bien como un conjunto de instrucciones que realizadas en un orden concreto conducen a obtener la solución de un problema.

Es importante dedicar tiempo previo al desarrollo del programa para realizar lo mejor posible el algoritmo, ya que como dice Luis Joyanes (2008), programador experto y autor de numerosos libros de lógica y programación "en la ciencia de la computación y en la programación, los algoritmos son más importantes que los lenguajes de programación o las computadoras. Un lenguaje de programación es sólo un medio para expresar un algoritmo y una computadora es sólo un procesador para ejecutarlo".

Los algoritmos deben tener unas características determinadas, las cuales son:

- Debe ser **preciso**, sin dar lugar a ambigüedades.
- Tiene que estar bien **definido**, es decir, si lo ejecutamos dos veces se obtendrá un resultado idéntico.
- Ha de ser **finito**.
- Obligatoriamente debe producir un **resultado**.

Llegamos a la conclusión de que un algoritmo es suficiente para alcanzar la solución de un problema. En caso de que haya varios algoritmos que lleguen a la misma solución, se escogerá el que lleve menos tiempo. De eso trata la optimización, tema central del presente proyecto.

Para la resolución de un problema, es conveniente seguir una serie de pasos. Estos son:

- **Definición del problema y análisis**. Consiste en tener claros los datos proporcionados por el problema (datos de entrada) y entender bien que es necesario obtener como resultado (datos de salida).
- **Diseño del algoritmo.** Éste se realiza normalmente mediante pseudocódigos o diagramas de flujo. Éstos tienen la estructura mostrada en la Tabla 1.

Tabla 1: Elementos de un diagrama de flujo. Fuente: intef.es

símbolo	Función	Símbolo	Función
Terminal	Indicar el inicio y fin del diagrama	Teclado	Introducir datos manualmente por el teclado
Entrada/salida	Entrada o salida simple de información	Decisión	Indica operaciones lógicas o de comparación y tienen dos salidas dependiendo del resultado.
Proceso	Realizar cualquier operación o calculo con la información		
Salida a Impresora	Salida de informacion a la impresora	Conectores	Une dos partes del diagrama a la misma o diferente página
Salida a Pantalla	Mostrar información de salida a la pantalla	Flechas de Flujo	Indica la direccion del flujo de la información

- **Codificación.** A partir del diagrama de flujo creado en el paso anterior se crea un código en un determinado lenguaje de programación que represente cada uno de los pasos establecidos en el diagrama.
- **Compilación.** En este paso, el software seleccionado por el usuario convierte el código generado anteriormente en instrucciones que son reconocibles para el ordenador.
- **Ejecución.** Se ejecuta el programa para observar los resultados obtenidos
- **Depuración.** Según los resultados de la ejecución, se corrigen los errores y se optimiza el código para llegar a la solución deseada sin ningún tipo de error, cumpliendo en todo momento los requisitos de los algoritmos mencionados anteriormente.
- **Evaluación de resultados y documentación.** Una vez completados todos los pasos anteriores, se realiza un estudio sobre los resultados obtenidos en las diferentes ejecuciones del programa, ya que existe la posibilidad de que no veamos ningún error de compilación, pero sí existan errores en el programa que hagan que no lleguemos a la solución deseada.

Es importante documentar todo lo realizado en cualquier momento del proceso de elaboración del programa, ya sean los errores cometidos o los éxitos, para así tener unas referencias claras tanto para partes siguientes del proyecto como para proyectos futuros.

Si bien un algoritmo siempre tiene como objetivo alcanzar la solución óptima a un problema (o varios) determinado, hay ocasiones en las que dichos problemas son tan complejos que no es posible llegar a dicha solución. Es en el caso de, por ejemplo, ecuaciones no lineales, en el que el resultado que damos por bueno es el resultado que más se aproxime a la solución entre todas las iteraciones que hayamos realizado durante el experimento. En ocasiones sería posible alcanzar dicho valor óptimo, pero el tiempo de cálculo sería tan descabellado que no merecería la pena el tiempo perdido. Un método heurístico tiene como objetivo encontrar una solución que contenga el equilibrio entre fiabilidad de dicha solución y tiempo que tarda en alcanzarla.

Estos métodos están basados en ideas determinadas para cada caso al que se aplican, por lo que normalmente son métodos del topo ad hoc. Es decir, cada método heurístico es diseñado específicamente para un problema en concreto y cada vez que comenzábamos un nuevo problema debíamos comenzar de cero un método heurístico personalizado para el mismo.

Pero esto ha cambiado en los últimos años, ya que hoy en día existen métodos metaheurísticos, los cuales pueden ser aplicados a una mayor variedad de problemas. Es un método general el cual da una serie de pautas y características para poder desarrollar un método heurístico. Estos métodos son los más utilizados actualmente en el mundo de la ingeniería y se utilizarán en el presente proyecto mediante el software HeuristicLab.

4.3.1 HeuristicLab

El software empleado para realizar la evaluación externa mediante el uso de algoritmos metaheurísticos será HeuisticLab, que es un marco para algoritmos heurísticos y evolutivos desarrollado por miembros del *Heuristic and Evolutionary Algorithms Laboratory* (HEAL) desde 2002.

Figura 6: Logo de HeuristicLab. Fuente: heuristiclab.com

El equipo de desarrolladores utiliza esta página para coordinar esfuerzos para mejorar y extender HeuristicLab. Es un software libre que dispone de librerías de algoritmos para implementar en problemas de optimización. Su descarga está disponible desde su sitio web, en el cual también se puede encontrar otra información de interés, como publicaciones realizadas por los miembros de HEAL, basadas en el uso de metaheurísticos para distintos ámbitos, tales como finanzas, ingeniería o problemas biomédicos.

En este proyecto los algoritmos utilizados serán los evolutivos, más concretamente el algoritmo genético y la estrategia evolutiva basada en la covarianza matricial (CMA).

4.3.2 Algoritmos genéticos

Los algoritmos genéticos son un tipo de metaheurísticos que se diferencian notablemente de los demás. Pertenecen al grupo de los algoritmos evolutivos y tienden a ser particularmente efectivos explorando diversas partes de la región factible y evolucionan gradualmente hacia las mejores soluciones. Es por ello por lo que serán uno de los utilizados en este proyecto.

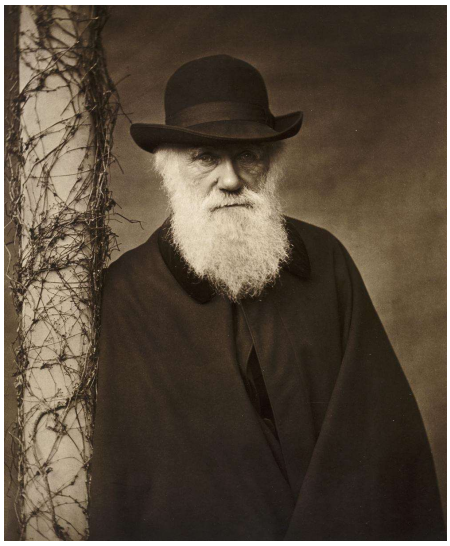

Figura 7: Charles Darwin. Fuente: National Geographic.

Es fácilmente apreciable la influencia que presenta este tipo de algoritmos con la Teoría de la evolución formulada por Charles Darwin a mediados del siglo XIX. El científico británico observó como cada especie, ya sea planta o animal, tiene una gran

variación individual. Son los que transmiten una ventaja de supervivencia a través de una adaptación mejorada al medio en el que se desenvuelven los que tienen mayor probabilidad de sobrevivir a la generación siguiente. Este fenómeno, acuñado por el mismo Darwin, es conocido como supervivencia por adaptación.

Hoy en día, el campo de la genética ha avanzado a pasos agigantados y podemos ayudarnos de ese avance para saber más en detalle de este fenómeno. En toda especie que tenga un sistema de reproducción sexual, se transmite un determinado número de cromosomas a la nueva generación de cada uno de los padres. Son éstos los que determinan las características del hijo, por lo que si el hijo hereda los mejores genes de generaciones pasadas tiene mayor probabilidad de desenvolverse correctamente hasta que de paso a la siguiente generación y así hacer lo mismo con sus descendientes, proporcionándole de nuevo unos buenos genes que ayuden a prosperar de una forma todavía mejor a la siguiente generación. Por lo tanto, podría decirse que, con el paso de las generaciones, la población tiende a mejorar mediante la repetición sucesiva de este proceso y de forma lenta.

Además, cabe destacar la existencia de un segundo factor que puede acelerar o decelerar el proceso: las mutaciones aleatorias de bajo nivel en el DNA (en español ADN, Ácido Desoxirribonucleico). Estas mutaciones son las que, en ocasiones, generen mejoras en el cromosoma que un hijo hereda de su padre. Pero esto ocurre de manera ocasional, ya que la mayoría de las mutaciones no tienen efecto sobre el desenvolvimiento de los hijos.

Los conceptos básicos de esta teoría se pueden aplicar a problemas de optimización. El equivalente a las diferentes especies son las soluciones a un determinado problema y la adaptación de cada miembro se mide por el valor de una función que estableceremos como función objetivo.

Una de las claves de este tipo de algoritmos y lo que lo diferencian de otros también conocidos (tales como el recocido simulado o la búsqueda tabú) es que, en lugar de procesar una solución por cada iteración, se realizan pruebas con una población de soluciones prueba. Para cada iteración del algoritmo genético existe una población, que contiene las soluciones prueba actuales. Dichas soluciones son la equivalencia a los miembros vivos de la especie: la generación actual. Algunos de esos miembros son seleccionados como padres (dos soluciones de la población actual), los cuales tendrán hijos (nuevas soluciones de prueba) con algunos genes (características) idénticos a sus antecesores. Como los miembros de la población con mejores resultados tienen mayor probabilidad de ser padres que los demás, con la sucesión de las iteraciones el algoritmo va generando poblaciones que otorgan mejores resultados de la función objetivo.

Las mutaciones ocurren en un bajo porcentaje, y aplicado a la optimización se traduce en que las nuevas soluciones prueba tienen características diferentes a la de sus padres. Esto permite explorar una parte nueva de la región factible, lo cual puede llevar a resultados mejores.

Por lo tanto, al final de un número de iteraciones (tan elevado como complejo sea el proceso) o tras un intervalo de tiempo especificado, el algoritmo concluirá con una población de soluciones prueba que será próxima a la óptima.

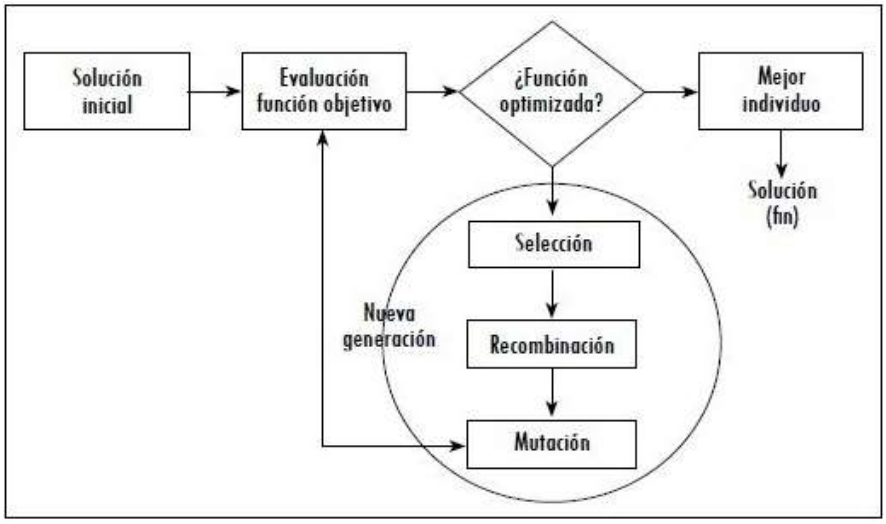

Figura 8: Diagrama de flujo para algoritmo evolutivo. Fuente: researchgate.net

Es de destacar que el proceso tiene claras equivalencias con la teoría de la evolución, pero no es necesario ajustarse a la analogía con todo detalle. De esta filosofía nacen algunos tipos de algoritmos genéticos, los cuales permiten que una solución prueba sea el padre repetidamente en numerosas iteraciones de este. Concluimos con que la equivalencia entre la teoría de la relatividad y este tipo de algoritmos es más un punto de partida que una metodología rígida a la que seguir sin variaciones, y adaptarla a cada problema para conseguir los mejores resultados.

4.3.3 Estrategias evolutivas

Al igual que los ya mencionados algoritmos genéticos, las estrategias evolutivas se basan en la teoría de la evolución establecida por Darwin para la optimización de una función objetivo. Sin embargo, la recombinación de población que da lugar a siguientes generaciones (mutación) no se hace de forma aleatoria, sino que es un proceso determinístico o estocástico (según la estrategia evolutiva que se use), realizado a través de una distribución normal multivariante (o distribución gaussiana multivariante). Esta distribución es una generalización de la distribución normal unidimensional a dimensiones superiores.

Un pseudocódigo general para este tipo de algoritmos es el siguiente:

```
0 dado ρ, μ, λ ∈ N+
1 inicializar P = {(xk; f(xk)) | 1 ≤ k ≤ μ}
2 Q = {}
3 para k ∈ {1, ... , λ}
4 seleccionados = seleccionar_compañeros(ρ, P)
5 xk = recombinar(seleccionados)
6 xk  = mutar(xk)
7 Q = Q + (xk; f(xk))
8 P = P U Q
9 P = selección_por_edad(P)
10 P = seleccionar_mejor(μ, P)
```

Donde:

- **μ:** Tamaño de la población
- **ρ:** Número de padres seleccionados para recombinarse
- **λ:** Número de individuos en la descendencia

Existen multitud de estrategias evolutivas, y cada una tiene asociado un método mediante el cual realizan dicho proceso de mutación. El más comúnmente utilizado en la práctica es el denominado CMA-ES (Covariance Matrix Adaptative, en español Matriz de Covarianza Adaptativa). Como su propio nombre indica, en este tipo de estrategias evolutivas se analiza la covarianza entre dos muestras de valores consecutivas para así determinar el camino a seguir en su evolución hacia siguientes generaciones.

Todo ello se traduce en que los resultados obtenidos mediante estrategia evolutiva tienen una velocidad de convergencia mayor que la de los algoritmos genéticos, además de tener menor tendencia a estancarse en máximos locales de la función objetivo dentro del horizonte de simulación.

5 CASOS DE SIMULACIÓN

En este apartado se procederá con la realización de varios casos sobre los que aplicar la optimización mediante el uso de algoritmos metaheurísticos. Los casos irán en orden creciente de dificultad, siendo el primero uno básico el que se realizarán pruebas con los algoritmos de HeuristicLab para poder depurar y validar el método. A continuación, se realizará sobre un caso similar al primero, pero con un requerimiento computacional mucho mayor, lo cual dificultará la tarea de optimización. Por último, se abordará el caso real proporcionado por la UMI, el cual es el modelo de un taller de nudos para la fabricación de celosías.

5.1 Caso base: tres procesadores en serie

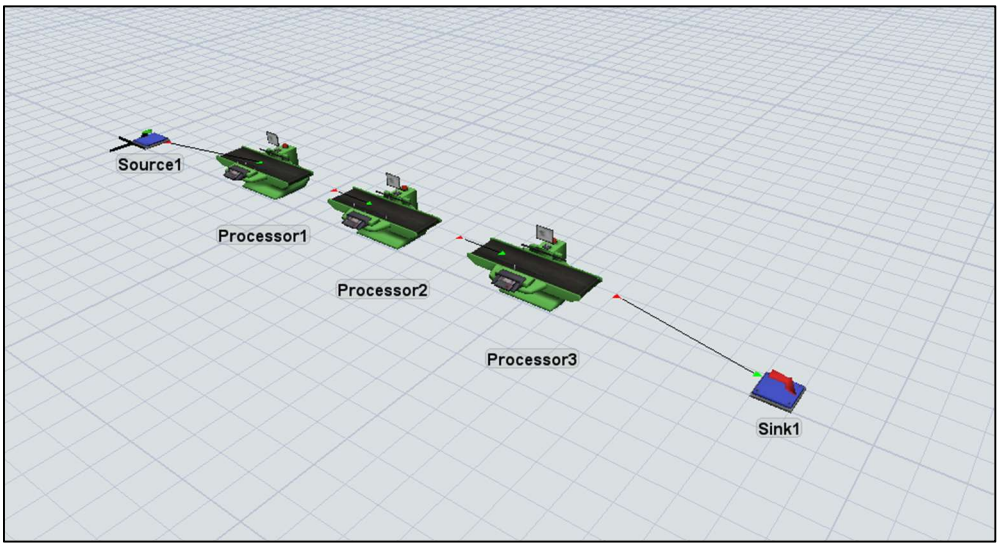

Figura 9: Imagen 3D del modelo base

Como primera toma de contacto con el proyecto, se ha propuesto un modelo muy sencillo para realizar la optimización mediante la aplicación de algoritmos. Este modelo, denominado *Modelo Base*, consiste en una serie de procesadores (Processors en FlexSim), una fuente (Source) y un sumidero (Sink). De esta forma al ejecutar el modelo comenzarán a salir cajas de la fuente, pasarán por los tres procesadores en serie y se destinarán al sumidero. Las llegadas están configuradas mediante una distribución exponencial tal y como muestra la Figura 10:

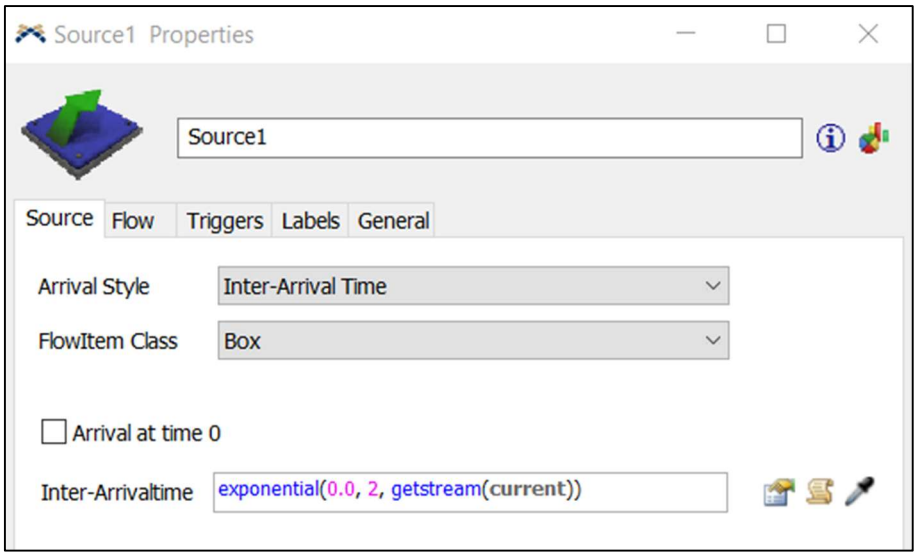

Figura 10: Configuración de la fuente para el Caso Base

En cuanto a los procesadores, el tiempo de procesado ha sido configurado mediante una distribución normal logarítmica, la cual está caracterizada por media y desviación típica. Ninguno de los procesadores requiere de tiempo adicional antes de comenzar el proceso, es decir, el tiempo que transcurren las cajas en cada uno de ellos es el definido en la pestaña Process Time. En la Tabla 2 puede observarse como los procesadores 1 y 3 se comportan de manera idéntica (con una media de 10 y desviación típica de 1), mientras que el 2 la media también es la misma pero la desviación típica tiene un valor de 2.

Tabla 2: Configuración de los procesadores para el Caso Base

Ítem	Media	Desviación
Processor 1	10	1
Processor 2	10	2
Processor 3	10	1

5.1.1 Optimización con OptQuest

A continuación, se procederá a realizar la optimización mediante el uso del optimizador interno de FlexSim, el cual funciona mediante la tecnología OptQuest. Ésta se realiza desde el menú de **Experimenter**, situado en la barra de herramientas (menú Statistics).

Al entrar en el experimentador, entre otras herramientas, se encuentra el optimizador. Para poder elaborar una optimización, es necesario establecer la función que queremos optimizar (el equivalente a la calidad en los algoritmos evolutivos) y todas las variables de las que depende (las capacidades de los procesadores).

Las variables se crean dentro del apartado **Scenarios**, y se establecen las capacidades máximas seleccionándolas con la herramienta clonar desde el árbol de propiedades de cada elemento.

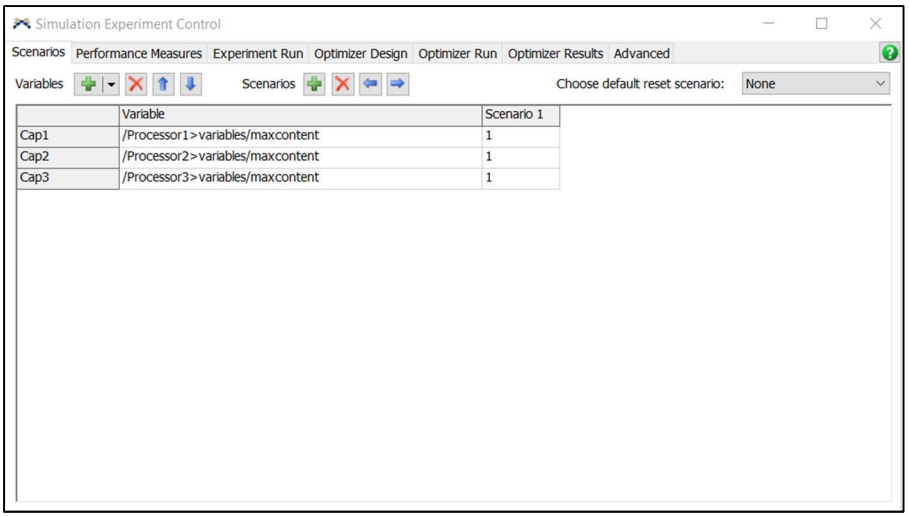

Figura 11: Configuración de las variables de decisión para el Caso Base.

Una vez establecidas las variables, definimos la función objetivo beneficio, la cual depende de todas ellas, además del valor input dentro del sumidero al final de cada simulación. Este valor representa el número total de ítems que han sido procesados en el modelo al trascurrir el tiempo de simulación.

$$Beneficio = 1 \cdot Productos - 5 \cdot Capacidades$$

Donde:

- **Productos:** Número de ítems que han sido enviados al sumidero al finalizar la simulación
- **Capacidades:** Suma de las capacidades de los tres procesadores.

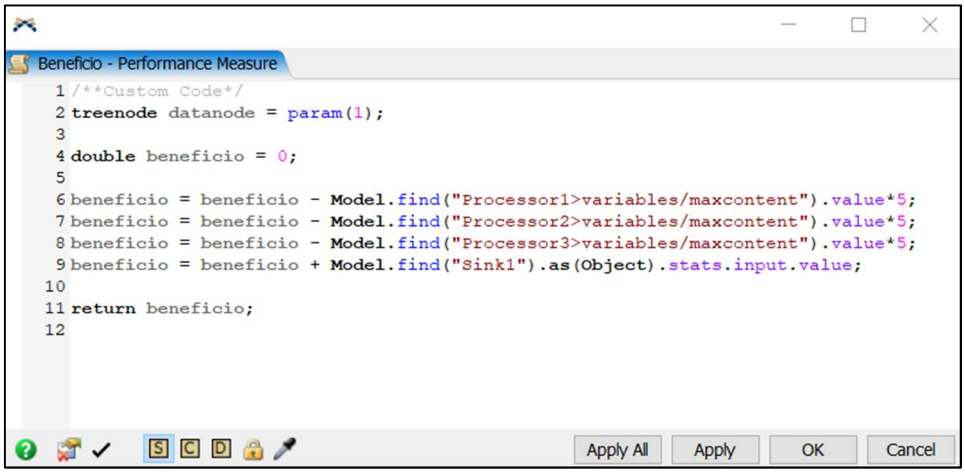

Figura 12: Configuración de la función objetivo para el Caso Base.

Una vez listos los parámetros, procedemos a efectuar la optimización de la función objetivo. Esto se realiza estableciendo los limites superiores e inferiores de las variables, así como el tipo de variable de que se trata y el step entre distintas iteraciones.

En nuestro caso, todas ellas son de tipo entero, con límite inferior y superior de 1 y 10 respectivamente, y con un step igual a la unidad.

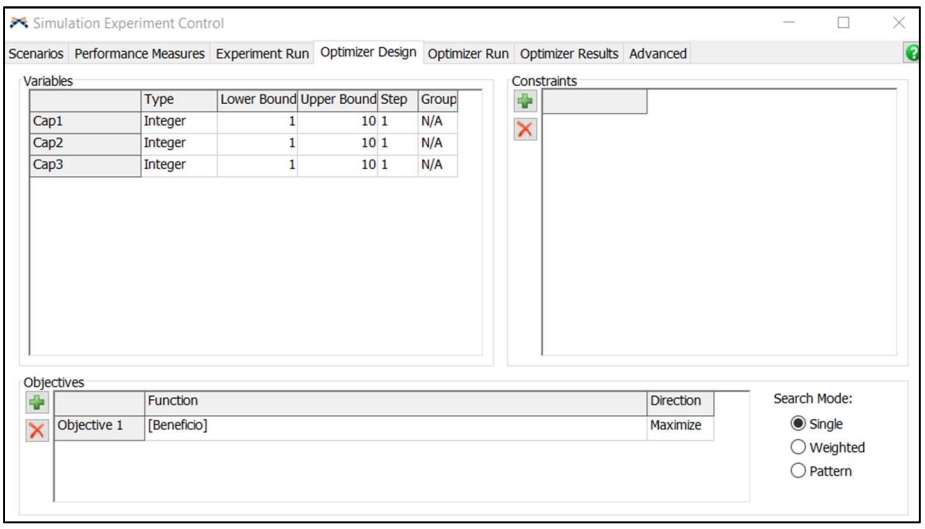

Figura 13: Diseño de optimización en OptQuest Para el Caso Base

Como puede apreciarse en la parte derecha de la Figura 13, también existe la posibilidad de establecer restricciones que debe cumplir la optimización, función que no será utilizada en el presente proyecto.

A continuación, pasamos a configurar la pestaña Optimizer Run. En ella se podrán establecer diversos parámetros, tales como el número de soluciones que queremos que se nos proporcione o el número máximo y mínimo de réplicas por escenario. En lugar de realizar un determinado número de réplicas, FlexSim nos permite realizar cuantas replicas sean necesarias hasta alcanzar un intervalo de confianza que seleccionemos.

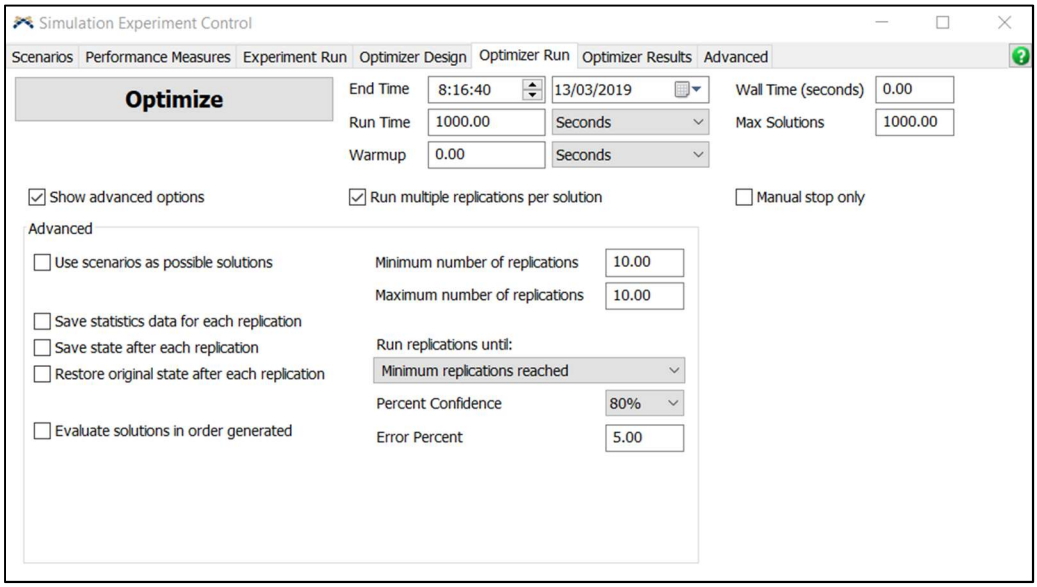

Figura 14: Configuración de Optimizer Run para el Caso Base

Para el caso que estamos tratando en particular, seleccionaremos 1000 soluciones de 10 replica cada una, utilizando un tipo de búsqueda singular. Los resultados obtenidos son los mostrados en la imagen:

Tabla 3: Mejor configuración del Caso Base obtenida con OptQuest

Solution ID	Processor1	Processor2	Processor3	Beneficio
129	10	7	6	364,1

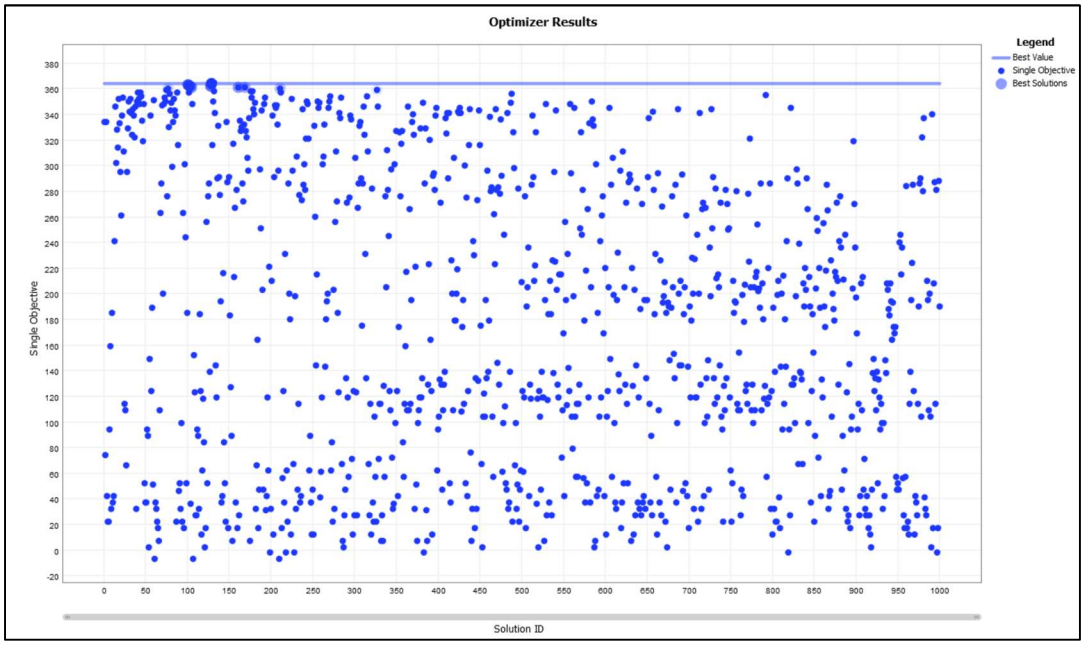

Figura 15: Resultado de optimización del Caso Base con OptQuest

Como se puede apreciar en la Figura 15, debido a la escasa complejidad del caso, OptQuest apenas tarda una centena de iteraciones en aproximarse considerablemente a la configuración que proporciona el valor óptimo del beneficio. Dicho valor se alcanza en la iteración 129 y corresponde a una configuración de capacidades máximas de 10 para el primer procesador, 7 para el segundo y 6 para el tercero. De esta manera, se ha obtenido un beneficio de 364,1€. Este valor el resultante de la media de beneficio en las 10 iteraciones realizadas en ese escenario, por lo tanto, puede ser un valor decimal.

5.1.2 Optimización con HeuristicLab

El primer paso para realizar la optimización mediante un software externo es el de conectar los dos programas (FlexSim y HeuristicLab). Esto se realizará mediante conexión de sockets, por lo que deberemos elaborar un script en FlexSim que genere dicha conexión en un puerto disponible y acceder desde el optimizador a ese socket para transferir información entre ambos programas. De esta forma será posible realizar la optimización del modelo de simulación.

Es en el script de FlexSim donde establecemos parámetros como el tiempo de simulación, capacidades de los procesadores, etc. Estos datos son enviados al

optimizador y éste devuelve los resultados que ha generado, que serán los más favorables según una función objetivo que se haya establecido. Dicha función objetivo, en este caso, es el beneficio industrial. Al ser un ejemplo básico hemos decidido que esa función dependerá exclusivamente de la tasa de producción del proceso y de las capacidades de los procesadores.

A continuación, es necesario conectar el optimizador externo a FlexSim. Para ello, ejecutamos el software HeuristicLab y seleccionamos el apartado de Optimizer. Una vez ejecutado, nos aparecerá una interfaz en la que, en la parte superior izquierda, podremos crear un nuevo ítem, lo que nos permite seleccionar el algoritmo que queramos usar para nuestro problema de optimización.

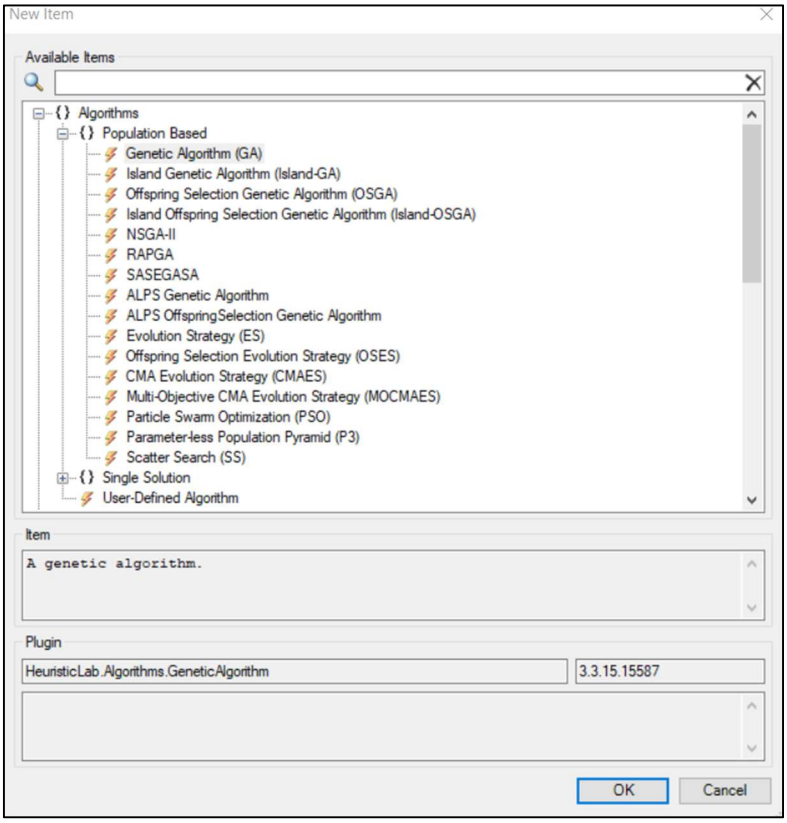

Figura 16: Interfaz del optimizador de HeuristicLab

En este proyecto, se utilizarán algoritmos basados en una población, por lo que se seleccionará ese desplegable y, a continuación, se escoge uno de los disponibles. Como ejemplo, se hará uso del algoritmo genético (Genetic Algorithm (GA)).

Una vez seleccionado el algoritmo, lo primero que necesitamos establecer es el problema que deseamos optimizar. En el caso del presente proyecto, es necesario elaborar un script que permita la conexión del optimizador con FlexSim mediante el socket abierto previamente. Por lo tanto, el tipo de problema seleccionado será el de problema programable (Programmable Problem (single-objective)).

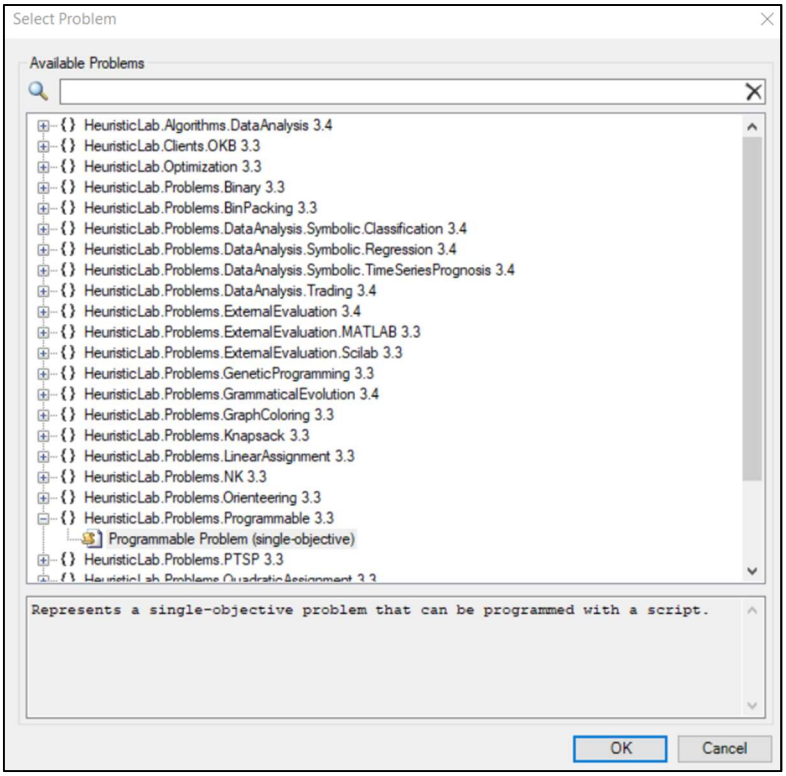

Figura 17: Interfaz de selección de problema en HeuristicLab

En él se introducirá el código que será el encargado de que amos software se transfieran la información necesaria para realizar la optimización correctamente.

Por lo tanto, en orden cronológico, ocurrirá lo siguiente:

- En primer lugar, desde el script de FlexSim, se abrirá un puerto para permitir la conexión entre ambos programas.
- A continuación, HeuristicLab se conectará a éste para realizar la comunicación. Se ejecutarán una serie de simulaciones (establecidas según convenga) y de cada una de ellas se obtendrán unos resultados diferentes de la función objetivo (Beneficio). Así, al tratarse de un algoritmo evolutivo, cuantas más simulaciones hagamos, el optimizador más se aproximará a la mejor solución, que será la que obtenga un mayor beneficio.
- Al acabar el número de simulaciones total, seleccionará la simulación que ha obtenido mayor beneficio y devolverá a un vector las capacidades de los procesadores que dieron lugar a ese valor máximo, y lo sustituirá en el modelo de simulación de FlexSim.

5.1.2.1 Implementación de algoritmo genético

Una vez seleccionado el algoritmo y definido el problema que se quiere optimizar, es necesario determinar qué parámetros son los óptimos para el caso de simulación concreto. En el caso del algoritmo genético, podemos seleccionar la cantidad de generaciones, así como la población de cada una de ellas y la tasa de mutación.

Para hacer una comparativa equitativa, se ha decidido establecer un número máximo de iteraciones de 1.000 para todos los métodos empleados. Dicho número de iteraciones, en el caso del algoritmo genético, viene dado por el producto entre el número de generaciones y la población. Por lo tanto, se debe cumplir que:

$$nIteraciones = Generaciones \times Población = 1.000$$

En base a esta restricción, serán probadas las configuraciones mostradas en la Tabla 4.

Tabla 4: Optimizaciones realizadas con algoritmo genético para el Caso Base

ID Optimización	Generaciones	Población	% Mutación
1	100	10	10
2	100	10	5,0
3	100	10	2,5
4	50	20	10
5	50	20	5,0
6	50	20	2,5
7	20	50	10
8	20	50	5,0
9	20	50	2,5
10	10	100	10
11	10	100	5,0
12	10	100	2,5

Una vez realizadas todas las simulaciones, se obtienen los resultados mostrados en la Tabla 5.

Tabla 5: Resultados obtenidos con algoritmo genético para el Caso Base

ID Optimización	Processor1	Processor2	Processor3	Quality
1	8	7	7	372
2	8	7	6	370
3	7	7	6	363
4	9	7	7	379
5	8	6	6	364
6	7	7	6	355
7	9	7	7	368
8	8	7	7	367
9	7	6	6	359
10	9	8	6	364
11	9	7	6	368
12	9	7	7	370

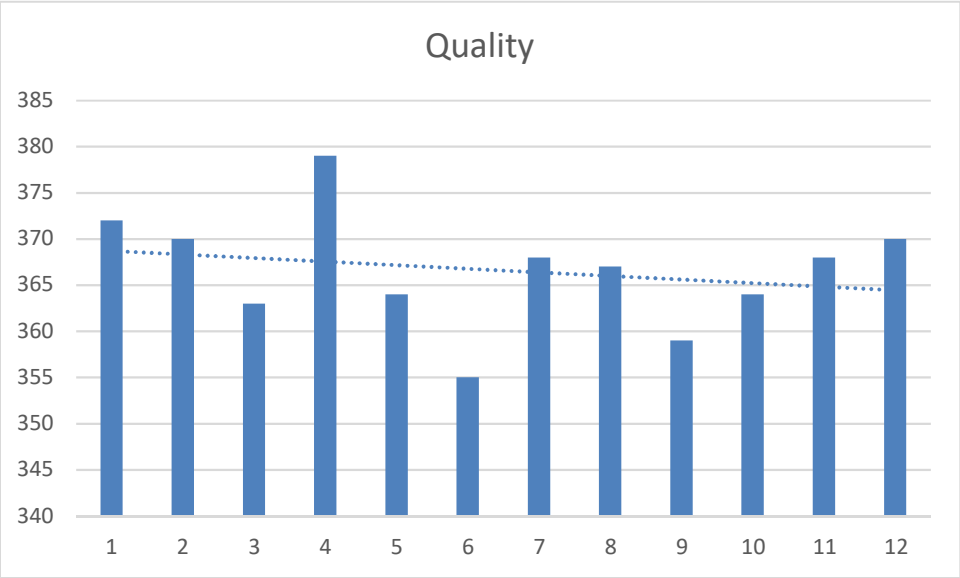

Figura 18: Resultados de optimización con algoritmo genético en el Caso Base

En la Figura 18 puede apreciarse que, aunque existe una cierta aleatoriedad entre las soluciones obtenidas, el beneficio tiene una considerable tendencia a ser mayor cuando mayor sea la relación Generaciones/Población. Además, puede apreciarse como el factor de mutación parece afectar positivamente a la mejoría del beneficio, a excepción del caso de 10 generaciones de 100 soluciones de población.

Pasando a analizar ahora cada simulación por separado, puede observarse de manera más clara la influencia de la relación generaciones/población y la tasa de mutación. Por ejemplo, en las simulaciones de 100 generaciones de 10 soluciones de población, el algoritmo genético ha convergido de la forma siguiente:

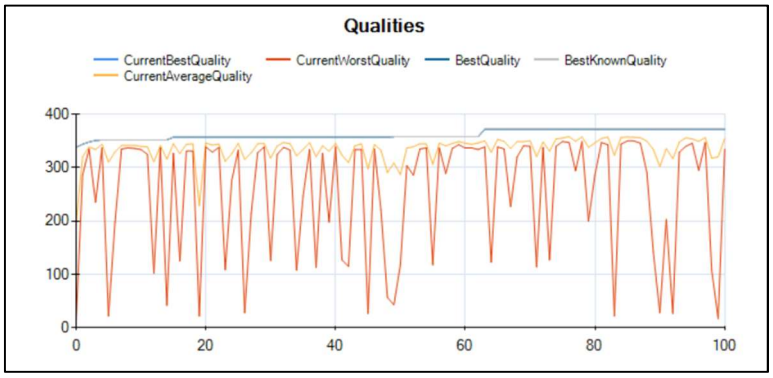

Figura 19: Solución 1 con algoritmo genético para el Caso Base

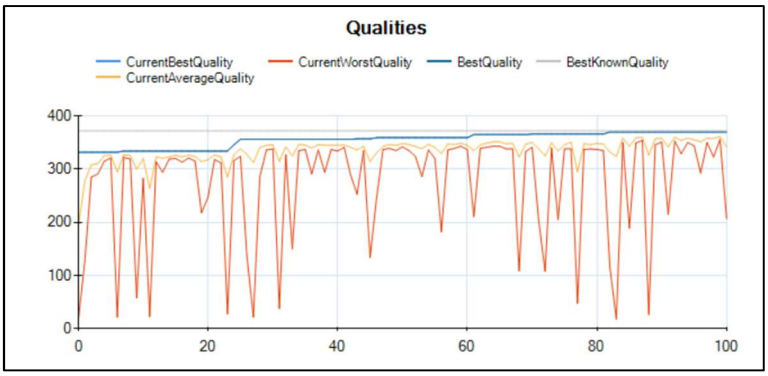

Figura 20: Solución 2 con algoritmo genético para el Caso Base

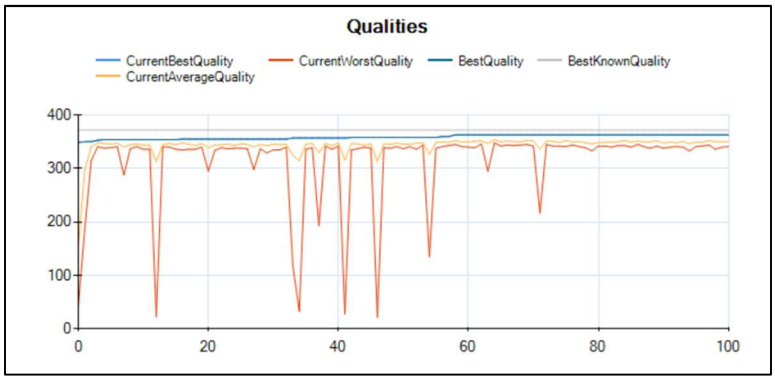

Figura 21: Solución 3 con algoritmo genético para el Caso Base

En los gráficos de las Figuras 19, 20 y 21, en el eje de ordenadas se indica el valor de la función objetivo y el eje de abscisas corresponde con el número de generaciones transcurrido. En ellas se aprecia de manera clara que, a medida que disminuye la tasa de mutación, la evolución de las generaciones hacia la mejor solución se realiza de manera mucho más homogénea. Este hecho tiene como inconveniente la mayor facilidad para alcanzar convergencias prematuras en máximos locales del horizonte de resultados, ya que la probabilidad de que una parte de la población explore otras soluciones se reduce. Por otro lado, si son comparadas tres soluciones con igual tasa de mutación, se obtienen las gráficas de las Figuras 22, 23 y 24.

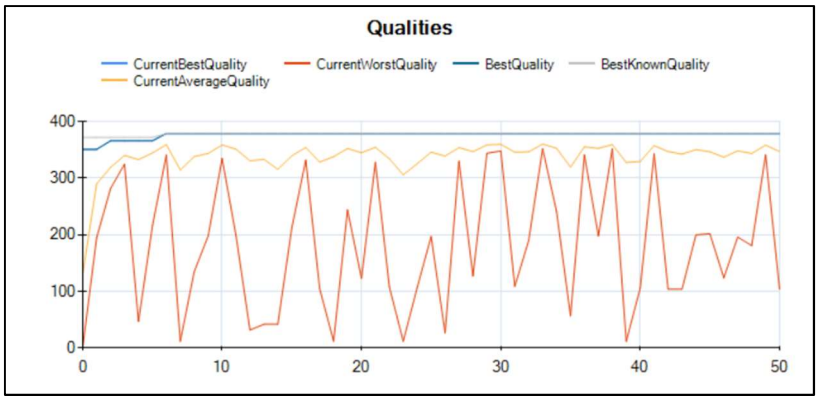

Figura 22: Solución 4 con algoritmo genético para el Caso Base

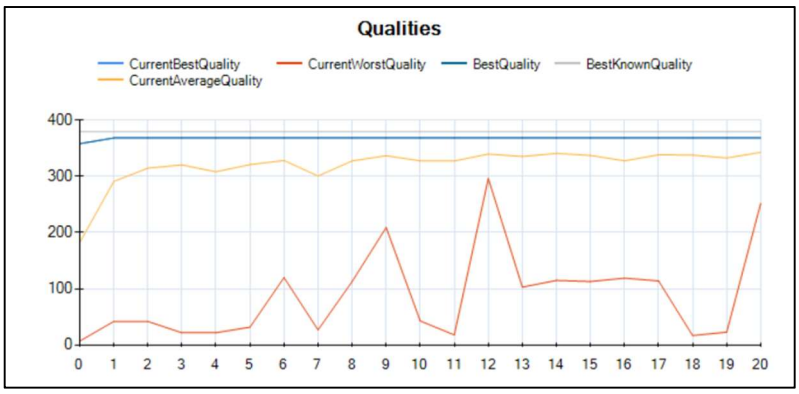

Figura 23: Solución 7 con algoritmo genético para el Caso Base

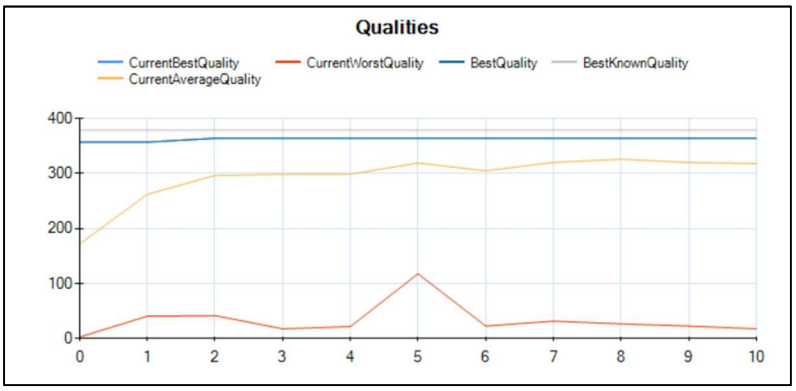

Figura 24: Solución 10 con algoritmo genético para el Caso Base

En los gráficos puede apreciarse como las soluciones 4, 7 y 10 tienen convergencias dispares. A medida que baja el número de generaciones existe más distanciamiento entre las líneas de mejor y peor calidad obtenida, lo que se traduce en que la población de cada generación abarca un rango mucho mayor. Además, al haber menor número de generaciones, la evolución de la población que las compone es escasa, por lo que la velocidad de convergencia se ve afectada negativamente.

5.1.2.2 Implementación de estrategia evolutiva CMA

Se procederá a continuación a la implementación de la estrategia evolutiva a la optimización del modelo. El procedimiento es similar al anterior, pero en este caso será necesario comunicar ambos softwares mediante un vector de números reales, ya que es el tipo de conjunto con el que trabaja el CMA. Como las capacidades de los procesadores son valores enteros, una vez optimizada cada solución, es necesario truncar los valores que envía HeuristicLab previamente al cálculo de la función objetivo.

En este caso se realizarán 4 simulaciones, variando exclusivamente el número de generaciones y su población, tal y como se muestra en la Tabla 6.

Tabla 6: Optimizaciones realizadas con CMA para el Caso Base

ID Simulación	Generaciones	Población
1	100	10
2	50	20
3	20	50
4	10	100

Una vez realizadas las 4 simulaciones, se obtienen los resultados mostrados en la Tabla 7.

Tabla 7: Valores de variables de decisión obtenidos con CMA en el Caso Base

ID Simulación	Processor1	Processor2	Processor3	Quality
1	9	7	7	376
2	9	7	6	376
3	9	7	6	374
4	9	7	7	379

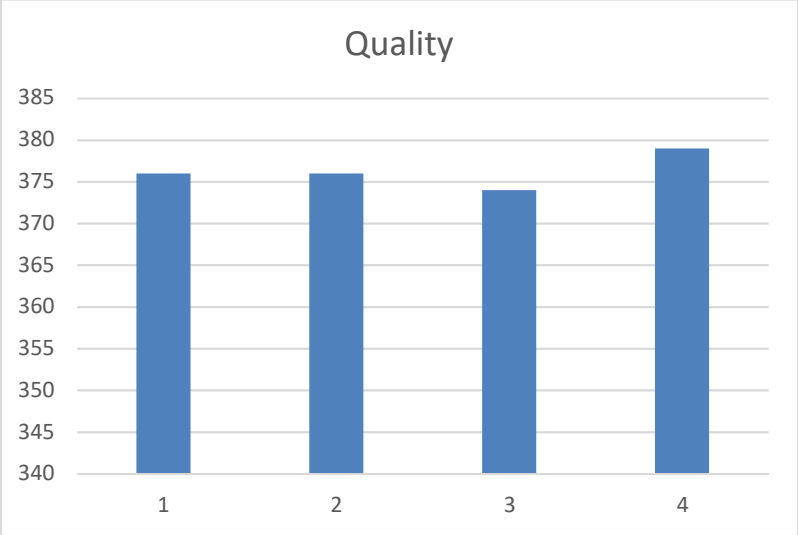

Figura 25: Resultados de optimización con CMA en el Caso Base

Como se puede apreciar en la Figura 25, los cuatro resultados tienen unos valores elevados de la calidad, ya que en todos ellos superan los 370€. Además, existe muy poca variación entre ellos, debido a la escasa dificultad del problema de optimización. Esas diferencias son debidas muy probablemente a la desviación que tienen los valores del tiempo de procesamiento de los procesadores.

De nuevo, analizando la convergencia de cada una de las simulaciones realizadas podemos sacar conclusiones sobre la dependencia que tiene esta de los parámetros que se han variado:

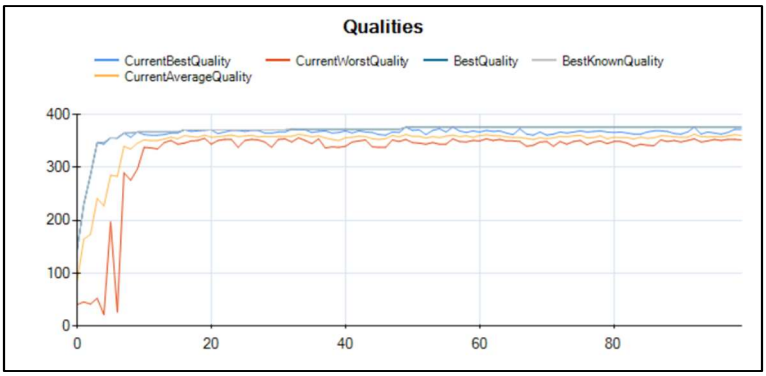

Figura 26: Solución 1 con CMA para el Caso Baase

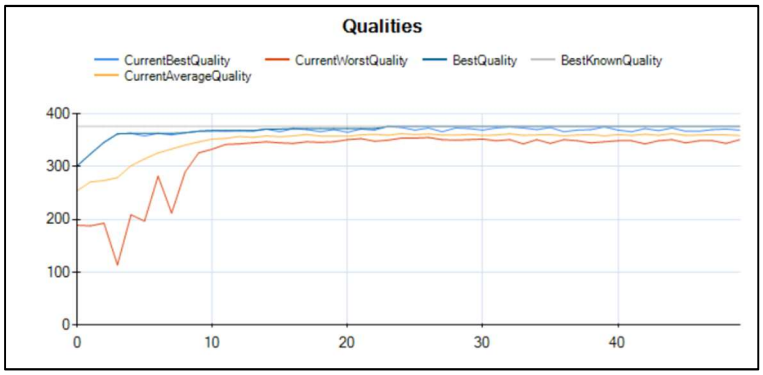

Figura 27: Solución 2 con CMA para el Caso Base

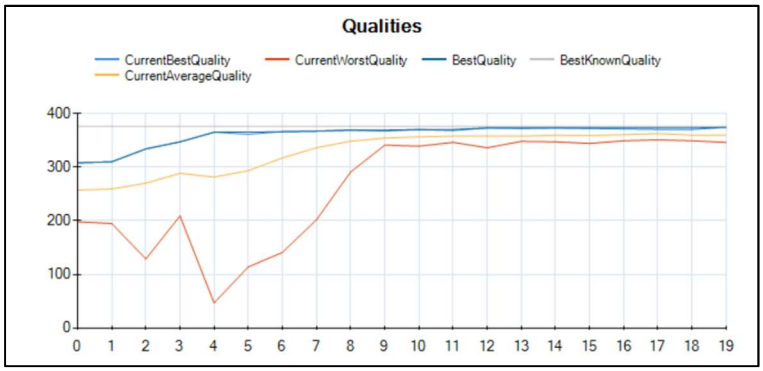

Figura 28: Solución 3 con CMA para el Caso Base

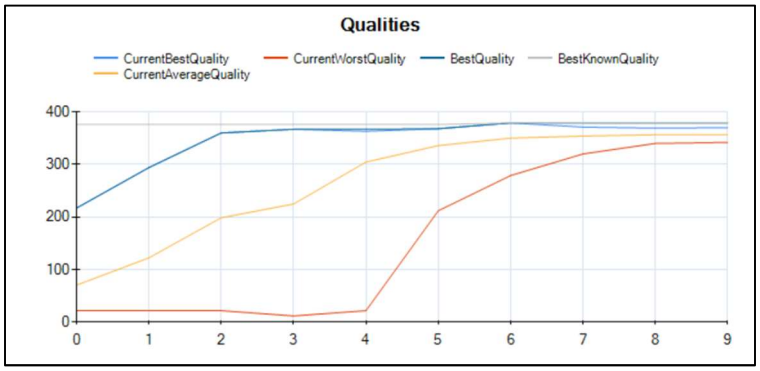

Figura 29: Solución 4 con CMA para el Caso Base

Aunque en todas las simulaciones se alcanza la convergencia de manera ágil, es notable la influencia de la relación generaciones/población. Igual que en el caso del algoritmo genético, en el CMA disminuye la velocidad de convergencia cuando disminuimos el número de generaciones (para un mismo número de iteraciones).

5.1.3 Verificación de resultados

Una vez realizadas las simulaciones mediante los 3 métodos (OptQuest, algoritmo genético y estrategia evolutiva CMA), pasamos a verificar que los resultados obtenidos en ellas son fiables. Para ello, son seleccionados las mejores configuraciones de cada algoritmo y se crea un escenario para cada una de ellas. Como en el caso del algoritmo genético y el CMA coincide la mejor solución obtenida, se verificarán las dos primeras, tal y como se indica en la Tabla 8:

Tabla 8: Escenarios de verificación para el Caso Base

ID Escenario	Processor1	Proceessor2	Processor3
1	10	7	6
2	9	7	7
3	8	7	7
4	9	7	6

Posteriormente, se ejecutarán esos escenarios con un número de réplicas de 10.000, obteniéndose como resultado el mostrado en la Figura 30.

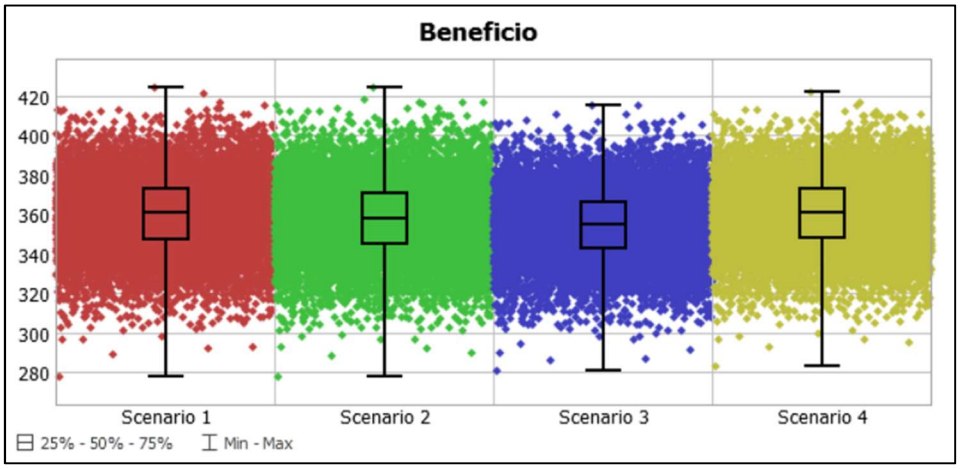

Figura 30: Resultados de la verificación del Caso Base

Analizando los resultados, se concluye con que las configuraciones de las capacidades máximas en todos los escenarios son muy similares, por lo que la elección de cualquiera de ellos daría lugar a unos resultados de función objetivo prácticamente idénticos.

Una vez comprendida la metodología en el ejemplo base, pasamos a analizar un modelo de mayor complejidad, que será más parecido a lo que podría darse en un proceso real.

5.2 Segundo caso: siete procesadores en serie con colas

En esta ocasión, serán 7 el número de procesadores que compondrán el sistema, con sus respectivas colas a sus entradas.

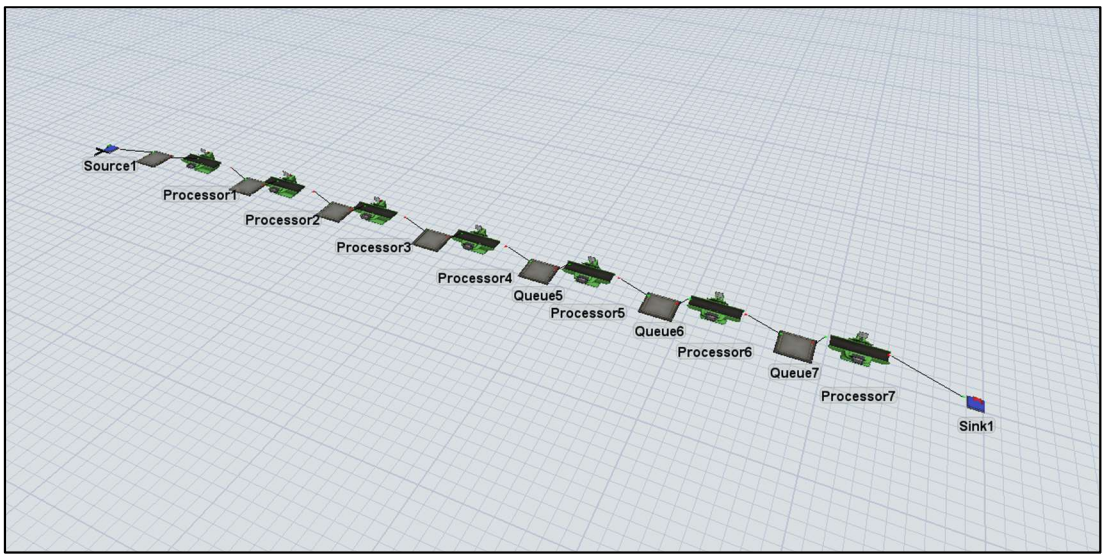

Figura 31: Imagen del segundo modelo en 3D

Para aumentar la variabilidad del proceso, los tiempos de ciclo de los procesadores han sido configurados mediante una hoja Excel, en la cual se han seleccionado aleatoriamente unos valores comprendidos entre 2 y 10 para la media de dicho tiempo de procesamiento. El valor que se ha establecido para la desviación estándar es del 50% de la media para cada procesador.

Tabla 9: Media y desviación típica del tiempo de procesamiento

Processor	1	2	3	4	5	6	7
Media	2	7	2	2	6	10	4
Desviación	1,0	3,5	1,0	1,0	3,0	5,0	2,0

Una vez establecidos los tiempos, el objetivo a realizar para este caso también es el de optimizar las distintas capacidades. La diferencia con el caso anterior (además de haber aumentado significativamente el número de procesadores) reside en que ahora también será necesario establecer la capacidad óptima de los buffers o colas que los preceden, por lo que la complejidad de la optimización será mayor.

Además, como era de esperar, estas nuevas capacidades también van ligadas a un coste, ya que no supondrá un mismo coste disponer de un buffer de capacidad 2 que uno de capacidad 10. Por lo tanto, es necesario añadir un nuevo término a la función

objetivo: el coste por unidad de capacidad de buffer. Dicha función es muy similar a la establecida en el caso base:

$$Beneficio = 1 \cdot Productos - 50 \cdot \sum Cap_{Proc} - 30 \cdot \sum Cap_{Buffe}$$

Donde:

- **Productos:** Número de ítems que han sido enviados al sumidero al finalizar la simulación
- $\sum \boldsymbol{Cap_{Proc}}$**:** Suma de las capacidades máximas de los procesadores
- $\sum \boldsymbol{Cap_{Buffer}}$**:** Suma de las capacidades máximas de las colas

Puede observarse como los coeficientes que multiplican a las capacidades han aumentado considerablemente. Esto es debido a que en este caso se simulará un tiempo mayor (10.000seg). Para que los valores posibles de la función beneficio estén comprendidos en zonas positivas y negativas, se ha modificado la distribución exponencial que rige el tiempo entre llegadas de los ítems, que en este caso es de 3 segundos.

Cabe destacar que la cola iniciar se considera de capacidad infinita y, por lo tanto, no será necesaria su optimización.

5.2.1 Optimización con OptQuest

De manera similar a lo ya realizado en el primer caso, se procede a definir las variables de decisión de nuestra función objetivo, tal y como muestra la Figura 32.

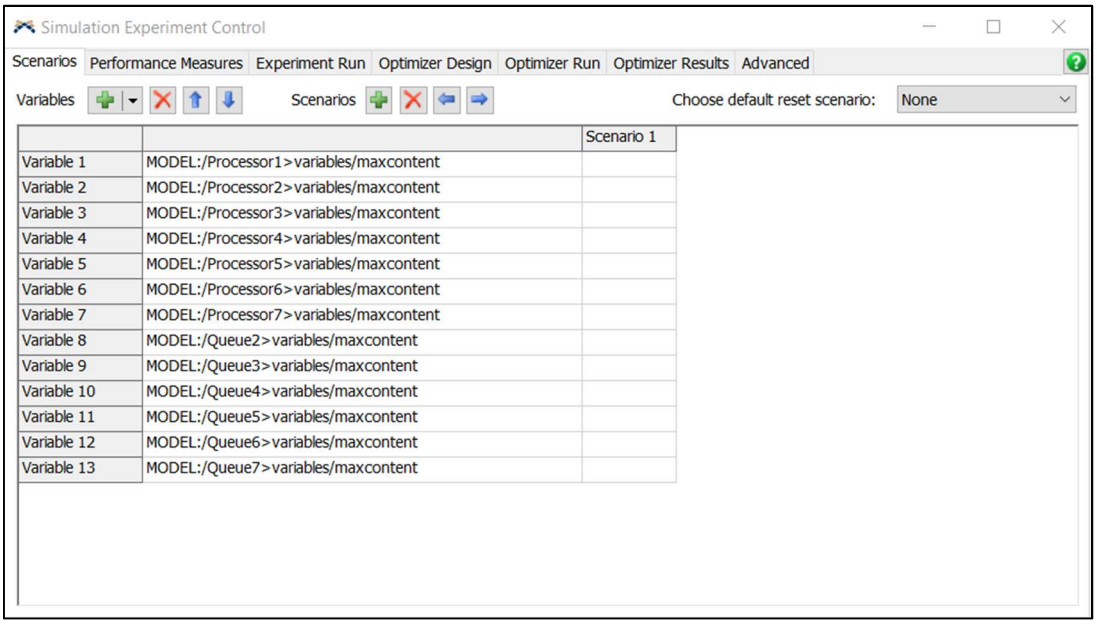

Figura 32: Configuración de las variables de decisión para el Caso 2.

La función objetivo también ha de ser modificada, tal y como se muestra en la figura 33.

```
1 /**Custom Code*/
2 treenode datanode = param(1);
3
4 double beneficio = 0;
5
6 beneficio = beneficio - Model.find("Processor1>variables/maxcontent").value*50;
7 beneficio = beneficio - Model.find("Processor2>variables/maxcontent").value*50;
8 beneficio = beneficio - Model.find("Processor3>variables/maxcontent").value*50;
9 beneficio = beneficio - Model.find("Processor4>variables/maxcontent").value*50;
10 beneficio = beneficio - Model.find("Processor5>variables/maxcontent").value*50;
11 beneficio = beneficio - Model.find("Processor6>variables/maxcontent").value*50;
12 beneficio = beneficio - Model.find("Processor7>variables/maxcontent").value*50;
13 beneficio = beneficio - Model.find("Queue2>variables/maxcontent").value*30;
14 beneficio = beneficio - Model.find("Queue3>variables/maxcontent").value*30;
15 beneficio = beneficio - Model.find("Queue4>variables/maxcontent").value*30;
16 beneficio = beneficio - Model.find("Queue5>variables/maxcontent").value*30;
17 beneficio = beneficio - Model.find("Queue6>variables/maxcontent").value*30;
18 beneficio = beneficio - Model.find("Queue7>variables/maxcontent").value*30;
19
20 beneficio = beneficio + Model.find("Sink1").as(Object).stats.input.value;
21
22 return beneficio;
```

Figura 33: Configuración de la función objetivo a optimizar para el Caso 2.

Una vez definidas las variables de decisión y la función objetivo, procedemos a efectuar la optimización. De nuevo seleccionaremos 1000 soluciones de 10 réplicas cada una, utilizando un tipo de búsqueda singular. La mejor configuración obtenida se muestra en la Tabla 10, mientas que todos los valores analizados se pueden ver en la Figura 34.

Tabla 10: Mejor configuración obtenida con OptQuest para el Caso 2

Solution ID	765
Rank	1
Variable 1	3
Variable 2	4
Variable 3	4
Variable 4	3
Variable 5	2
Variable 6	3
Variable 7	3
Variable 8	1
Variable 9	1
Variable 10	1
Variable 11	1
Variable 12	1
Variable 13	1
Beneficio	2066,6

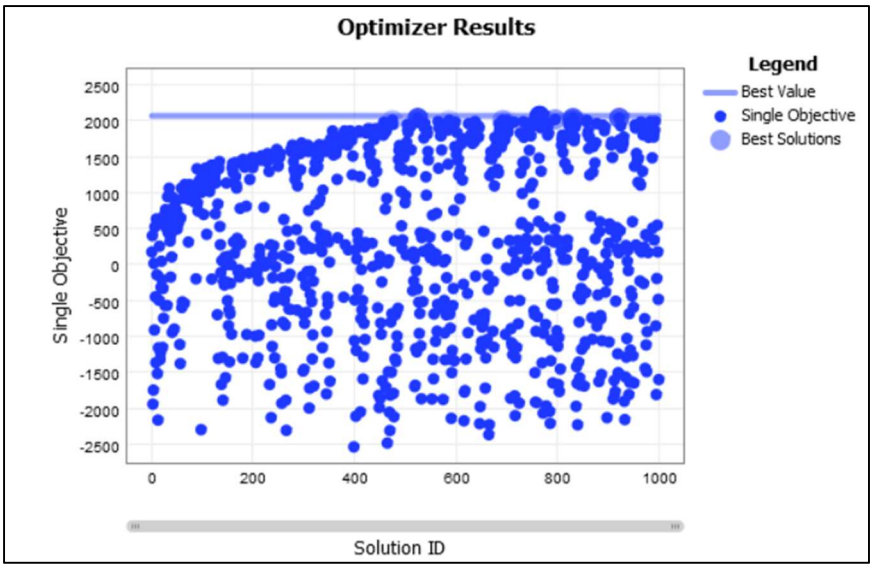

Figura 34: Resultado de optimización del Caso 2 con OptQuest

Igual que en el caso anterior, existe una elevada aleatoriedad a la hora de buscar una solución óptima, ya que la dispersión entre los resultados es aproximadamente la misma a lo largo de toda la simulación. Pese a ello, puede observarse como OptQuest tiene una tendencia desde el comienzo a acercarse a la zona óptima del horizonte de simulación. Debido a la mayor dificultad del problema, la solución óptima se alcanza en la iteración 765, frente a la 129 que se había obtenido en el caso base.

5.2.2 Optimización con HeuristicLab

5.2.2.1 Implementación de algoritmo genético

Se procede ahora a realizar la optimización mediante el uso del algoritmo genético. Para ello, serán ejecutadas diversas simulaciones variando distintos parámetros del algoritmo genético: la población, la tasa de mutación y el número de generaciones.

Tabla 11: Optimizaciones realizadas con algoritmo genético en el Caso 2

ID Simulación	Generaciones	Población	% Mutación
1	100	10	10
2	100	10	5,0
3	100	10	2,5
4	50	20	10
5	50	20	5,0
6	50	20	2,5
8	20	50	10
9	20	50	5,0
10	20	50	2,5
11	10	100	10
12	10	100	5,0
13	10	100	2,5

Una vez realizadas todas, obtenemos el gráfico mostrado en la Figura 35.

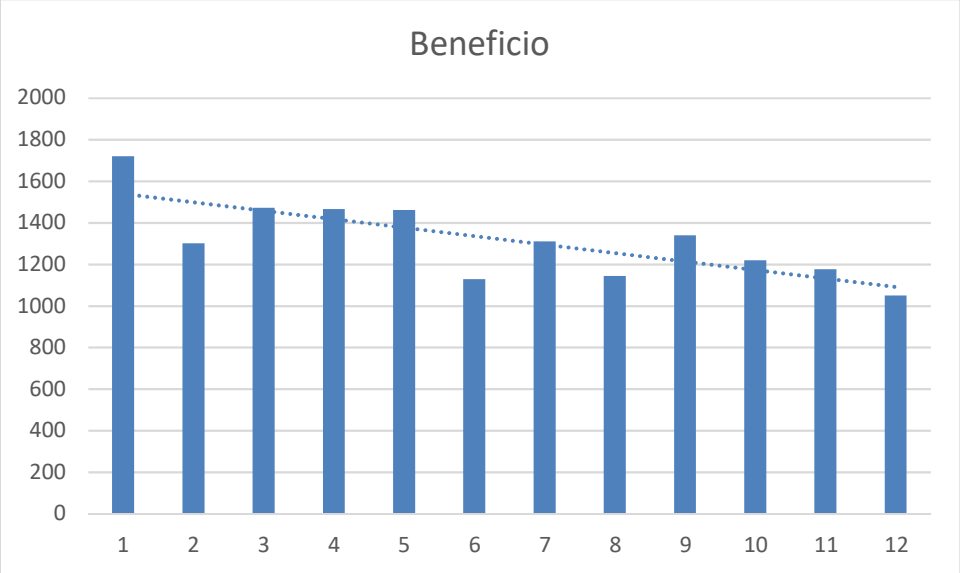

Figura 35: Resultados de las simulaciones con algoritmo genético en el Caso 2

Los resultados obtenidos son muy similares a los del caso anterior. En el gráfico, es apreciable la influencia de la relación generaciones/población, ya que la calidad de la solución aumenta de forma proporcional a esta. Por otro lado, pese a que existe un factor de aleatoriedad importante debido a la desviación de los tiempos de procesado, también existe cierta tendencia a obtener mejores resultados a medida que aumenta la tasa de mutación. Se concluye por lo tanto que la mejor solución obtenida mediante algoritmo genético es la 1, la cual está mostrada en la Tabla 12.

Tabla 12: Mejor configuración obtenida con algoritmo genético en el Caso 2

ID Simulación	**1**
Processor1	3
Processor2	4
Processor3	5
Processor4	2
Processor5	3
Processor6	3
Processor7	3
Queue2	5
Queue3	1
Queue4	3
Queue5	1
Queue6	3
Queue7	3
Beneficio	**1725**

La convergencia de esta solución tiene una forma muy similar a la vista en el caso anterior:

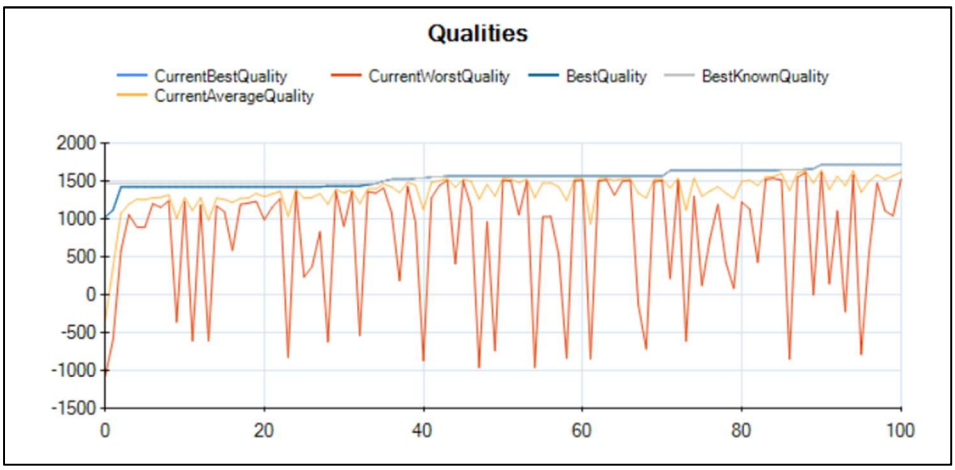

Figura 36: Solución 1 con algoritmo genético en el Caso 2

De nuevo puede observarse como la elevada tasa de mutación (10%) hace que el algoritmo explore constantemente nuevas zonas del horizonte de simulación, lo que permite que las soluciones pertenecientes a zonas de máximos locales evolucionen mientras que otras soluciones exploren aleatoriamente nuevas zonas del horizonte en busca de otros máximos.

5.2.2.2 Implementación de estrategia evolutiva

En este caso, debido a la mayor complejidad de la optimización, será probado un nuevo parámetro de la estrategia evolutiva: la sigma inicial. Ésta determina la desviación estándar que tendrán las soluciones iniciales del algoritmo, lo que permite configurar el rango de soluciones desde el comienzo de la optimización. Así, los parámetros a variar serán las generaciones simuladas, la población de cada una de ellas, y la sigma inicial. Serán omitidas de este caso las relaciones bajas de generaciones/población, debido a los peores resultados que se han obtenido en experimentos anteriores. En su lugar, será probada una nueva configuración en la que la relación generaciones/población aumenta hasta 40:

Tabla 13: Optimizaciones realizadas con CMA en el Caso 2

ID Optimización	Generaciones	Población	Sigma Inicial
1	200	5	0,75
2	200	5	0,5
3	200	5	0,25
4	100	10	0,75
5	100	10	0,5
6	100	10	0,25

Una vez realizadas las optimizaciones, se muestra la mejor solución en la Tabla 14:

Tabla 14: Configuración de mejor resultado obtenido con CMA en el Caso 2

ID Simulación	6
Processor1	3
Processor2	4
Processor3	4
Processor4	2
Processor5	2
Processor6	3
Processor7	3
Queue2	1
Queue3	1
Queue4	1
Queue5	2
Queue6	1
Queue7	1
Beneficio	**2090**

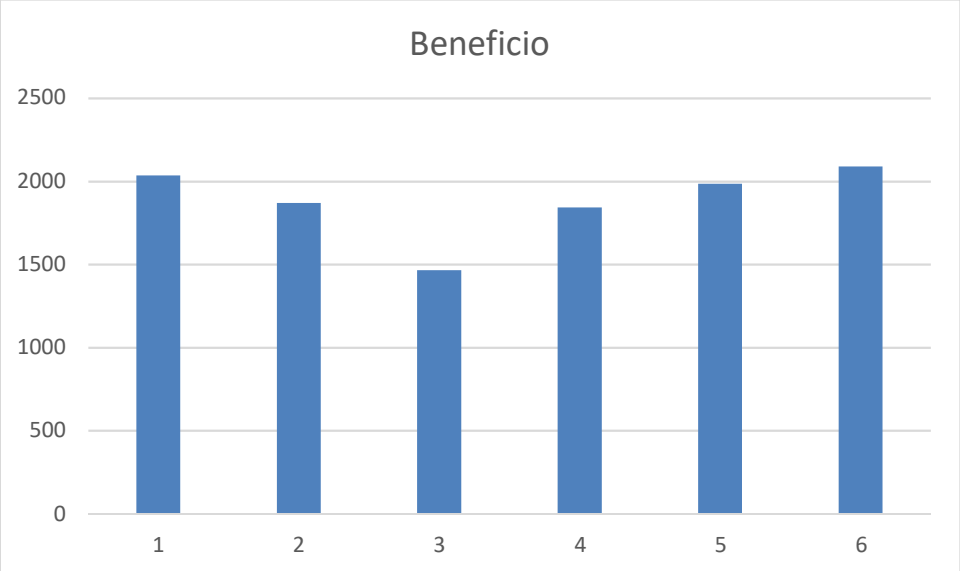

Figura 37: Resultados de optimización con CMA en el Caso 2

Como se puede apreciar en la Figura 37, la desviación estándar inicial tiene efectos contrarios en las dos relaciones generaciones/población. En las 3 primeras soluciones, la calidad obtenida disminuye a medida que aumenta dicha desviación, mientras que en las soluciones 4, 5 y 6 aumenta. Manteniendo constante la relación generaciones/ población y variando la desviación estándar inicial, se introducen los gráficos de convergencia de cada una de las soluciones:

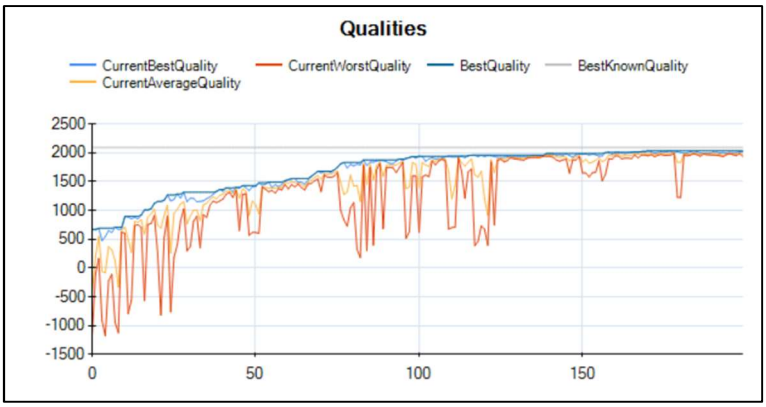

Figura 38: Solución 1 con CMA en el Caso 2

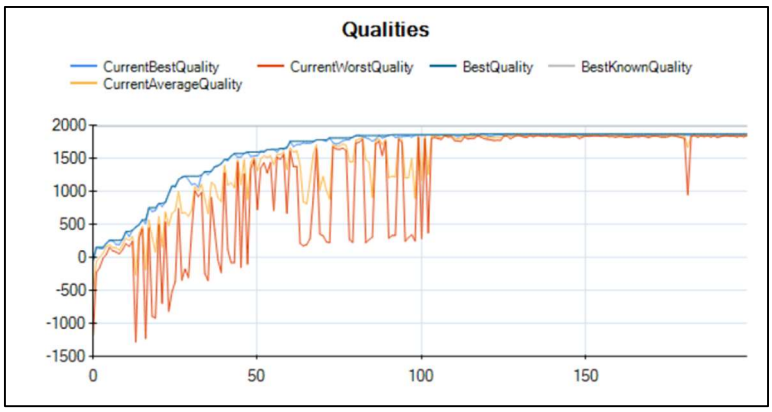

Figura 39: Solución 2 con CMA en el Caso 2

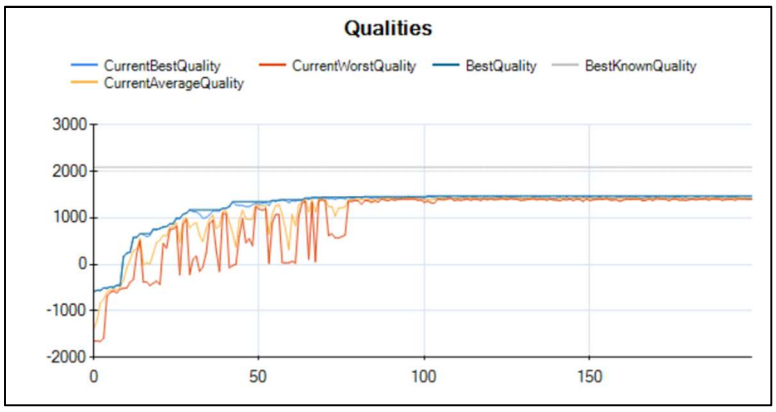

Figura 40: Solución 3 con CMA en el Caso 2

Como se ha indicado con anterioridad, en el caso de 200 generaciones de población 5, a medida que disminuye la desviación estándar inicial se obtienen peores resultados. De una forma similar a lo que se ha visto en la influencia de la tasa de mutación en el algoritmo genético, la desviación inicial influye en la evolución de las soluciones explorando una mayor o menor zona del horizonte de búsqueda. Además, cuanto menor sea la sigma inicial, antes se alcanza la convergencia, lo que aumenta la probabilidad de que se produzca una convergencia prematura en máximos locales de la función objetivo.

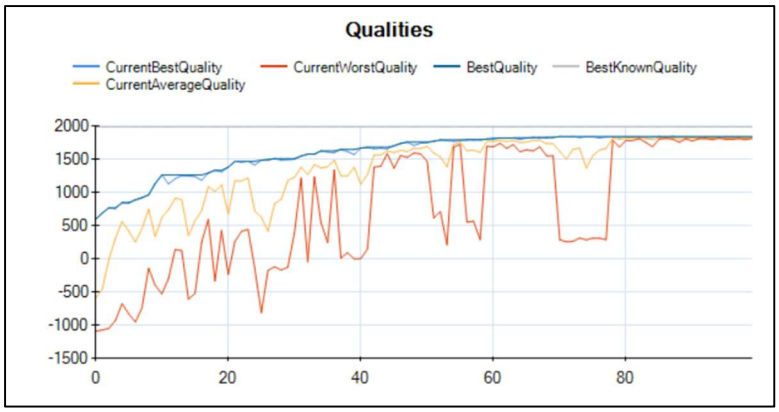

Figura 41: Solución 4 con CMA en el Caso 2

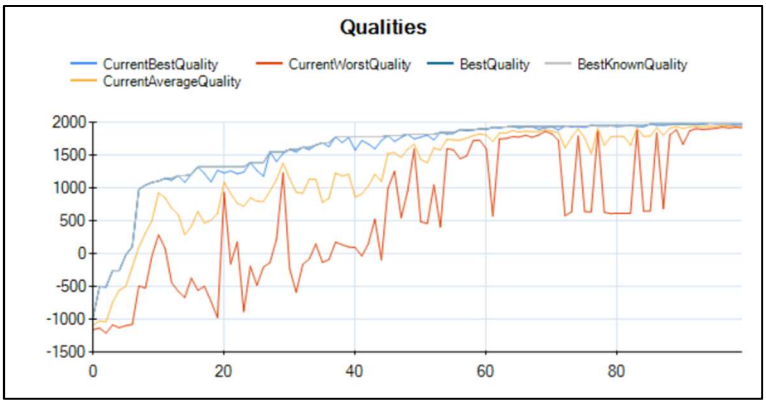

Figura 42: Solución 5 con CMA en el Caso 2

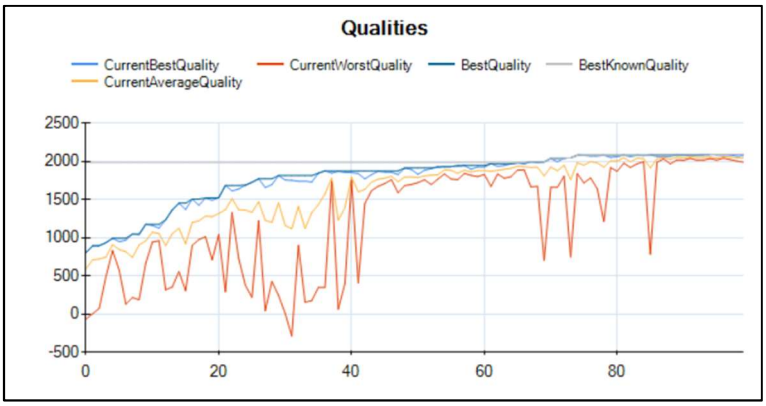

Figura 43: Solución 6 con CMA en el Caso 2

Analizando ahora los resultados obtenidos en el caso de 100 generaciones de 10 elementos de población, destaca que la desviación estándar inicial influye de manera opuesta a como lo hacía con una relación generaciones/población de 40. A medida que dicha desviación disminuye, el algoritmo proporciona mejores resultados. De igual forma que en los tres primeros casos, en las soluciones 4, 5 y 6 puede apreciarse como la evolución del algoritmo hacia la mejor solución abarca un mayor rango de soluciones cuanto mayor sea la desviación inicial.

5.2.3 Verificación de resultados

Para proceder a la verificación de los resultados obtenidos, se han generado 3 escenarios con las mejores soluciones obtenidas por OptQuest, el algoritmo genético y la estrategia evolutiva CMA.

Tabla 15: Configuración de escenarios de verificación en el Caso 2

ID Escenario	1	2	3
Variable 1	3	3	3
Variable 2	4	4	4
Variable 3	4	5	4
Variable 4	3	2	2
Variable 5	2	3	2
Variable 6	3	3	3
Variable 7	3	3	3
Variable 8	1	5	1
Variable 9	1	1	1
Variable 10	1	3	1
Variable 11	1	1	2
Variable 12	1	3	1
Variable 13	1	3	1

Donde el escenario 1 es el obtenido con OptQuest, el 2 con el algoritmo genético y el 3 con la estrategia evolutiva CMA. Todos ellos son ejecutados un numero de réplicas de 10.000, obteniendo como resultado:

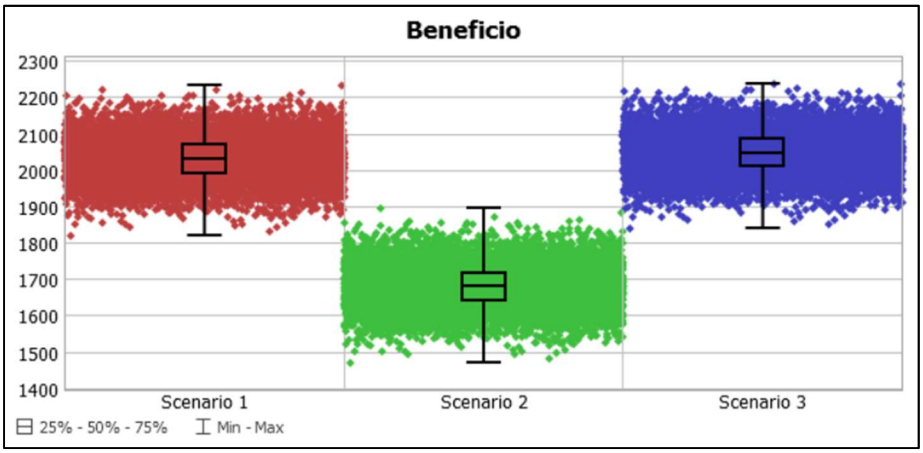

Figura 44: Resultados de la verificación del Caso 2

En la Figura 44 se confirma que los resultados que hemos obtenido son fiables, ya que se encuentran dentro del intervalo de confianza. Por otro lado, los resultados obtenidos con OptQuest y con CMA son muy similares, mientras que la configuración obtenida con el algoritmo genético es claramente inferior, ya que el mejor resultado obtenido en 10.000 réplicas es similar al peor en cualquiera de los otros dos escenarios.

5.3 Tercer caso: taller de nudos proporcionado por el GII

El tercer modelo sobre el cual será realizada la optimización ha sido proporcionado por el GII. Este se muestra en la Figura 45.

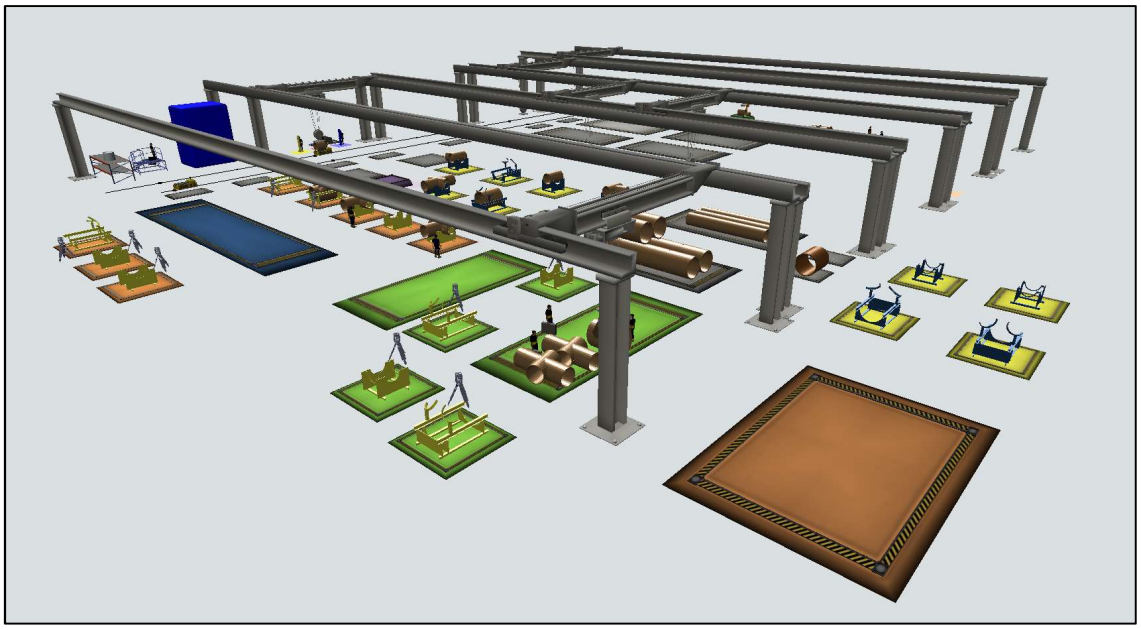

Figura 45: Imagen del tercer modelo en 3D

Se trata de un proyecto real en el cual se está trabajando para implementarlo en un futuro. Este representa el proceso que se sigue en un taller de elaboración de nudos de soldadura en barras para así formar celosías. El número de nudos necesarios para formar el producto final es de 12, por lo que habrá que ajustar el tiempo de ciclo de la suma de todos ellos. El tiempo de ciclo deseado es de 80 horas para el producto terminado, por lo que cada nudo debe cumplir un tiempo de ciclo de 80/12, que es de 6,67 horas por nudo.

El modelo depende de numerosas variables, por lo tanto, se han seleccionado las más representativas para la elaboración de la función objetivo. Dichas variables se muestran en la Tabla 16.

Una vez establecidas las variables de decisión, es elaborada la función objetivo:

$$Calidad = nºOperarios + \alpha_1 \cdot nºCamas + \alpha_2 \cdot \sum Cap_{Buffer}$$

Donde:

- nº Operarios: total de operarios implicados en el proceso.
- nº Camas: suma de todas las camas del proceso.
- $\sum \boldsymbol{Cap_{Buffer}}$: sumatorio de las capacidades máximas de las colas del proceso.
- $\boldsymbol{\alpha_1}$: relación entre el coste que supone un operario y el que supone una cama.

- $\boldsymbol{\alpha_2}$: relación entre el coste que supone un operario y el que supone una unidad de capacidad en una cola.
- t_{ciclo}: tiempo de ciclo de un nudo, en días naturales.

Tabla 16 : Variables de decisión en el Caso 3

Variables de decisión	Mínimo	Máximo
Número de camas de ensamblado Y	1	4
Número de camas de ensamblado X	1	4
Número de camas de CD1Y	1	4
Número de camas de CD1X	1	4
Número de camas de CD2Y	1	4
Número de camas de CD2X	1	4
Número de camas de CD3Y	1	4
Número de camas de CD3X	1	4
Soldadores	1	4
Parejas de inspectores	1	4
Parejas de ensayistas	1	4
Gruistas	1	4
Carros	1	2
Capacidad de colas de zona de espera a soldar	1	10
Capacidad de colas de zona de espera a CD3	1	10

En caso de que el tiempo de ciclo sea superior al establecido, la función objetivo tendrá una penalización proporcional a la diferencia:

$$Calidad = nºOperarios + \alpha_1 \cdot nºCamas + \alpha_2 \cdot \sum Cap_{Buffer} + (t_{ciclo} - \frac{80}{12}) \cdot 100$$

Análogamente, si el tiempo de ciclo es menor, también se aplica una penalización, pero en menor medida:

$$Calidad = nºOperarios + \alpha_1 \cdot nºCamas + \alpha_2 \cdot \sum Cap_{Buffer} + (\frac{80}{12} - t_{ciclo}) \cdot 5$$

Para calcular los coeficientes $\boldsymbol{\alpha_1}$ y $\boldsymbol{\alpha_2}$, serán utilizados los datos proporcionados por la UMI:

$$\boldsymbol{\alpha_1} = \frac{Coste\ cama}{Coste\ operario} = 0{,}7815$$

$$\boldsymbol{\alpha_2} = \frac{Coste\ capacidad\ cola}{Coste\ operario} = 0{,}078125$$

Al contrario que en los casos anteriores, en esta ocasión el objetivo reside en minimizar la función objetivo, ya que de esta forma alcanzaríamos la configuración que permita cumplir los plazos de producción con los menores costes.

Una vez definidas todas las variables y la función objetivo, será realizada la optimización del modelo.

5.3.1 Optimización con OptQuest

Igual que en los anteriores casos, en primer lugar, se definen las variables de decisión tal y como se muestra en la Figura 46.

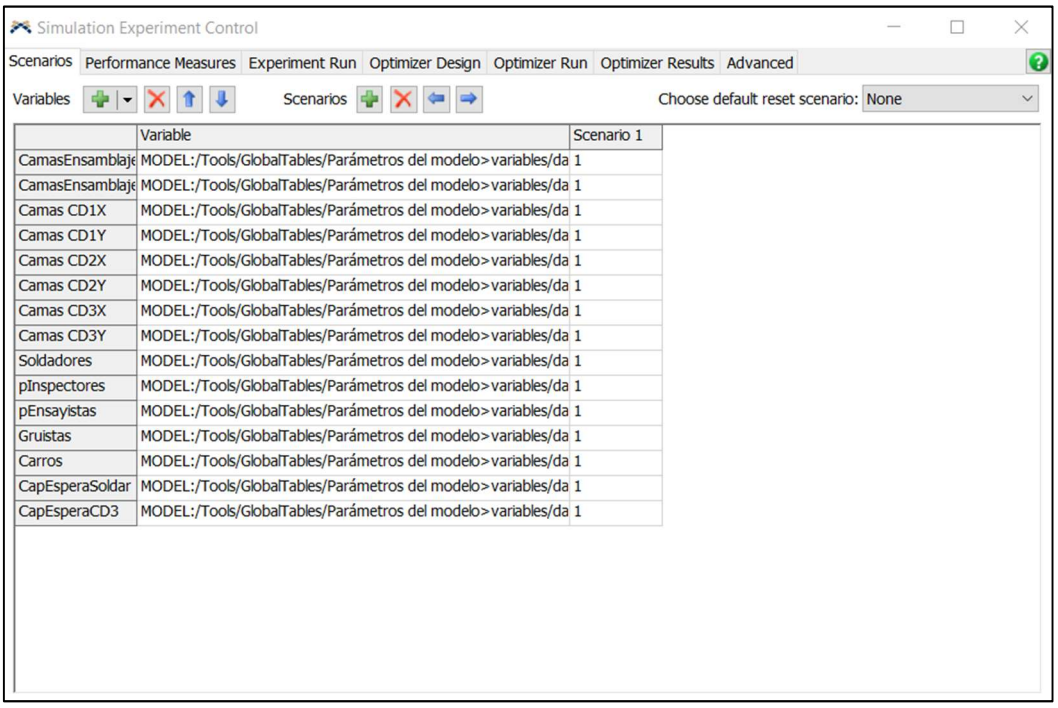

Figura 46: Configuración de las variables de decisión para el Caso 3

Una vez establecidas las variables de decisión, se define la función objetivo, la cual depende de todas las variables de decisión, además del tiempo de ciclo de los nudos, como se muestra en la Figura 47.

```
/**Custom Code*/
treenode datanode = param(1);

double calidad = 0;
double tciclo;
//Número de camas
calidad = calidad + Table("Parámetros del modelo")[1][2]*0.7815; //EENSAMBLAJE Y
calidad = calidad + Table("Parámetros del modelo")[2][2]*0.7815; //ENSAMBLAJE X
calidad = calidad + Table("Parámetros del modelo")[3][2]*0.7815; //CD1Y
calidad = calidad + Table("Parámetros del modelo")[4][2]*0.7815; //CD1X
calidad = calidad + Table("Parámetros del modelo")[5][2]*0.7815; //CD2Y
calidad = calidad + Table("Parámetros del modelo")[6][2]*0.7815; //CD2X
calidad = calidad + Table("Parámetros del modelo")[7][2]*0.7815; //CD3Y
calidad = calidad + Table("Parámetros del modelo")[8][2]*0.7815; //CD3X

//Número de operarios
calidad = calidad + Table("Parámetros del modelo")[9][2]; //Soldadores
calidad = calidad + Table("Parámetros del modelo")[10][2]*2; //Inspectores
calidad = calidad + Table("Parámetros del modelo")[12][2]*2; //Ensayistas
calidad = calidad + Table("Parámetros del modelo")[13][2]; //Gruistas
calidad = calidad + Table("Parámetros del modelo")[14][2]; //Carros

//Capacidades máximas de colas
calidad = calidad + Table("Parámetros del modelo")[24][2]*0.078125; //Espera a soldar
calidad = calidad + Table("Parámetros del modelo")[25][2]*0.078125; //Espera a CD3

//Tiempo de ciclo
tciclo = TrackedVariable("Nudos").average;
if (tciclo > 80/12)
        {
            double dif = tciclo - 80/12;
            calidad += dif * 100;
        }
        if (tciclo < 80/12)
        {
            double dif = 80/12 - tciclo;
            calidad += dif * 5;
        }

return calidad;
```

Figura 47: Configuración de la función objetivo a optimizar para el Caso 3

Cabe destacar que las variables correspondientes a los inspectores y ensayistas están multiplicadas por 2, debido a que se tratan de parejas.

Igual que en los casos anteriores, realizaremos una optimización de 1000 soluciones, pero en este caso tan sólo realizaremos 1 réplica. Esto es debido a que en cada simulación se realizará un tiempo de 3000 horas, el cual es más que suficiente para establecer una media de tiempos de ciclo fiable. Los resultados de la simulación con OptQuest de forma gráfica pueden verse en la Figura 48, y la mejor configuración obtenida se define en la Tabla 17.

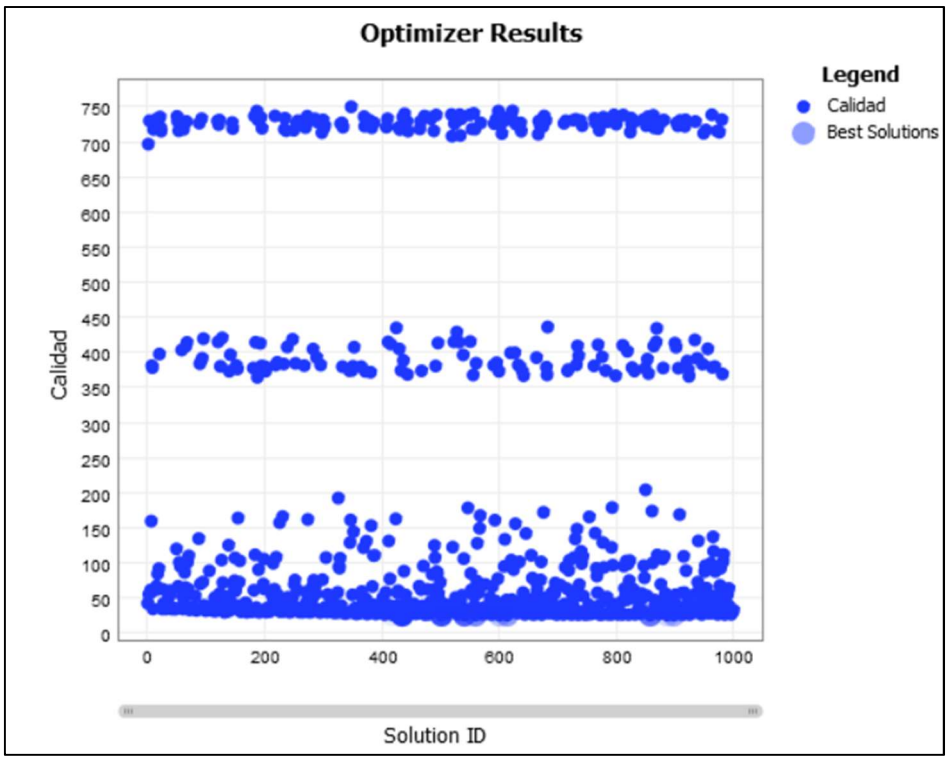

Figura 48: Resultado de optimización del Caso 3 con OptQuest

Tabla 17: Mejor configuración obtenida con OptQuest en el Caso 3

Solution ID	435
CamasEnsamblajeY	1
CamasEnsamblajeX	2
Camas CD1X	1
Camas CD1Y	1
Camas CD2X	2
Camas CD2Y	3
Camas CD3X	1
Camas CD3Y	1
Soldadores	1
pInspectores	2
pEnsayistas	3
Gruistas	2
Carros	1
CapEsperaSoldar	1
CapEsperaCD3	1
Calidad	25,039382
Tiempo de ciclo Nudos	6,36564

Cabe destacar que el proceso ha sido mucho más lento que en los casos anteriores (en torno a 4 horas), ya que además de ser una función objetivo más compleja, el modelo tiene muchos más elementos, lo que dificulta la tarea de optimización.

En la Figura 48 puede apreciarse la diferencia existente entre las soluciones que están cerca de cumplir el tiempo de ciclo y las que no, tal y como se ha establecido en la función objetivo.

5.3.1 Optimización con HeuristicLab

En esta ocasión la tarea de conectar ambos programas se complica. Habrá que enviar los datos de las variables de decisión en varios vectores, ya que cada tipo de variable tiene unos límites específicos. Además, igual que en el caso de optimización con OptQuest, tan solo será simulada 1 réplica por solución. Este hecho no repercutirá en exceso sobre la solución final, ya que, pese a la complejidad del modelo, la desviación existente no es muy acusada.

Por otro lado, dados los resultados obtenidos en las anteriores optimizaciones, en esta ocasión tan solo se realizará la optimización con la estrategia evolutiva CMA, ya que ha otorgado unos resultados notablemente mejores que el algoritmo genético.

5.3.1.1 Implementación de estrategia evolutiva CMA

Es de destacar que en este caso el CMA tiene una desventaja, ya que no permite la implementación de múltiples vectores para su evaluación. Para solventar esta carencia, se ha generado un único vector el cual está formado por todas las variables de decisión. Dicho vector tiene como límites inferior y superior 0 y 1, respectivamente. Es en el script de FlexSim donde estos valores serán interpolados para así adaptarse a los límites de cada variable.

Realizando una simulación de 100 generaciones con una población de 10 y una tasa de mutación del 5%, se obtiene el resultado mostrado en la Tabla 18.

Tabla 18: Mejor configuración obtenida con CMA en el Caso 3

CamasEnsamblajeY	1
CamasEnsamblajeX	3
Camas CD1X	1
Camas CD1Y	2
Camas CD2X	1
Camas CD2Y	3
Camas CD3X	1
Camas CD3Y	2
Soldadores	1
pInspectores	1
pEnsayistas	3
Gruistas	2
Carros	1
CapEsperaSoldar	2
CapEsperaCD3	3
Calidad	23

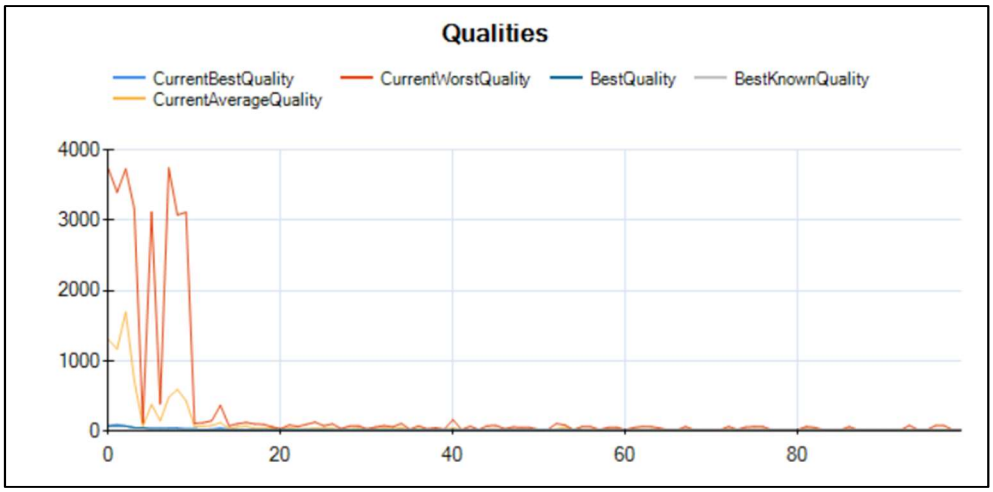

Figura 49: Convergencia de la solución con CMA para el Caso 3

Como se puede observar en la Figura 49, la estrategia evolutiva alcanza rápidamente (en torno a la generación 10) valores próximos a los óptimos, obteniéndose al final de la optimización un valor de la función calidad de 23. Se procederá en el siguiente apartado a la verificación de los resultados obtenidos.

5.3.2 Verificación de resultados

Como se ha hecho en los casos anteriores, en este apartado se procederá a realizar réplicas múltiples de las mejores soluciones obtenidas en la optimización, en este caso mediante OptQuest y la estrategia evolutiva CMA. Las configuraciones de las variables de decisión obtenidas en ambos casos son las mostradas en la Tabla 19.

Tabla 19: Configuración de escenarios de verificación en el Caso 3

ID Escenario	1	2
CamasEnsamblajeY	1	1
CamasEnsamblajeX	2	1
Camas CD1X	1	1
Camas CD1Y	1	1
Camas CD2X	2	1
Camas CD2Y	3	1
Camas CD3X	1	1
Camas CD3Y	1	1
Soldadores	1	1
pInspectores	2	1
pEnsayistas	3	1
Gruistas	2	1
Carros	1	1
CapEsperaSoldar	1	1
CapEsperaCD3	1	1
Calidad	25,039382	23

Donde el escenario 1 se corresponde con la solución obtenida con OptQuest y escenario 2 con CMA. Tras la realización de 1000 réplicas en el experimentador de FlexSim se obtienen los resultados mostrados en la Figura 50 y 51.

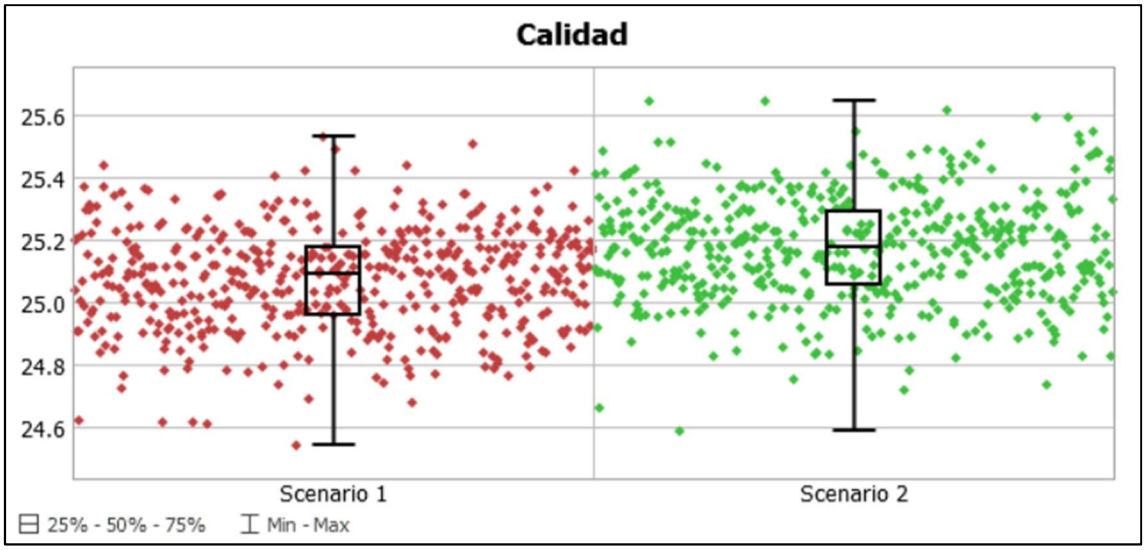

Figura 50: Resultados de la verificación del Caso 3: Calidad

Como se puede observar en la figura 49, ambos resultados obtenidos otorgan un valor de la función calidad en torno a 25, siendo la obtenida mediante CMA ligeramente superior.

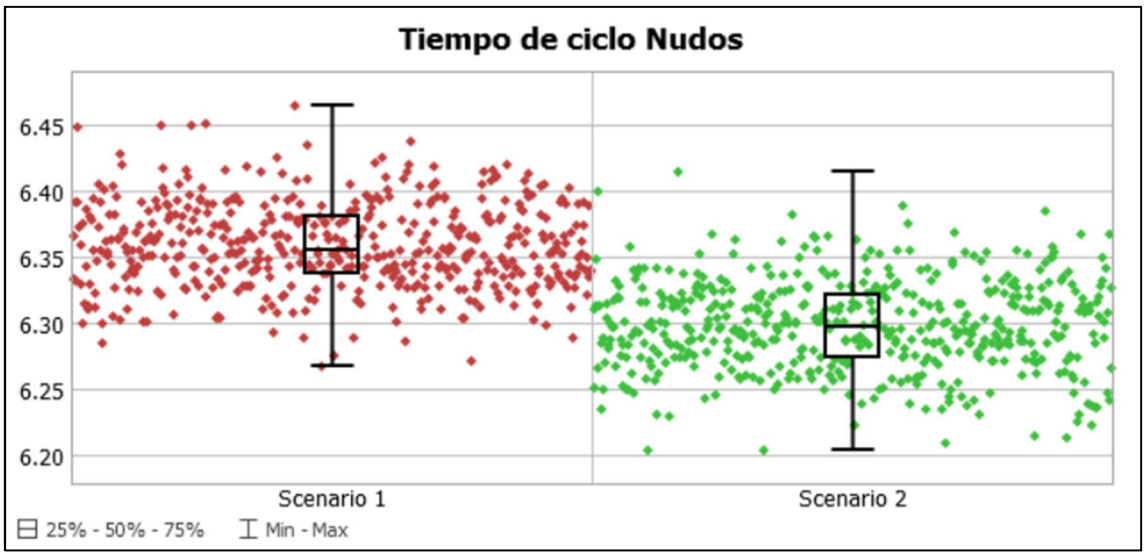

Figura 51: Resultados de la verificación del Caso 3: Tiempo de ciclo

Por otro lado, observando el gráfico del tiempo de ciclo, puede apreciarse como el escenario 2 da como resultado un tiempo de ciclo adelantado con respecto al tiempo de ciclo objetivo (6,67). Esto también ocurre en el escenario 1, aunque en menor medida, por lo que la función objetivo tiene mejores resultados en este. No obstante, en ambos

casos se cumplen de una forma aceptable los tiempos de ciclo de los nudos, por lo que la solución es válida.

Una vez comprobada la validez de los resultados obtenidos, se calcula la función de calidad sin tener en cuenta las penalizaciones por tiempo, de modo que se obtiene un valor asociado al coste que supone cada una de las configuraciones obtenidas.

Así, para el escenario 1, obtenido con OptQuest:

$$Calidad_1 = nºOperarios + \alpha_1 \cdot nºCamas + \alpha_2 \cdot \sum Cap_{Buffer} = 23{,}5345$$

Y para el escenario 2, obtenido con CMA:

$$Calidad_2 = nºOperarios + \alpha_1 \cdot nºCamas + \alpha_2 \cdot \sum Cap_{Buffer} = 23{,}3316$$

Por lo tanto, en términos de costes, el escenario obtenido mediante el uso de la estrategia evolutiva CMA es mejor, permitiendo además cumplir los tiempos de ciclo en un tiempo menor.

6 CONCLUSIONES Y TRABAJO FUTURO

Una vez analizados todos los casos propuestos en el presente proyecto, se procede a analizar los resultados obtenidos con el método de evaluación externa. Además, se propondrán una serie de trabajos futuros a realizar a partir de este proyecto.

6.1 Conclusiones

Una vez realizadas las distintas optimizaciones se ha comprobado que el método de evaluación externa es funcional en diversos modelos de simulación.

Una de las ventajas del uso de algoritmos metaheurísticos de código abierto es la posibilidad de variar parámetros de estos para adaptarlos de la mejor forma posible al problema que se quiera optimizar. Como se ha comprobado en los casos analizados, los resultados obtenidos han cambiado notablemente tras variar los parámetros de los que dependen. Los parámetros comunes a ambos algoritmos son el número de generaciones y la población, debido a que ambos son algoritmos evolutivos. Además, en el algoritmo genético se ha variado la tasa de mutación y en la estrategia evolutiva la desviación estándar inicial, dando lugar a resultados diferentes en cada caso. Si bien esta versatilidad que presentan los algoritmos metaheurísticos es en esencia una ventaja, tiene como contrapartida que para alcanzar los valores óptimos de los parámetros modificables es necesaria la realización de múltiples optimizaciones para poder comparar los resultados obtenidos. Por lo tanto, en el caso de hacer uso de algoritmos metaheurísticos, el tiempo requerido para encontrar una solución satisfactoria se multiplica, ya que es necesario realizar pruebas con varios valores de los parámetros para así quedarnos con la mejor de todas ellas.

En el caso base, puede apreciarse como en modelos básicos otorgan buenos resultados los dos algoritmos empleados: el algoritmo genético y la estrategia evolutiva basada en la covarianza matricial (CMA).

Sin embargo, al aumentar la dificultad de la optimización, se comprueba que el algoritmo genético genera peores resultados que la estrategia evolutiva, tal y como se puede ver en el Caso 2.

Por último, se ha analizado un caso práctico proporcionado por el GII, el cual es un modelo real. Este hecho hace que cobre especial interés el método empleado en el presente proyecto, ya que se verifica que funciona con múltiples variables, lo que amplía considerablemente el rango de aplicación. Además, el modelo optimizado en el tercer caso es muy pesado, por lo que el tiempo de simulación es elevado y las optimizaciones se realizan de forma lenta. Este tiempo podría verse reducido en el caso de realizar una mejor calibración del algoritmo y de la función objetivo, tarea que se propone para la realización de trabajos futuros.

6.2 Trabajo futuro

En último lugar, y tras observar los resultados analizados en el presente proyecto, se proporcionan una serie de posibles trabajos futuros que se consideran de interés de cara a ampliar el trabajo realizado.

- Con el fin de agilizar la búsqueda de los mejores parámetros de los algoritmos, sería conveniente el desarrollo de un código que permita generar numerosas optimizaciones consecutivas, sin la necesidad de ejecutar cada una de ellas por separado.
- También sería de interés el ampliar la librería de algoritmos presente en HeuristicLab. Sería conveniente desarrollar un tipo de algoritmo específico para casos de simulación de procesos industriales, ya que los empleados en HeuristicLab, pese a tener un funcionamiento comprobado, no son específicos para modelos de simulación.
- Por último, tal y como se ha mencionado en el apartado de conclusiones, sería conveniente, (especialmente en modelos pesados como el del taller de nudos analizado en este proyecto) realizar una calibración y análisis más detallado del algoritmo empleado para su posterior implementación, de tal forma que se reduciría el tiempo de optimización.

7 BIBLIOGRAFÍA

- Puche, J. F. (2005). Guía para la simulación de procesos industriales. agosto 20, 2019, de CETEM Sitio web: enlace al pdf

- Wilkin, D. y Akre, B. (23 de marzo, 2016). Influences on Darwin - Advanced (Lo que influyó sobre Darwin-avanzado). En CK-12 biology advanced concepts. Tomado de http://www.ck12.org/book/CK-12-Biology-Advanced-Concepts/section/10,18/.

- Trigo, V. (2016). Algoritmos. mayo 2, 2019, de www.vicentetrigo.com Sitio web: http://vicentetrigo.com/wp-content/uploads/2016/09/algoritmos.pdf

- Joyanes, L. (2008). Fundamentos de programación. Madrid: McGraw Hill.

- Guasch, A., Piera, M.À., Casanovas, J. & Figueras, J. (2004). Modelado y simulación. Aplicación a procesos logísticos de fabricación y servicios. Barcelona: Edicions UPC.

- Cantor, G. A., Gómez, J. (2008, diciembre 15). Asignación de parámetros en los algoritmos evolutivos. RE´TAKVN, I, pp.62-66.

- Anónimo (2019). Estrategia evolutiva. 10/10/2019, de Wikipedia Sitio web: https://es.wikipedia.org/wiki/Estrategia_evolutiva

- Anónimo (2019). CMA-ES. 11/10/2019, de Wikipedia Sitio web: https://en.wikipedia.org/wiki/CMA-ES

- Hansen, N. (2006), "The CMA evolution strategy: a comparing review", Towards a new evolutionary computation. Advances on estimation of distribution algorithms, Springer, pp. 1769–177