AJUSTE DE UN MODELO DE REDES DE CONTAGIO DE VARICELA MEDIANTE ALGORITMOS DE OPTIMIZACIÓN HEURÍSTICOS EN UN ENTORNO DE CÁLCULO DISTRIBUIDO

José Gabriel García Caro
E.T. S. I. Informática, Universidad Nacional de Educación a Distancia, España
jggarciacaro@gmail.com

Matilde Santos
Facultad de Informática, Universidad Complutense de Madrid, Madrid, España
msantos@ucm.es

Resumen

La búsqueda de los parámetros que mejor hacen evolucionar un modelo de redes aleatorias es una tarea computacionalmente muy costosa y en muchos casos inabordable mediante la evaluación exhaustiva de todas las soluciones. En este trabajo se va a modelar, mediante redes aleatorias, el comportamiento del virus de la varicela. Para el ajuste de este modelo se presenta una alternativa que combina algoritmos meta-heurísticos, computación distribuida y almacenamiento en la nube. Esta arquitectura permite la sinergia de elementos totalmente desacoplados (sistema de computación, modelo de redes y generador de soluciones) para distintas plataformas sin tener que modificarlos. El modelo resultante se ajusta bastante bien a los datos reales disponibles y permitirá tomar acciones para poner en marcha nuevas campañas de vacunación.

Palabras clave: Modelado, Redes aleatorias, Optimización heurística, MOPSO, Varicela.

1 INTRODUCCIÓN

La varicela es una enfermedad altamente contagiosa causada por el virus de la varicela-zóster (VZV) [8], de la familia de los herpesvirus. Aunque en la mayoría de los casos es una infección benigna, tiene una altísima prevalencia en todos los países del mundo. La tasa de transmisión de la varicela es variable según la época del año [1].

Debido a esto, sería muy interesante poder predecir el comportamiento de esta enfermedad para poder así tomar las medidas necesarias, por ejemplo, programar las campañas de vacunación determinando el mejor calendario, a cuánta gente hay que vacunar, con qué coste, etc.

Las ecuaciones diferenciales son una representación bien conocida y potente [7] para estudiar la dinámica de muchos sistemas pero, cuando se utilizan en el entorno de un modelo epidemiológico, tienen como principal inconveniente las limitaciones para distinguir entre individuos concretos [2]. Introducir en el modelo elementos tales como edad, sexo, enfermedades previas, etc., se convierte en algo muy complejo. Por eso en este trabajo se ha optado por usar modelos de redes. En los últimos años las redes aleatorias se han popularizado como medio para simular los patrones de difusión de enfermedades infectocontagiosas en redes de gran tamaño [4,10].

Hasta ahora los estudios llevados a cabo usando este tipo de redes se han restringido a un número relativamente pequeño de individuos, normalmente no más de 10000 [6], pero en muchos casos (por ejemplo, una pandemia), el número de sujetos involucrados es del orden de millones. Esto tiene un alto precio en términos de coste computacional ya que, salvo casos concretos como las redes con distribución de probabilidad potencial, su ajuste implica una búsqueda exhaustiva, haciéndolo inviable utilizando medios tradicionales u obligando a importantes restricciones, como tamaños de la red reducidos (en nodos y/o en relaciones), o limitaciones en la exploración de los parámetros de ajuste.

Por ello en este trabajo se presenta un sistema capaz de ajustar estos modelos con un coste computacional mucho menor. Para ello se sustituye la búsqueda exhaustiva tradicional por un algoritmo metaheurístco, MOPSO (Multi-Objective Particle Swarm Optimization), el cual se combina con un sistema de computación paralela, con la finalidad de obtener tiempos de cálculo razonables y abordables. Además, para mantener un bajo acoplamiento entre ambos, los dos sistemas se apoyan en un servicio de almacenamiento de archivos en la nube.

https://doi.org/10.17979/spudc.9788497497169.483

La estructura del artículo es la siguiente. En la sección 2 se presenta el modelo de redes aleatorias del virus de la varicela. En la sección 3 se describe la arquitectura del sistema utilizada para el ajuste del mismo. En la sección 4 se presentan y comentan los resultados. El trabajo termina con las conclusiones y trabajos futuros.

2 MODELO DE RED ALEATORIA DE LA VARICELA

Los modelos de redes se gestionan como un grafo en el que cada nodo es un individuo con una serie de atributos concretos, como por ejemplo: edad, sexo, estado respecto a la enfermedad (susceptible, infectado, recuperado, en latencia, etc.), y con una serie de aristas que los unen a otros nodos y que representan las relaciones (sociales o de otro tipo). Estos elementos definen la estructura de la red y las vías por las que se contagia la enfermedad. Hay distintas maneras de implementar este tipo de redes. La más tradicional es la de Erdős y Rényi [5], aunque han ido surgiendo otras alternativas.

La base de nuestro modelo es una red de nodos interconectados de manera aleatoria. Estos simulan a los individuos en una población concreta, es decir, cada nodo en el modelo tiene una edad, un sexo, etc. La interconexión es de vital importancia en el modelo porque define el contacto y, por tanto, que el virus pueda ser transmitido. Si no existe la conectividad suficiente entre nodos, la enfermedad no se propagará debidamente o si, por el contrario, existe demasiada conectividad, la enfermedad tomará un comportamiento irreal.

El modelo de redes aleatorias que hemos diseñado utiliza el propuesto por Erdős y Rényi, basado en la teoría de grafos y la teoría de la probabilidad. En nuestro caso se ha implementado con 250000 nodos, 1040 turnos de simulación y 20 años de contagios del virus. Cada nodo representa a un individuo en la población (sexo, edad, …) y su estado respecto a la enfermedad (recuperado, latente, infectado, o susceptible).

La varicela no afecta a todos los individuos de la población de igual manera, sino que es muy dependiente del grupo de edad del individuo en cuestión. Por ello en el modelo se diferencian diferentes probabilidades de contagio para cada grupo de edad: 6 a 12 meses, 1 a 4 años, 5 a 9 años, 10 a 14 años, 15 o más años. Se dispone de datos reales proporcionados por la Comunidad Valenciana de los de los diez últimos años [9]. En la Tabla 1 se resumen el porcentaje de infectados según grupo de edad y en la Figura 1 se muestra la cantidad de infectados por cada 100000 habitantes, especificados para cada semana del año (52 semanas).

Con estos datos facilitados por la Comunidad Valenciana se observa que el virus se comporta de manera más o menos contagiosa dependiendo de la semana del año. Por ello en el modelo se define una probabilidad de contagio específica para cada una de las semanas del año.

Tabla 1: Porcentaje de infectados de varicela según grupo de edad.

Grupo de edad	% de infectados
6 a 12 meses	3%
1 a 4 años	48%
5 a 9 años	32%
10 a 14 años	6%
15 o más años	11%

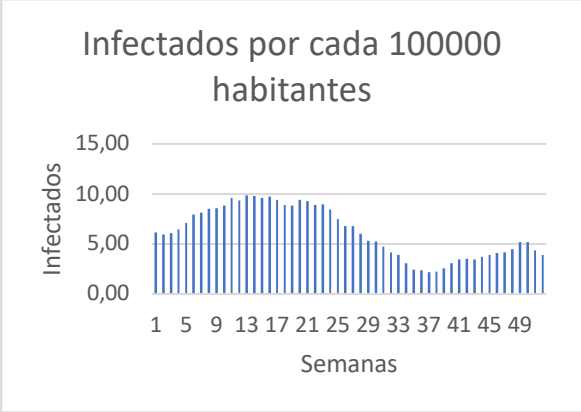

Figura 1: Infectados por varicela

Por lo tanto el modelo de redes aleatorias de la varicela consta de 58 parámetros que hay que ajustar para obtener un comportamiento lo más realista posible. Estos parámetros son:

- Grado de conectividad media de los nodos.
- Probabilidad de contagio para cada uno de los 5 grupos de edad (5).
- Probabilidad de contagio de cada semana del año (52).

2.1 AJUSTE DEL MODELO DE TRANSMISIÓN DEL VIRUS DE LA VARICELA

El objeto de este estudio es ajustar los parámetros de un modelo para simular la transmisión del virus infectocontagioso de la varicela en un tiempo razonable.

Dado el número de parámetros a ajustar, la búsqueda exhaustiva de cada parámetro requeriría un tiempo de cómputo inabordable (en torno a 3 años). Para reducirlo se introduce, como primer paso,

computación distribuida que, aunque disminuye el tiempo sustancialmente, sigue haciendo inviable el cálculo del modelo para una investigación a corto plazo ya que requiere alrededor de 84 días.

Para optimizar la búsqueda de los parámetros anteriormente mencionados se ha optado por un algoritmo metaheurístico de optimización, MOPSO [3], que aunque no asegura encontrar el óptimo en todos los casos, se acerca suficientemente para que sea válido para el modelo.

Los dos objetivos del algoritmo de optimización son el porcentaje de infectados por grupo de edad y los infectados por semana.

Para validar el modelo se compara cada parámetro por separado con su valor real (Tabla 1 y Figura 1). Se utiliza el error cuadrático medio (RMSE) entre el valor calculado por el modelo y el valor real, que debe ser menor que 10 para cada uno de los parámetros.

3 ARQUITECTURA DEL SISTEMA DE AJUSTE DEL MODELO

En la Figura 2 se muestra un esquema de la arquitectura del sistema.

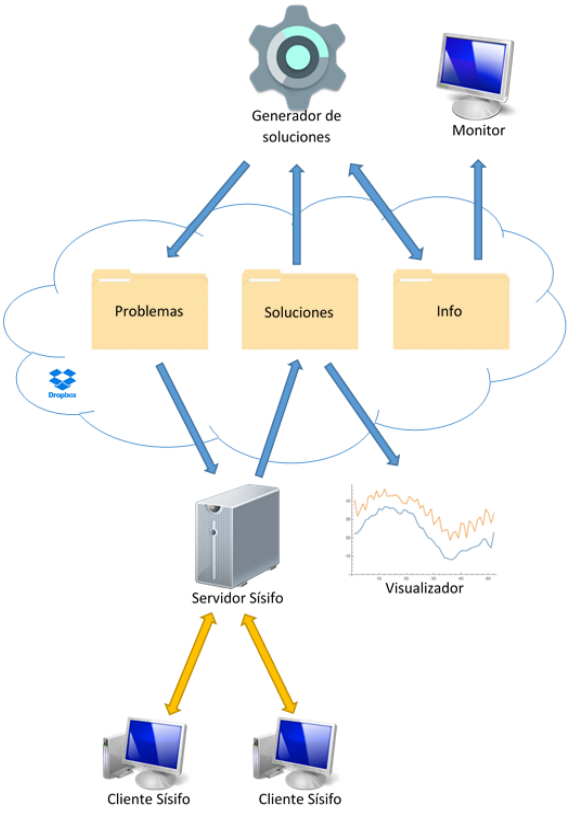

Figura 2: Arquitectura del sistema

Los elementos que componen esta arquitectura son:

- Generador de soluciones: Procedimiento que genera fichas de problemas que contienen los parámetros del modelo de varicela. Para generar y evaluar estas fichas se utiliza el algoritmo MOPSO.
- Sísifo: Es un sistema cliente-servidor encargado de solucionar las fichas de problemas generadas previamente para su posterior análisis en el generador de soluciones.
- Monitor: Es un entorno web para monitorizar el estado del sistema (fichas por resolver, iteraciones restantes, ficheros en el frente de Pareto, ajustes…).
- Dropbox: Contiene un directorio compartido entre el generador Sísifo y el propio monitor. Consta de tres directorios: problemas, soluciones e info.
- Visualizador: Es el encargado de mostrar gráficamente las soluciones para observar el ajuste obtenido por el generador de soluciones. Se ejecuta bajo demanda. Es opcional y muy útil.

3.1 GENERADOR DE SOLUCIONES

El generador de soluciones (GS), como su nombre indica, genera posibles soluciones al modelo de la varicela. El algoritmo que éste utiliza para generar los parámetros y ajustar dicho modelo es MOPSO y está implementado en lenguaje Python.

3.1.1 MOPSO

La optimización por enjambre de partículas, conocida por sus siglas en inglés PSO de *Particle Swarm Optimization,* hace referencia a una metaheurística que evoca el comportamiento de las partículas en la naturaleza (para una descripción formal del algoritmo PSO ver [3]).

En problemas de optimización, en bastantes ocasiones se requiere la optimización simultánea de más de un objetivo (optimización multiobjetivo). Habrá que optimizar por tanto una función de la forma $f: S \rightarrow T$, donde $S \subset \mathbb{R}^k$, {R}^{n} y {R}^{k}. Pero el problema está en que normalmente no existe un elemento de S que produzca un óptimo de forma simultánea para cada uno de los k objetivos que componen f. Esto se deberá a la existencia de conflictos entre objetivos, que harán que la mejora de uno de ellos de lugar a un empeoramiento de algún otro. Habrá que llegar por tanto a una situación de compromiso en la que todos los objetivos sean satisfechos en un grado aceptable, desde el punto de vista de diseño.

El PSO también se ha aplicado a problemas multiobjetivo (MOPSO).
La diferencia entre PSO y MOPSO es como trata el conflicto entre objetivos, ya que la evaluación de la función objetivo tiene en cuenta la "dominancia de Pareto" al mover las partículas, de manera que las soluciones no-dominadas son aproximadas al frente de Pareto.

Un frente de Pareto puede definirse como un conjunto de soluciones tal que, tomada una solución cualquiera del frente de Pareto, sus correspondientes salidas no pueden ser mejoradas todas ellas simultáneamente por ninguna otra solución. Los frentes de Pareto cumplen la condición de que, elegido el vector de salida $\bar{y}_P$ de una solución P perteneciente al frente de Pareto, y dado un vector de salida $\bar{y}_A$ e $\bar{y}_P$ que:

$$y_{A,i} \leq y_{P,i} \forall i \quad (1)$$

La solución P mejora a cualquier otra solución en al menos uno de sus objetivos. Se consideraría que P está en el frente de Pareto sólo si la desigualdad se cumple para al menos una de las componentes del vector $\bar{y}_P$, pues en caso contrario, P no estaría en el frente.

3.1.2 Generador de soluciones

Los parámetros del algoritmo de optimización MOPSO son configurables a través de un fichero de texto. Estos son, entre otros, numero de partículas, velocidad, tasa de mutación, distancia, iteraciones, etc.

Los parámetros se pasan al modelo de varicela usando fichas de problemas que son generadas como ficheros de texto, que contienen todos los parámetros del modelo para simular una red de contagio del virus. El generador se queda a la espera de que se le devuelvan las soluciones de las partículas generadas. Con esta primera iteración ya se tiene un conjunto de partículas en el frente de Pareto. Entonces se eligen, aleatoriamente, un mejor global para cada partícula y se actualizan las partículas añadiendo a la partícula actual la velocidad anterior de un vector que tiene en cuenta la mejor posición que ha tenido la partícula actual y la mejor partícula global que ha sido elegida aleatoriamente. A continuación se ejecutan iteraciones donde en cada una se generan las nuevas fichas, se simulan y se evalúan, actualizando las velocidades y las partículas. Se repite hasta alcanzar la condición de parada (número de iteraciones = 500).

Cada vez que el algoritmo completa una iteración escribe en el fichero monitor.txt los ajustes de cada partícula que se encuentran en el frente de Pareto. De esta forma el usuario puede monitorizar la iteración actual y el mejor ajuste hasta el momento.

3.2 SÍSIFO

Sísifo (http://sisifo.imm.upv.es/) es un sistema cliente-servidor desarrollado por la Universidad Politécnica de Valencia que permite que un problema sea resuelto utilizando computación distribuida según la filosofía Desktop Grid, permitiendo el uso de un conjunto de ordenadores heterogéneos.

Éste asigna tareas a un conjunto de PCs, espera a que las tareas terminen, y recoge los resultados para su posterior análisis. En nuestro caso, cada cliente Sísifo tiene corriendo el modelo de varicela de modo que cuando el servidor encuentra problemas sin resolver, se lo asigna a uno de sus clientes libres. Cuando el fichero está resuelto, le devuelve la solución al servidor y éste se encarga de almacenar dicha solución debidamente. En cuanto a características técnicas de Sísifo, está compuesto por 15 equipos:

- 10 equipos XEON X3230 a 2.66 GHz con 6.6 GB de RAM
- 5 equipos XEON X3430 a 2.4 GHz con 16 GB de RAM.

3.3 DROPBOX

Drobpbox (www.dropbox.com) es un servicio de alojamiento de archivos multiplataforma en la nube, operado por la compañía del mismo nombre. Este servicio permite a sus usuarios almacenar y sincronizar archivos en línea y compartir estos archivos y carpetas con otros usuarios. Existe un bajo acoplamiento entre los sistemas que lo utilizan. Los sistemas sólo conocen un directorio con el que trabajan y además los permite trabajar en redes diferentes, lo ofrece una gran ventaja: se pueden reemplazar sin que afecte al resto del sistema.

En esta aplicación se ha usado Dropbox para compartir un directorio con las siguientes carpetas:

- Problemas: Se almacenan los problemas generados por el GS pendientes de resolver y Sísifo las recoge.
- Soluciones: Se almacenan las soluciones creadas por Sísifo y el GS lo toma para analizarlo.
- Info: Se almacena información del ajuste actual.

3.4 MONITOR DEL SISTEMA

El monitor del sistema es el encargado de monitorizar en tiempo real el estado de los cálculos y, por consiguiente, del ajuste de la red. Es decir, lee del directorio info los ficheros restantes que quedan

por resolver en la iteración actual, número de iteraciones, parámetros de configuración del algoritmo MOPSO, ficheros en el frente de Pareto y los errores correspondientes, y parámetros de cada uno de los ficheros.

La arquitectura del monitor lo componen los elementos mostrados en la Figura 3.

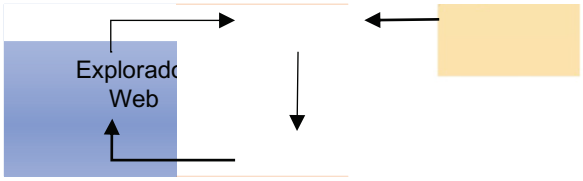

Figura 3: Arquitectura del sistema

Para acceder al monitor basta con abrir un explorador cualquiera de internet y acceder a la url: varicela.ddns.net; si el servidor Apache Tomcat está corriendo se podrá visualizar el monitor tal como muestra la Figura 3. Es importante mencionar que el servidor puede alojarse en cualquier máquina que tenga capacidad de ver la carpeta compartida "Info", ya que sólo necesita ese directorio para funcionar correctamente.

3.5 VISUALIZADOR DE SOLUCIONES

El visualizador de soluciones es una aplicación implementada usando el lenguaje Mathematica, cuya finalidad es permitir visualizar gráficamente el ajuste de las soluciones que se consideren mejores y así poder comprender y analizar mejor el ajuste del modelo.

Las partes de la interface de este visualizador, que se muestran en la Figura 4, son:

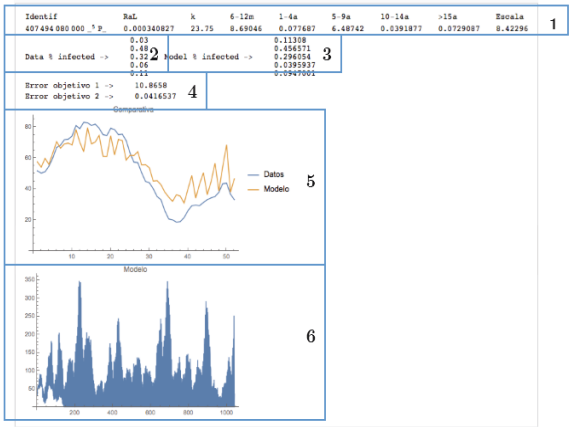

Figura 4: Interface del visualizador

1. Parámetros del modelo
2. Porcentaje de infectados por grupo de edad real
3. Porcentaje de infectados por grupo de edad modelo
4. Errores en los objetivos:
   a. Error objetivo 1: RMSE entre las gráficas Datos (en azul) y Modelo (amarilla) del bloque 5
   b. Error objetivo 2: RMSE entre los datos reales (bloque 2) y datos calculados por el modelo (bloque 3)
5. Datos reales y los datos calculados por el modelo
6. Gráfica que asegura que la enfermedad no ha desaparecido

3.6 FUNCIONAMIENTO DEL SISTEMA

Por último, se resumen los pasos del sistema para obtener las soluciones (Figura 5).

1. El GS carga la configuración que usa MOPSO
2. El GS genera las N fichas de problemas y las almacena en el directorio de problemas
3. El servidor Sísifo detecta nuevos problemas y los reparte a sus clientes disponibles
4. Cada cliente resuelve el problema asignado y lo devuelve al servidor Sísifo
5. El servidor Sísifo almacena las soluciones en el directorio de soluciones
6. El GS analiza cada fichero
7. El GS crea más fichas de problemas volviendo al punto 1 o termina su ejecución, dependiendo del número de iteraciones programadas.

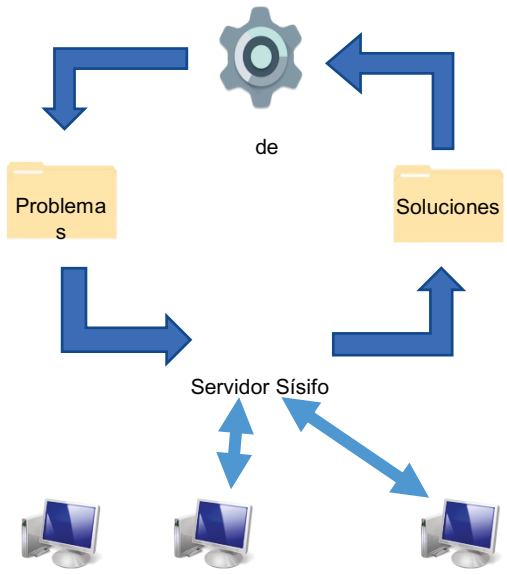

Figura 5: Esquema del funcionamiento del sistema

https://doi.org/10.17979/spudc.9788497497169.483

4 RESULTADOS DE LA SIMULACIÓN DEL MODELO

La configuración del GS para los resultados obtenidos es la siguiente:

Tabla 1: Configuración del GS

Parámetros	Valores
ITMAX (no. Iteraciones)	500
Partículas	100
Nrepositorio	200
Tasa_mutación	0.5
Dist_mutación	20
Aleat_elec_global	0.1

Tras varias pruebas se ha demostrado que con un número máximo de iteraciones de 500 se obtienen soluciones con un RMSE menor que diez en cada uno de los objetivos; con 100 partículas se obtiene la suficiente variedad en los parámetros de los problemas generados, y se evita una convergencia prematura del algoritmo. Para el resto de parámetros se han elegido los valores típicos para cada uno de ellos y han funcionado correctamente.

Respecto al número de nodos, se han realizado pruebas con valores entre 1000 y 300000. Un mayor número de nodos en la red implica una menor probabilidad de que la enfermedad desaparezca. Esto asegura siempre soluciones válidas para seguir generando fichas, pero el tiempo de cálculo al simular dicha red es también mayor. En las pruebas se ha encontrado en 250000 el ajuste de las soluciones es igual de bueno que con 300000, pero se reduce el tiempo de cálculo de manera significativa, por lo que se ha fijado a ese valor.

El número de turnos equivale a semanas, por lo que 1040 supone simular contagios en la red durante 20 años. Los datos que se ajustan son sólo los 10 últimos años. Los 10 primeros son necesarios para que el modelo de varicela se estabilice y se puedan ajustar debidamente.

Con la configuración descrita se ha obtenido el siguiente ajuste (Tabla 3).

Tabla 3. RMSE en objetivos

Objetivos	RMSE
Infectados por semana	8.38863
Infectados por grupo de edad	0.0422087

Se considera que es un buen ajuste de los parámetros de la red pues el RMSE de ambos objetivos está por debajo de 10. En concreto el error para la simulación de los infectados según el grupo de edad es muy pequeño, del orden de centésimas.

En la Figura 6 se muestra en el visualizador de soluciones la mejor de las obtenidas.

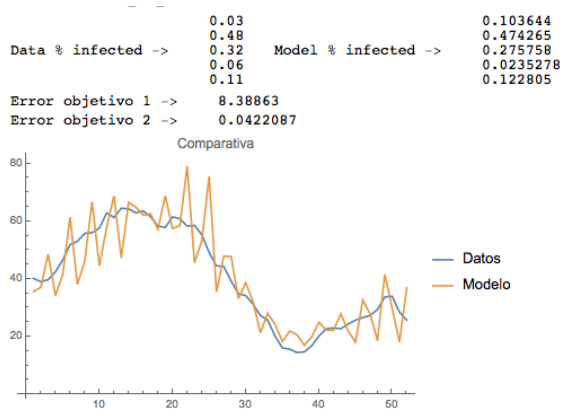

Figura 6: Mejor solución obtenida

Se ha comparado el tiempo de cálculo usando MOPSO y computación distribuida frente a una estimación realizada con otros métodos más tradicionales, como la búsqueda exhaustiva de los parámetros.

El modelo de varicela tarda en resolver un problema 15 minutos de media; calculando que se necesiten 120000 problemas para obtener al menos una solución igual de buena que la obtenida por MOPSO, esto hace un total de 1250 días para un sistema de búsqueda exhaustiva. Si a esa búsqueda exhaustiva le añadimos computación distribuida, en nuestro caso Sísifo con 15 clientes disponibles, se reduce a 83.3 días.

Usando MOPSO y computación distribuida, la condición de parada era 500 iteraciones pero en la iteración 173 se obtuvo la mejor solución. En cada iteración se resuelven 100 problemas, por lo que para resolver 17300 problemas se requieren un total de 12 días.

Es decir, introducir un entorno de cálculo distribuido en un método clásico de búsqueda mejora enormemente el tiempo de cálculo pero, en casos como éste, sigue sin ser abordable a corto plazo. Por ello combinar un algoritmo como MOPSO con computación distribuida permite obtener una calidad en la solución igual o superior a la obtenida mediante metodologías clásicas, con un tiempo de cálculo mucho menor.

Por último se ha comprobado, tras variar pruebas, que se puede reducir el número de iteraciones ya que se obtiene al menos una solución equivalente a la presentada entre las iteraciones 173 y 231.

5 CONCLUSIONES Y TRABAJOS FUTUROS

En este trabajo se ha abordado el ajuste de los parámetros de un modelo de redes aleatorias para estudiar el contagio de una enfermedad, muy común en ciertos grupos de edad, como es la varicela.

Ante el elevado coste computacional con métodos clásicos como búsqueda exhaustiva, se ha usado computación distribuida como una primera mejora. Aun así, los tiempos siguen siendo muy elevados. Esto ha hecho plantear como alternativa el usar un algoritmo metaheurístico, en concreto MOPSO. Se ha conseguido un buen ajuste del modelo de varicela en 12 días de tiempo de CPU, el cuál puede verse reducido a unas pocas horas si se dispone de mayor potencia de cálculo.

Para abordar este tipo de problemas tan costosos computacionalmente se ha creado una arquitectura que integra el generador de soluciones basado en MOPSO con Sísifo, un clúster de cálculo distribuido. Esta arquitectura ha permitido introducir almacenamiento en la nube, para que diferentes sistemas colaboren entre sí para un mismo fin. Esto proporciona una alta cohesión y a su vez un muy bajo acoplamiento, pues los elementos pueden sustituirse sin que repercuta negativamente en la propia arquitectura.

Como trabajos futuros se plantea, con respecto al modelo de varicela, realizar diversas simulaciones para determinar las posible estrategias de vacunación y el coste que supone vacunar a la población existente.

Con respecto a la arquitectura del sistema, se podría utilizar para ajustar otros modelos de redes existentes. Esto permitiría, por un lado, validarla y, por otro, hacerla completamente general, lo que abriría un amplio abanico de posibles aplicaciones.

English summary

ADJUSTING A CHICKENPOX INFECTION MODEL BY HEURISTIC OPTIMIZATION ALGORITHMS IN A DISTRIBUTED COMPUTING ENVIRONMENT

Abstract

The searching for the parameters that best adjust a model of random networks is computationally very expensive and in many cases unapproachable through the exhaustive evaluation of all the solutions. In this work, the behavior of the chicken pox (varicella) virus will be modeled by random networks. For the adjustment of this model a strategy that combines metaheuristic algorithms, distributed computing and cloud storage is presented. This architecture allows the synergy of totally decoupled elements (computer system, network model and solution generator) for different platforms without having to modify them. The resulting model fits fairly well with the real data available and will allow actions to be taken in order to launch a new vaccination campaign.

Keywords: Modelling, Random networks, Heuristic optimization, MOPSO, Chickenpox (Varicella).

Referencias

[1] Acedo, L., Moraño, J. A., Santonja, F. J., Villanueva, R. J. (2016) "A deterministic model for highly contagious diseases: The case of varicella", *Physica A: Statistical Mechanics and its Applications*, 450, 278-286.

[2] Ahmed, E., Agiza, H. N. (1998) "On modeling epidemics including latency, incubation and variable susceptibility", *Physica A: Statistical Mechanics and its Applications*, 253(1), 347-352.

[3] Alonso-Zotes, F., Santos, M., (2017) Heuristic optimization of interplanetary trajectories in aerospace missions. *Revista Iberoamericana de Automática e Informática Industrial* 14(1), 1-15.

[4] Bisset, K. R., Aji, A. M., Bohm, E., Kale, L. V., Kamal, T., Marathe, M. V., Yeom, J. S. (2012, May) "Simulating the spread of infectious disease over large realistic social networks using charm++". In: Parallel and Distributed Processing Symp (IPDPSW), 2012 IEEE 26th Int (pp. 507-518). IEEE.

[5] Bollobás, B. (1998) Random graphs. In: Modern Graph Theory (pp. 215-252). Springer New York.

[6] Christakis, N. A., Fowler, J. H. (2008) "The collective dynamics of smoking in a large social network", *New England Journal of Medicine*, 358(21), 2249-2258.

[7] Hethcote, H. W. (2000) "The mathematics of infectious diseases", *SIAM Review*, 42(4), 599-653.

[8] Lenne, X., Domingo, J. D., Gil, A., Ridao, M., Lluch, J. A., Dervaux, B. (2006) "Economic evaluation of varicella vaccination in Spain—results from a dynamic model", *Vaccine*, 24(47), 6980-6989.

[9] Varicella report bulletin of the Community of Valencia (2014). http://www.sp.san.gva.es/DgspPortal/docs/Inf Varicela 2014.pdf

[10] Witten, G., Poulter, G. (2007) "Simulations of infectious diseases on networks", *Computers in Biology and Medicine*, 37(2), 195-205.