Máster en Ingeniería de Sistemas y de Control





Ajuste de parámetros de un modelo de redes de contagio de varicela mediante algoritmos de optimización heurísticos en un entorno de cálculo distribuido

Autor:

José Gabriel García Caro

Directores:

Matilde Santos Peñas

Francisco Javier Villanueva Oller

Curso académico: 2015 - 2016

Septiembre 2016



Máster en Ingeniería de sistemas y de control

Ajuste de parámetros de un modelo de redes de contagio de varicela mediante algoritmos de optimización heurísticos en un entorno de cálculo distribuido

Tipo de proyecto: Específico propuesto por el alumno

Autor: José Gabriel García Caro

Directores: Matilde Santos Peñas

Francisco Javier Villanueva Oller

Convocatoria: Septiembre 2016



Autorización

Autorizamos a la Universidad Complutense y a la UNED a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a sus autores, tanto la memoria de este Trabajo Fin de Máster, como el código, la documentación y/o el prototipo desarrollado.

Firmado: José Gabriel García Caro



Firma del alumno



Resumen

Normalmente, al tratar con modelos de redes aleatorios nos encontramos con que la búsqueda de los parámetros que mejor hacen evolucionar al modelo es una tarea computacionalmente muy costosa. Esto es así porque la manera habitual de afrontarlos es mediante la evaluación exhaustiva de todas las soluciones, una tarea que en muchos de los casos es prácticamente inabordable por cualquier sistema, dado que el tiempo de cálculo invertido en esta tarea suele ser desproporcionado.

En este trabajo se va a modelar mediante redes aleatorias el comportamiento del virus de la varicela. Dado que el ajuste de este modelo es computacionalmente muy costoso, esto ha dado pie a la búsqueda de una optimización de este proceso que permita disminuir el coste de recursos y tiempo necesarios. Por ello se presenta una alternativa que combina algoritmos metaheurísticos, computación distribuida y almacenamiento en la nube. Esta arquitectura nos ha permitido hacer trabajar juntos a elementos creados de manera independiente, es decir, 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 una nueva campaña de vacunación. Esto contribuirá tanto a la mejora de la salud de la población en general, como a las empresas encargadas de distribuir dichas vacunas ya que podrán hacerlo de una manera más eficiente y económica.

Palabras clave

Varicela, modelado, redes aleatorias, cálculo distribuido, algoritmos de optimización, heurística, almacenamiento en la nube.

"Mi vida está guiada por un elemento fundamental: la constancia. Ser constante implica sacrificios, sin sacrificios no hay resultados".

Carlos Coloma

ÍNDICE

Capitulo 1: INTRODUCCIÓN 13
1.1 Motivación 14
1.2 Objetivos 16
1.2.1 Objetivo general 16
1.2.2 Objetivos específicos 16
1.3 Estructura de la memoria 17
Capitulo 2: ESTADO DEL ARTE 19
2.1 Modelos de redes aleatorias 19
2.2 Optimización por enjambre de partículas 22
2.2.1 Descripción del algoritmo PSO 22
2.2.2 Tipos de algoritmos PSO 25
2.2.3 Topologías del enjambre 27
2.2.4 MOPSO 29
Capitulo 3: DISEÑO DEL MODELO 33
3.1 Ajuste del modelo 37
Capitulo 4: ESTRUCTURA DEL SISTEMA DE AJUSTE DEL MODELO 39
4.1 Arquitectura del sistema 39
4.2 Generador de soluciones 41
4.2.1 Configuración 42
4.2.2 Fichas de problemas 43
4.2.3 Ficheros solución 46
4.3 Sísifo 47
4.4 Dropbox 49
4.5 Monitor del sistema 50

4.6 Visualizador de soluciones 53
4.7 Funcionamiento del sistema.....55
Capitulo 5: RESULTADOS.....57
Capitulo 6: CONCLUSIONES Y TRABAJOS FUTUROS 63
6.1 Trabajos futuros 64
BIBLIOGRAFÍA.....65
ABREVIATURAS 69

LISTA DE FIGURAS

Figura 1. Virus varicela-zóster 13
Figura 2. Niño con varicela 14
Figura 3. Red generada mediante ER 20
Figura 4. Distribución de Poisson de conectividad de una red aleatoria 21
Figura 5. Movimiento de una partícula en el espacio de soluciones 25
Figura 6. Topologías del enjambre 28
Figura 7. Ejemplos de sociometrías en el enjambre 29
Figura 8. Topología de Von Neumann o cuadrada 29
Figura 9. Ejemplo de frente de Pareto para dos objetivos 31
Figura 10. Ejemplo de red vírica 33
Figura 11. Infectados por varicela 36
Figura 12. Arquitectura del sistema 40
Figura 13. Ejemplo de configuración 43
Figura 14. Ficha de problemas 45
Figura 15. Muestra de un fichero solución 46
Figura 16. Arquitectura de Sísifo 48
Figura 17. Arquitectura del monitor 50
Figura 18. Monitor del sistema 52
Figura 19. Visualización de una solución 54
Figura 20. Flujo de funcionamiento del sistema 56
Figura 21. Mejor solución obtenida 59
Figura 22. Problemas resueltos 61

LISTA DE TABLAS

Tabla 1. Porcentaje de infectados según grupo de edad 35
Tabla 2. Configuración del GS 57
Tabla 3. Configuración base de la red del modelo 58
Tabla 4. RMSE en objetivos 58
Tabla 5. Tiempos de cálculo 60



Capitulo 1: INTRODUCCIÓN

La varicela es una enfermedad altamente contagiosa causada por el virus de la varicela-zóster (VZV) [23], de la familia de los herpesvirus (Figura 1). Este virus es el responsable de la varicela y del herpes zóster y tiene una altísima prevalencia en todos los países del mundo. La varicela afecta principalmente a los niños. En la mayoría de casos es una infección benigna cuyos síntomas no exceden de una semana [25]. Ocasionalmente puede complicarse y tiene una tasa muy baja de mortalidad. Sin embargo las infecciones en adultos pueden resultar muy severas.

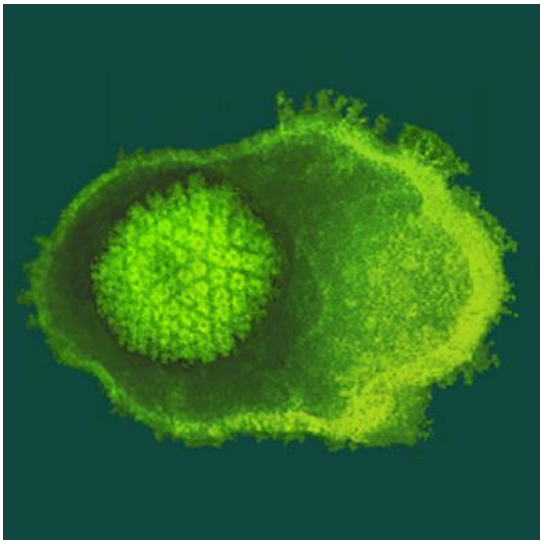


Figura 1. Virus varicela-zóster

Esta enfermedad se inicia con un periodo prodrómico semejante a un cuadro gripal con fiebre leve o moderada; luego aparece un exantema maculopapular, con evolución a vesículas y costras (Figura2). Se acompaña de adenopatía cervical y síntomas generales. Las lesiones dérmicas pueden llegar a producir cicatrices permanentes.



Figura 2. Niño con varicela

Como se ha demostrado en [5], la tasa de transmisión de la varicela es variable según la época del año. Debido a esto, es muy interesante poder predecir el comportamiento de esta enfermedad para poder así tomar las medidas necesarias, por ejemplo, un cambio en la campaña de vacunación.

Todo lo anteriormente expuesto nos hace preguntarnos si somos capaces de ajustar el modelo de varicela, ya que hay empresas farmacéuticas interesadas en este conocimiento, cuyo objetivo es dar una respuesta objetiva a preguntas como qué calendario es mejor para la vacunación, a cuánta gente hay que vacunar, con qué coste, etc.

1.1 Motivación

Los modelos de redes se han convertido en un paradigma de vital importancia en el análisis de sistemas complejos. Estos sistemas abarcan desde la biología evolutiva [26] a las redes neuronales [9], pasando por redes sociales [30], de transporte, o económicas [22]. Por ejemplo, estos modelos pueden usarse para estudiar enfermedades infectocontagiosas [6, 13, 17]. En los últimos tiempos las redes aleatorias se han popularizado como medio para simular los patrones de difusión de enfermedades en redes de gran

tamaño [33, 1, 10]. Además, este tipo de redes proporcionan una alternativa a los esquemas tradicionales basados en ecuaciones diferenciales, cuyo origen está en los trabajos de Kermack y Mckendrick [21, 15].

Es cierto que las ecuaciones diferenciales son un mecanismo conocido y potente [24, 18] que permite estudiar la dinámica de muchos sistemas, pero cuyo principal inconveniente es que, cuando se utilizan en el entorno de un modelo epidemiológico, tiene muchas limitaciones pues no pueden distinguir entre individuos concretos [7], por lo que introducir en el modelo elementos tales como edad, sexo, enfermedades previas, etc., se convierte en algo muy complejo.

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), que definen la estructura de la red misma 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 [11], pero han ido surgiendo otras alternativas como las scale-free networks [8] o las de Watts y Strogatz [32].

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 [12], 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 (que permiten un ajuste similar al de las ecuaciones diferenciales), su ajuste implica una búsqueda exhaustiva, haciéndolo inviable utilizando medios tradicionales u obligando a grandes limitaciones, como tamaños de la red reducidos (en nodos y/o en relaciones), o restricciones en la exploración de los parámetros de ajuste.

De los modelos de redes podemos encontrar trabajos similares al nuestro que tratan de simular el comportamiento de enfermedades

infectocontagiosas con el fin de estudiar el comportamiento de la enfermedad en cuestión [16]. Modelos de redes epidemiológicos ya se han analizado y resuelto con buenos resultados [2, 28, 3], incluso usando computación distribuida como en el caso del Virus Respiratorio Sincitial (VRS) [2], los impulsos cerebrales [4] o el meningococo C [27]. Sin embargo en algunos de ellos los tiempos de ejecución siguen siendo extremadamente altos. Por ejemplo, en el VRS el tiempo de ejecución usando como método de ajuste de sus parámetros solamente búsqueda exhaustiva se sobrepasan los 20 años de tiempo de cálculo, algo totalmente inviable [2].

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, el cual se combina con un sistema de computación paralela, con la finalidad de obtener tiempos razonables de cálculo 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 como Dropbox.

1.2 Objetivos

1.2.1 Objetivo general

Desarrollo de un modelo computacional distribuido del virus de varicela que disminuya sustancialmente los tiempos de ajuste de los parámetros definidos en el modelo de redes. Se simula el comportamiento del propio virus usando computación distribuida, computación en la nube y usando un algoritmo metaheurístico como MOPSO.

1.2.2 Objetivos específicos

- Implementación de un modelo de redes aleatorio que simula el contagio del virus de la varicela
- Diseño de una arquitectura modular que combina sistemas independientes para crear un sistema de ajuste aplicable a modelos de redes

- Implementación de un generador de soluciones basado en MOPSO para el modelo de redes de varicela que sea capaz de trabajar en un sistema de computación distribuida. Implementado en lenguaje Python;
- Implementación de un monitor de soluciones, cuya finalidad es poder monitorizar el ajuste de los parámetros. Implementado utilizando Servlets, Java y jsp;
- Implementación de un visualizador de resultados, el cual nos mostrará el ajuste obtenido por el sistema. Implementado en lenguaje Mathematica (Wolfram Mathematica).

1.3 Estructura de la memoria

La presente memoria se estructura en los siguientes capítulos:

En el Capítulo 1 se ha introducido la enfermedad objeto de estudio, presentando la motivación que ha llevado a su análisis y modelización.

En el Capítulo 2 se presenta el estado del arte, dónde se describen los modelos de redes aleatorias y la metaheurística utilizada.

En el Capítulo 3 se describe el problema que se aborda en la memoria incluyendo el ajuste del modelo que estudiamos.

En el Capítulo 4 se detalla todo el proceso de ajuste del modelo estudiado además de la arquitectura del sistema y cómo funciona cada uno de sus elementos.

En el Capítulo 5 se presentan los resultados obtenidos en base a todo lo explicado en capítulos anteriores.

En el Capítulo 6 se presentan las conclusiones y los trabajos futuros.

Finalmente se encuentran todas las referencias bibliográficas consultadas y citadas en este trabajo y una lista de siglas y acrónimos.



Capitulo 2: ESTADO DEL ARTE

En este capítulo se presentarán los modelos de redes aleatorias, ya que para reproducir el virus varicela se ha utilizado un modelo de este tipo. Además explicaremos qué es MOPSO, pues ha sido el método elegido de optimización de búsqueda de los parámetros que conforman la red del modelo antes mencionado.

2.1 Modelos de redes aleatorias

Se denomina red aleatoria a un grafo que es generado por algún tipo de proceso aleatorio. La teoría de los grafos aleatorios cae en la intersección entre la teoría de grafos y la teoría de probabilidades, y se fundamenta en el estudio de ciertas propiedades de los grafos aleatorios.

Su origen se remonta a la década de los 50 y culmina en 1960 con el trabajo pionero *On the evolution of random graphs* de Erdős y Rényi. Más actual es el libro Bollobás [19], que es el referente principal en este tipo de grafos. Erdős y Rényi (ER) propusieron un modelo de grafos aleatorios. En este modelo cada par de n vértices de un grafo tiene una probabilidad p de estar conectado por una arista, siendo esta probabilidad independendiente de la existencia o no de otras aristas con otros nodos.

El concepto del modelo ER es sencillo. Consideramos N nodos de una red donde estos están desconectados y distribuidos de forma aleatoria. En el instante inicial escogemos aleatoriamente una pareja de nodos y los unimos. Después de haber unido ambos nodos estos quedan descartados y cogemos otra pareja de nodos al azar y los unimos. Pueden ser escogidos nodos que ya estén enlazados con otros nodos pero, si casualmente elegimos dos que ya han sido conectados entre sí, descartamos ambos y seleccionamos nuevamente otro par. Si repetimos el proceso M veces eligiendo un par de nodos en cada turno al final habremos establecido como máximo M enlaces entre parejas de nodos. Si M es un valor pequeño con respecto al valor total

de nodos muchos de los nodos estarán desconectados, mientras que por el contrario otros nodos estarán formando pequeñas islas (Figura 3).

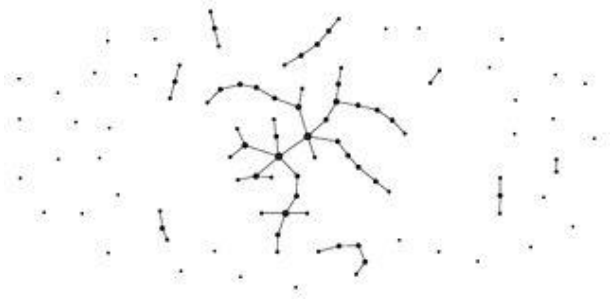


Figura 3. Red generada mediante ER

Por el contrario, si M es grande en comparación con N (el número total de nodos), es muy posible que casi todos los nodos estén enlazados entre sí.

Después de haber conectado M parejas en un conjunto total de N nodos, ¿cuál es la distribución de conexiones $P(k)$ en la red resultante? La red que resulta de este proceso tiene una distribución $P(k)$ de Poisson.

Calculemos ahora la probabilidad $P(k)$ para una red. Para comenzar notemos que el número total N_p de parejas que se pueden formar en un conjunto de N nodos es:

$$N_p = \frac{1}{2}N(N-1) \qquad (1)$$

Como enlazamos M parejas de nodos, la probabilidad p_e de que una pareja arbitraria seleccionada al azar esté enlazada es:

$$p_e = \frac{M}{N_p} = \frac{2M}{N(N-1)} \qquad (2)$$

Ahora enfoquemos nuestra atención sobre un nodo particular v_j de la red, escogido al azar. El número total de parejas que podrían contener a v_j es $N-1$, ya que v_j se podría haber enlazado con los $N-1$ nodos restantes de la red. Sin embargo, en los M enlaces que se llevaron a cabo, no necesariamente escogimos al nodo v_j todas las veces posibles que se podría haber escogido.

Supongamos entonces que de las M parejas que se escogieron, el nodo v_j estaba solamente en k de ellas. La probabilidad de que v_j esté contenido en k parejas de las $N-1$ posibles es:

$$P(k) = \binom{N-1}{k} (p_e)^k (1-p_e)^{N-1} \qquad (3)$$

Esta es una distribución binomial para N y M finitas. Pero si consideramos ahora que la red es muy grande y tomamos el límite $N \to \infty$ y $M \to \infty$ de tal forma que la cantidad permanezca finita, entonces la distribución se transforma en:

$$P(k) = e^{-z} \frac{2^k}{k!} \qquad (4)$$

Por lo que resulta ser la distribución de Poisson con promedio k (Figura 4).

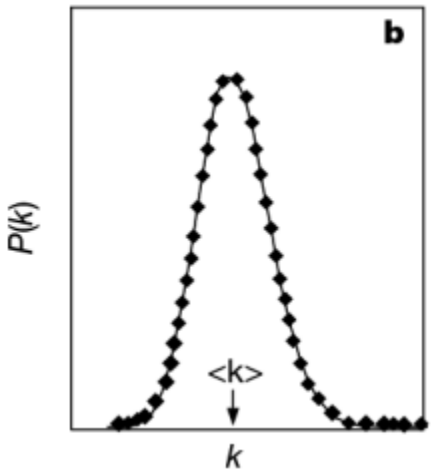


Figura 4. Distribución de Poisson de conectividad de una red aleatoria

2.2 Optimización por enjambre de partículas

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. PSO es una metaheurística, ya que asume pocas o ninguna hipótesis sobre el problema a optimizar y puede aplicarse en grandes espacios de soluciones candidatas. Sin embargo, como toda metaheurística, PSO no garantiza la obtención de una solución óptima en todos los casos.

Originalmente los métodos PSO se atribuyen a los investigadores Kennedy, Eberhart y Shi [19, 29]. Estos métodos en un principio fueron concebidos para elaborar modelos de conductas sociales [20] como el movimiento descrito por los organismos vivos en una bandada de aves o un banco de peces. Posteriormente el algoritmo se simplificó y se comprobó que era adecuado para problemas de optimización.

PSO permite optimizar un problema a partir de una población de soluciones candidatas, denotadas como "partículas", moviendo éstas por todo el espacio de búsqueda según reglas matemáticas que tienen en cuenta la posición y la velocidad de las partículas. El movimiento de cada partícula se ve influido por su mejor posición local hallada hasta el momento, así como por las mejores posiciones globales encontradas por otras partículas a medida que recorren el espacio de búsqueda. El fundamento teórico de esto es hacer que la nube de partículas converja rápidamente hacia las mejores soluciones.

2.2.1 Descripción del algoritmo PSO

Un algoritmo PSO consiste en un proceso iterativo y estocástico que opera sobre un enjambre de partículas como el anteriormente descrito. La posición de cada partícula representa una solución potencial al problema que se está resolviendo. Generalmente, una partícula p_i está compuesta de tres vectores y dos valores de adecuación (*fitness*):

- El vector $x_i = (x_{i1}, x_{i2}, \dots, x_{in})$ almacena la posición actual de la partícula en el espacio de búsqueda;
- El vector $pBest_i = (p_{i1}, p_{i2}, \dots, p_{in})$ almacena la posición de la mejor solución encontrada por la partícula hasta el momento;
- El vector $v_i = (v_{i1}, v_{i2}, \dots, v_{in})$ almacena la dirección (gradiente) según la cual se moverá cada partícula;
- El valor de $fitness_x_i$ almacena el valor del ajuste de la solución actual (vector x_i);
- El valor de $fitness_pBest_i$ almacena el valor del ajuste de la mejor solución local encontrada hasta el momento (vector $pBest_i$)

Generalmente el enjambre se inicializa generando aleatoriamente las posiciones y las velocidades iniciales de las partículas. Una vez generadas las posiciones, se calcula el valor de cada una y se actualizan los valores de $fitness_x_i$ y $fitness_pBest_i$.

Las velocidades se generan aleatoriamente, con cada componente en el intervalo $[-v_{max}, v_{max}]$, donde v_{max} será la velocidad máxima que pueda tomar una partícula en cada movimiento.

Una vez se inicializa el enjambre, las partículas se mueven dentro del proceso iterativo. Una partícula se mueve desde una posición del espacio de búsqueda hasta otra, simplemente añadiendo al vector posición x_i el vector velocidad v_i para obtener un nuevo vector posición:

$$x_i \leftarrow x_i + v_i \qquad (5)$$

Una vez calculada la nueva posición de la partícula, se evalúa actualizando $fitness_x_i$. Además, si el nuevo *fitness* es el mejor encontrado hasta el momento, se actualizan los valores de mejor posición $pBest_i$ y $fitness_pBest_i$. El vector velocidad de cada partícula es modificado en cada iteración utilizando la velocidad anterior, un componente cognitivo y un componente social. El núcleo del algoritmo PSO viene representado por las siguientes ecuaciones:

$$v_i^{k+1} = \omega \cdot v_i^k + \varphi_1 \cdot rand_1 \cdot \left(pBest_i - x_i^k\right) + \varphi_2 \cdot rand_2 \cdot (g_i - x_i^k) \qquad (6)$$

Donde:

- v_i^k: velocidad de la partícula i en la iteración k;
- ω: factor inercia que suele mantenerse entre 0.9 y 1.2 [14]. Valores altos provocan una búsqueda exhaustiva (más diversificación) y valores bajos una búsqueda más localizada (más intensificación);
- φ_1, φ_2: son velocidades de aprendizaje, también conocidos como pesos que controlan los componentes denominados cognitivo y social;
- $rand_1, rand_2$: son números aleatorios entre 0 y 1;
- x_i^k: posición actual de la partícula i en la iteración k;
- $pBest_i$: mejor solución (posición) encontrada por la partícula i en la iteración k;
- g_i: representa la posición de la partícula con el mejor $fitness_pBest$ del entorno de p_i (*lBest o localbest*) o de todo el enjambre (*gBest o globalbest*).

La ecuación 6 refleja la actualización del vector velocidad de cada partícula i en cada iteración k. El componente cognitivo está modelado por el factor $\varphi_1 \cdot rand_1 \cdot (pBest_i - x_i^k)$ y representa la distancia entre la posición actual y la mejor conocida por esa partícula, es decir, la decisión que tomará la partícula influenciada por su propia experiencia a lo largo de su vida. El componente social está modelado por $\varphi_2 \cdot rand_2 \cdot (g_i - x_i^k)$ y representa la distancia entre la posición actual y la mejor posición del vecindario, es decir, la decisión que tomará la partícula según la influencia que el resto del enjambre ejerce sobre ella.

$$x_i^{k+1} = x_i^{k+1} + v_i^{k+1} \qquad (7)$$

La ecuación 7 modela el movimiento de cada partícula *i* en cada iteración *k*.

La Figura 5 muestra una representación del movimiento de una partícula en el espacio de soluciones.

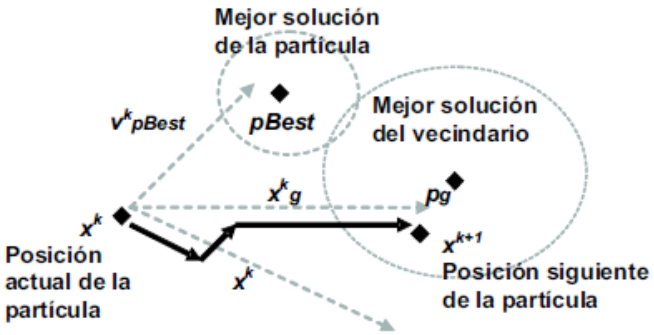


Figura 5. Movimiento de una partícula en el espacio de soluciones

En la Figura 5, las flechas de línea discontinua representan la dirección de los vectores de velocidad actual:

- v^k_{pBest}: es la velocidad de la mejor posición tomada por la partícula;
- v^k_g: es la velocidad de la mejor partícula encontrada en el vecindario;
- v^k: es la velocidad actual de la partícula.

La flecha de línea continua representa la dirección que toma la partícula para moverse desde la posición x^k hasta la posición x^{k+1}. El cambio de dirección de esta flecha depende de la influencia de las demás direcciones que intervienen en el movimiento.

2.2.2 Tipos de algoritmos PSO

Existen diferentes tipos de algoritmos PSO atendiendo a diversos factores de configuración, por ejemplo, según la importancia de los pesos cognitivo y social y según el tipo de vecindario utilizado.

Por una parte, dependiendo de la influencia de los factores cognitivo y social (valores $\varphi_1 \, y \, \varphi_2$, respectivamente) sobre la dirección de la velocidad que toma una partícula en el movimiento (ecuación 7), Kennedy [29] identifica cuatro tipos de algoritmos:

- Modelo Completo: $\varphi_1, \varphi_2 > 0$. Tanto el componente cognitivo como el social intervienen en el movimiento;
- Modelo Cognitivo: $\varphi_1 > 0$ y $\varphi_2 = 0$. Únicamente el componente cognitivo interviene en el movimiento;

- Modelo Social: $\varphi_1 = 0$ y $\varphi_2 > 0$. Únicamente el componente social interviene en el movimiento;
- Modelo Social exclusivo: $\varphi_1 = 0$, $\varphi_2 > 0$ y $g_i \neq x_i$. La posición de la partícula en sí no puede ser la mejor de su entorno.

Por otra parte, desde el punto de vista del vecindario, es decir, según la cantidad y posición de las partículas que intervienen en el cálculo de la distancia en la componente social, se clasifican dos tipos de algoritmos: PSO Local y PSO Global.

A continuación se muestran los pseudocódigos de los algoritmos PSO local y global:

PSO Local

S ← inicializaEnjambre()
mientras no condición de parada **hacer**
    para i=1 hasta tamaño(S) **hacer**
        Evaluar cada partícula x_i del enjambre S
        si fitness(x_i) mejor que fitness($pBest_i$) **entonces**
            $pBest_i \leftarrow x_i; fitness(pBest_i) \leftarrow fitness(x_i)$
        fin si
    fin para
    para i=1 hasta tamaño(S) **hacer**
        Escoger $lBest_i$
        $v_i = \omega \cdot v_i^k + \varphi_1 \cdot rand_1 \cdot (pBest_i - x_i\ ) + \varphi_2 \cdot rand_2 \cdot (g_i - x_i)$
        $x_i \leftarrow x_i + v_i$
    fin para
fin mientras

En el PSO Local, se calcula la distancia entre la posición actual de una partícula y la posición de la mejor partícula encontrada en el entorno local de la primera. El entorno local consiste en las partículas inmediatamente cercanas en la topología del enjambre.

PSO Global

S ← inicializaEnjambre()
mientras no condición de parada **hacer**
    para i=1 hasta tamaño(S) **hacer**
        Evaluar cada partícula x_i del enjambre S
        si fitness(x_i) mejor que fitness($pBest_i$) **entonces**
            $pBest_i \leftarrow x_i; fitness(pBest_i) \leftarrow fitness(x_i)$
        fin si
        si fitness($pBest_i$) mejor que fitness($gBest_i$) **entonces**
            $gBest_i \leftarrow pBest_i; fitness(gBest_i) \leftarrow fitness(pBest_i)$
        fin si
    fin para
    para i=1 hasta tamaño(S) **hacer**
        $v_i = \omega \cdot v_i^k + \varphi_1 \cdot rand_1 \cdot (pBest_i - x_i) + \varphi_2 \cdot rand_2 \cdot (g_i - x_i)$
        $x_i \leftarrow x_i + v_i$
    fin para
fin mientras

Para el PSO Global, la distancia en el componente social viene dada por la diferencia entre la posición de la partícula actual y la posición de la mejor partícula encontrada en el enjambre completo $gBest_i$.

La versión global converge más rápido pues la visibilidad de cada partícula es mejor y se acercan más a la mejor del enjambre. Por esta razón también cae más fácilmente en óptimos locales. El comportamiento de la versión local es el contrario, es decir, le cuesta más converger favoreciendo en este caso la diversificación, pero no cae fácilmente en óptimos locales.

2.2.3 Topologías del enjambre

Un aspecto muy importante a considerar es la manera en la que una partícula interacciona con las demás partículas de su vecindario. El desarrollo de una partícula depende tanto de la topología del enjambre como de la versión del algoritmo. Las topologías definen el entorno de interacción de una partícula individual con su vecindario. Los entornos pueden ser de dos tipos:

- Geográficos: se calcula la distancia de la partícula actual al resto y se toman las más cercanas para componer su entorno;
- Sociales: se define a priori una lista de vecinos para partícula, independientemente de su posición en el espacio.

En la Figura 6 se muestran gráficamente ejemplos de entornos geográficos y sociales. Los entornos sociales son los más empleados. Una vez definido un entorno, es necesario definir su tamaño, habitualmente valores de 3 y 5 pues suelen tener un buen comportamiento. Obviamente, cuando el tamaño es todo el enjambre el entorno es a la vez geográfico y social, obteniendo así un PSO global.

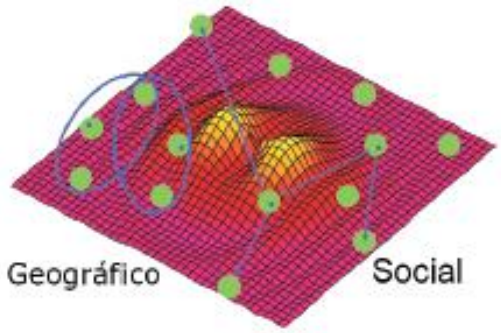


Figura 6. Topologías del enjambre

Es posible configurar varios tipos de topologías dentro de un entorno social. Una de las más comunes y que inicialmente fue muy utilizada es la sociometría gBest [29] (Figura 7, izquierda). En ésta se establece un vecindario global en el que toda partícula es vecina de la totalidad del enjambre en sí, lo que sería un PSO global, favoreciendo la explotación de espacio de soluciones. Otro ejemplo es la sociometría lBest (Figura 7, derecha), que es propuesta para tratar con problemas de mayor dificultad. En ésta cada partícula es conectada con sus vecinas inmediatas en el propio enjambre. Así, por ejemplo, la partícula p_i es vecina de la partícula p_{i-1} y de p_{i+1}. Esta topología ofrece la ventaja de establecer "sub-enjambres" que realizan búsquedas en diversas regiones del espacio del problema y de esta forma se favorece la exploración.

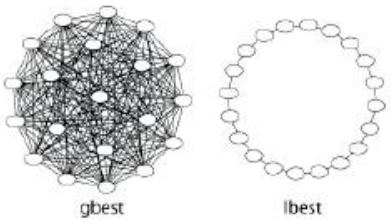


Figura 7. Ejemplos de sociometrías en el enjambre

No sólo existen dos tipos de topologías, hay gran variedad de estas, hay unas más usadas que otras, como en la Figura 8 donde se muestra la topología de Von Neumann a la izquierda y la cuadrada a la derecha.

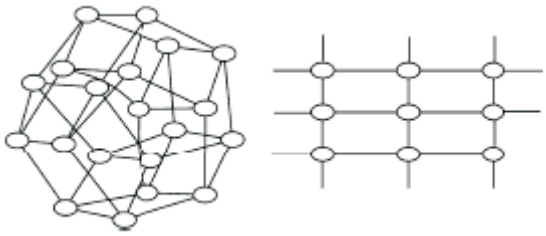


Figura 8. Topología de Von Neumann o cuadrada

2.2.4 MOPSO

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 $f: S \to T$, donde $S \subset \mathbb{R}^k$ y . 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. Pensemos por ejemplo en el caso de un avión con dos objetivos que fueran su velocidad y el ahorro de combustible: un aumento de la velocidad traería consigo una disminución del ahorro. 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.

Para tratar el problema comentado del conflicto entre objetivos se han utilizado varios métodos:

- Métodos basados en el concepto de eficiencia de Pareto;
- Métodos basados en la combinación de objetivos;
- Métodos basados en la asignación de prioridades.

El PSO también se ha aplicado a problemas multiobjetivo (MOPSO) cuyo algoritmo es exactamente igual al que hemos comentado en el apartado anterior. La diferencia entre PSO y MOPSO es como trata el conflicto entre objetivos, ya que, 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:

$$y_{A,i} \leq y_{P,i} \forall\, i \qquad (8)$$

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.

Un ejemplo del frente de Pareto se muestra en la Figura 9, para un problema de minimización de dos objetivos, donde las soluciones del frente

(puntos amarillos) superan a todas las demás soluciones (puntos negros) en al menos una de los dos objetivos considerados.

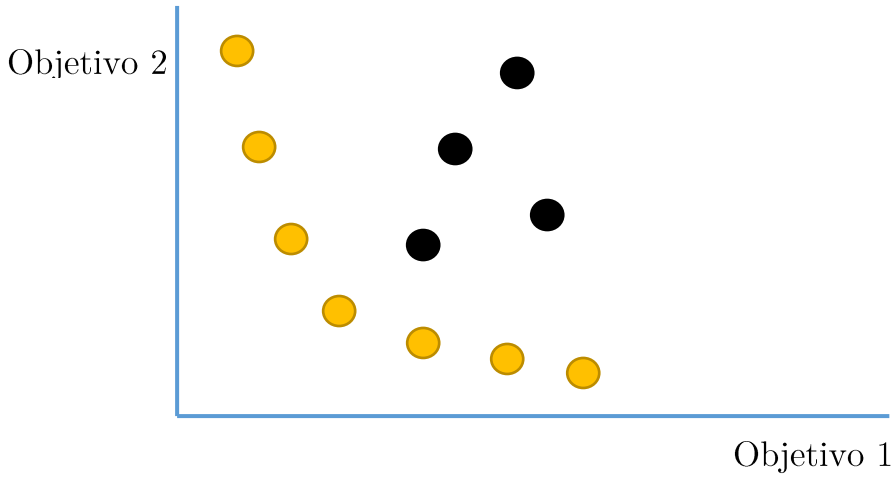


Figura 9. Ejemplo de frente de Pareto para dos objetivos



Capitulo 3: DISEÑO DEL MODELO

El objeto de nuestro estudio va a ser ajustar los parámetros de un modelo capaz de simular la transmisión del virus infectocontagioso de la varicela en un tiempo razonable.

Cuando se simula una enfermedad interesa generalmente diferenciar a cada individuo de la población, pues cada uno tiene unas características intrínsecas que le definen, como por ejemplo: edad, sexo, etc. Además, como podemos ver en la Figura 9, cada individuo posee un estado respecto a la enfermedad, ya que esta interactúa de manera única con cada uno de ellos. De esta forma, podemos diferenciar individuos: susceptibles, recuperados, etc.

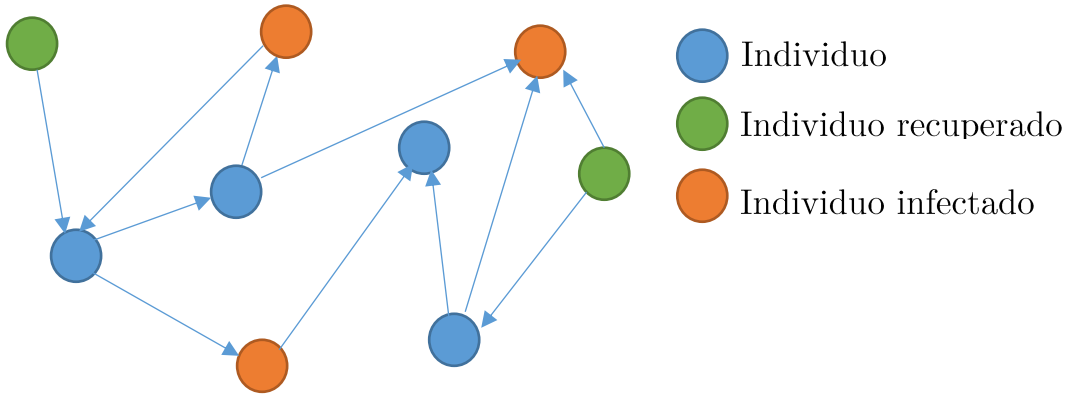


Figura 10. Ejemplo de red vírica

Como es obvio cada virus posee unas características propias que lo definen tales como: probabilidad de contagio, grupos de edad más o menos susceptibles a éste, etc.

Un modelo de red vírico como el de la varicela está compuesto por una serie de parámetros, los cuales hacen que el virus adquiera un comportamiento u otro. El comportamiento que nos interesa modelar es el que éste posee en la vida real ya que esto es lo que permite predecir dicho comportamiento en el futuro y así poder estudiar la evolución del virus en la población. Gracias a un análisis exhaustivo de la varicela se pueden planificar ingresos en hospitales, consultas médicas, etc. Además, un modelo de redes permite

el estudio de la rentabilidad y del impacto económico para una empresa, por ejemplo, farmacéutica, que estuviese interesada en distribuir la medicación para la enfermedad en cuestión, número de vacunas a repartir, a qué segmento de la población, etc.

Pero para que el modelo de varicela sea capaz de dotar al virus de un comportamiento lo más real posible es necesario parametrizar al modelo con los valores adecuados. Esta tarea no es nada fácil puesto que generalmente los parámetros son desconocidos y además, como se dijo en la introducción, la búsqueda de los valores adecuados para cada parámetro (ajuste del modelo) es computacionalmente costosa.

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. Esta interconexión es de vital importancia en el modelo porque es la que 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 otro comportamiento irreal.

La varicela no afecta a todos los individuos de la población de igual manera. Es sabido que dicho virus 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. Estos parámetros también son cruciales para modelar el virus adecuadamente ya que una probabilidad de contagio muy baja conseguiría que hubiese pocos infectados, lo que podría implicar que la propia enfermedad desapareciese. Por el contrario, si la probabilidad es muy alta estaríamos ante una pandemia, ya que sería muy fácil contagiarse de varicela. En ambos casos estamos ante un comportamiento que no se adecúa al comportamiento real del virus.

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%

Tabla 1. Porcentaje de infectados según grupo de edad

Contamos con datos reales proporcionados por la Comunidad Valenciana de los de los diez últimos años [31]. Estos se resumen en: porcentaje de infectados según grupo de edad (Tabla 1), ya que cada grupo tiene una probabilidad diferente de contagiarse, resultando algunos de estos más susceptibles que otros y el otro dato que se proporciona es la cantidad de infectados por cada 100000 habitantes (Figura 10), especificados para cada semana del año. Contamos por tanto con 52 semanas con una cantidad de infectados.

Con los datos facilitados por la Comunidad Valenciana se sabe que el virus se comporta de manera más o menos contagiosa dependiendo de la semana del año en la que nos encontramos. Por ello en el modelo se define una probabilidad de contagio específica para cada semana del año, es decir, tenemos 52 semanas por año, por lo que cada una de éstas está parametrizada en el modelo con una probabilidad de contagio específica para dicha semana.

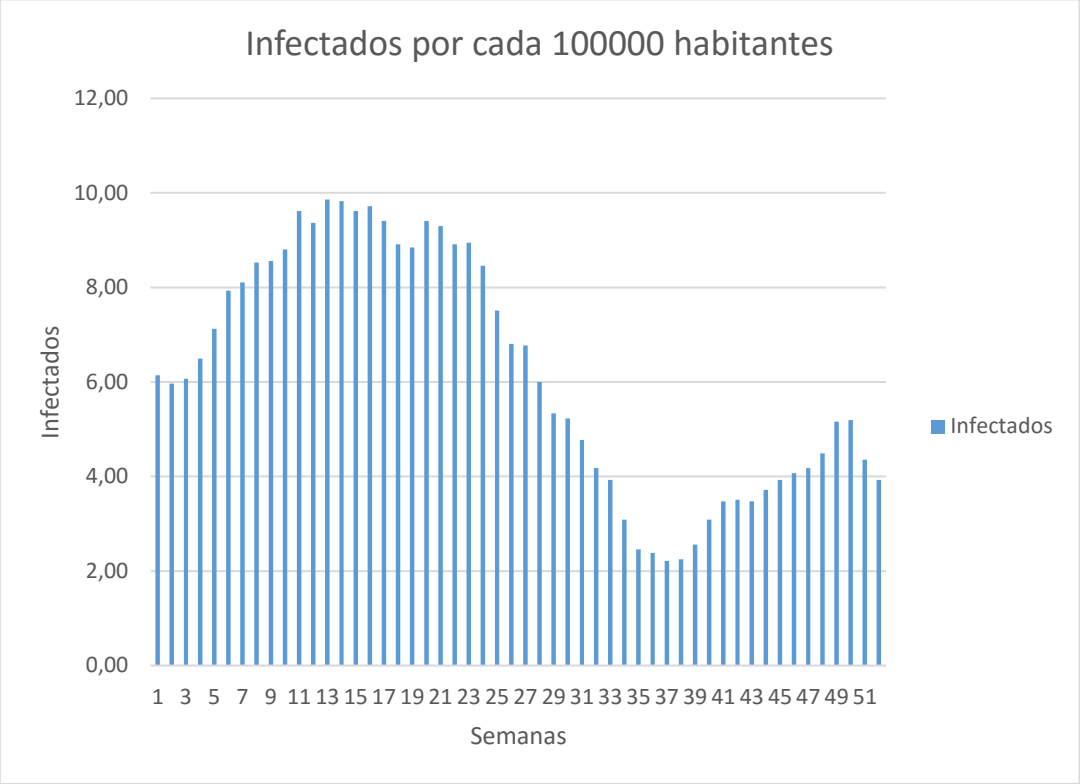


Figura 11. Infectados por varicela

En resumen, el modelo de la varicela consta de 58 parámetros que debemos ajustar debidamente para que podamos obtener un comportamiento lo más ajustado a la realidad posible:

- Grado de conectividad media de los nodos;
- Probabilidad de contagio de individuos de 6 a 12 meses;
- Probabilidad de contagio de individuos de 1 a 4 años;
- Probabilidad de contagio de individuos de 5 a 9 años;
- Probabilidad de contagio de individuos de 10 a 14 años;
- Probabilidad de contagio de individuos de 15 o más años;
- Probabilidad de contagio en la semana 1 del año;
- …
- Probabilidad de contagio en la semana 52 del año.

3.1 Ajuste del modelo

Al hablar de ajuste nos referimos a la capacidad de medir la calidad de una solución y así poder determinar si los parámetros obtenidos para un modelo son o no adecuados para la simulación realista del virus en cuestión.

Si tratamos de abordar el ajuste de este modelo de una forma tradicional, como sería realizando una búsqueda exhaustiva de cada parámetro, estaríamos hablando de un tiempo de cómputo de más de 3 años, algo totalmente inabordable. 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.

Entonces necesitamos una "herramienta" más que nos ayude a que los problemas a evaluar por el modelo sean de mayor calidad a medida que vamos avanzando en el tiempo, es decir, necesitamos optimizar la búsqueda de los parámetros anteriormente mencionados. Por ello se ha optado por un algoritmo metaheurístico de optimización como es MOPSO, ya que su uso permite reducir el tiempo de cómputo sustancialmente. Una metaheurística no asegura encontrar el óptimo en todos los casos, pero sí que en la mayoría de los casos podemos acercarnos suficientemente a éste y obtener un ajuste válido para el modelo.

Para poder realizar el ajuste y medir la bondad de las simulaciones realizadas por el modelo necesitamos de los datos reales del virus varicela para comparar dichas simulaciones con el comportamiento real de contagio del virus.

Como ya se ha dicho, disponemos de conjuntos de datos que definen el comportamiento del virus: el porcentaje de infectados por grupo de edad según la Comunidad Valenciana (Tabla 1) y la evolución por semanas (Figura 10).

Con estos dos conjuntos de datos podemos definir los dos objetivos en MOPSO que necesitamos para realizar el ajuste:

- Porcentaje de infectados por grupo de edad;
- Infectados por semana.

Cuando el modelo de varicela termina de simular el comportamiento del virus en base a los parámetros anteriormente mencionados, éste nos facilita los conjuntos de datos calculados. Es entonces cuando procedemos a comparar los datos reales con los simulados.

Cada parámetro se comprueba por separado. En primer lugar se comprueban los datos de los infectados por semana. Para realizar dicha comparación usamos el error cuadrático medio (RMSE) entre los infectados por semana calculados por el modelo y los infectados por semana de los casos reales facilitados por la Comunidad Valenciana. Luego se calcula el RMSE del porcentaje de infectados por grupo de edad calculado por el modelo y en los casos reales

El objetivo es encontrar un RMSE menor que 10 para cada uno de los parámetros.

Capitulo 4: ESTRUCTURA DEL SISTEMA DE AJUSTE DEL MODELO

En ese apartado vamos a describir el procedimiento y la arquitectura utilizados para ajustar el modelo de varicela anteriormente diseñado.

En primer lugar describiremos la arquitectura del sistema, con los elementos que lo componen. Después dedicaremos un subapartado a cada uno de ellos para describirlos con mayor nivel de detalle.

4.1 Arquitectura del sistema

Los elementos que componen la arquitectura son:

- **Generador de soluciones**: Es el 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 encargado de 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:** Una vez lo ejecutamos, es el encargado de mostrar gráficamente las soluciones que creamos convenientes con el fin de ver realmente el ajuste obtenido por el generador de soluciones. El visualizador se ejecuta bajo demanda como parte del ajuste de los cálculos. Es opcional y muy útil.

En la Figura 12 podemos ver un esquema de la arquitectura.

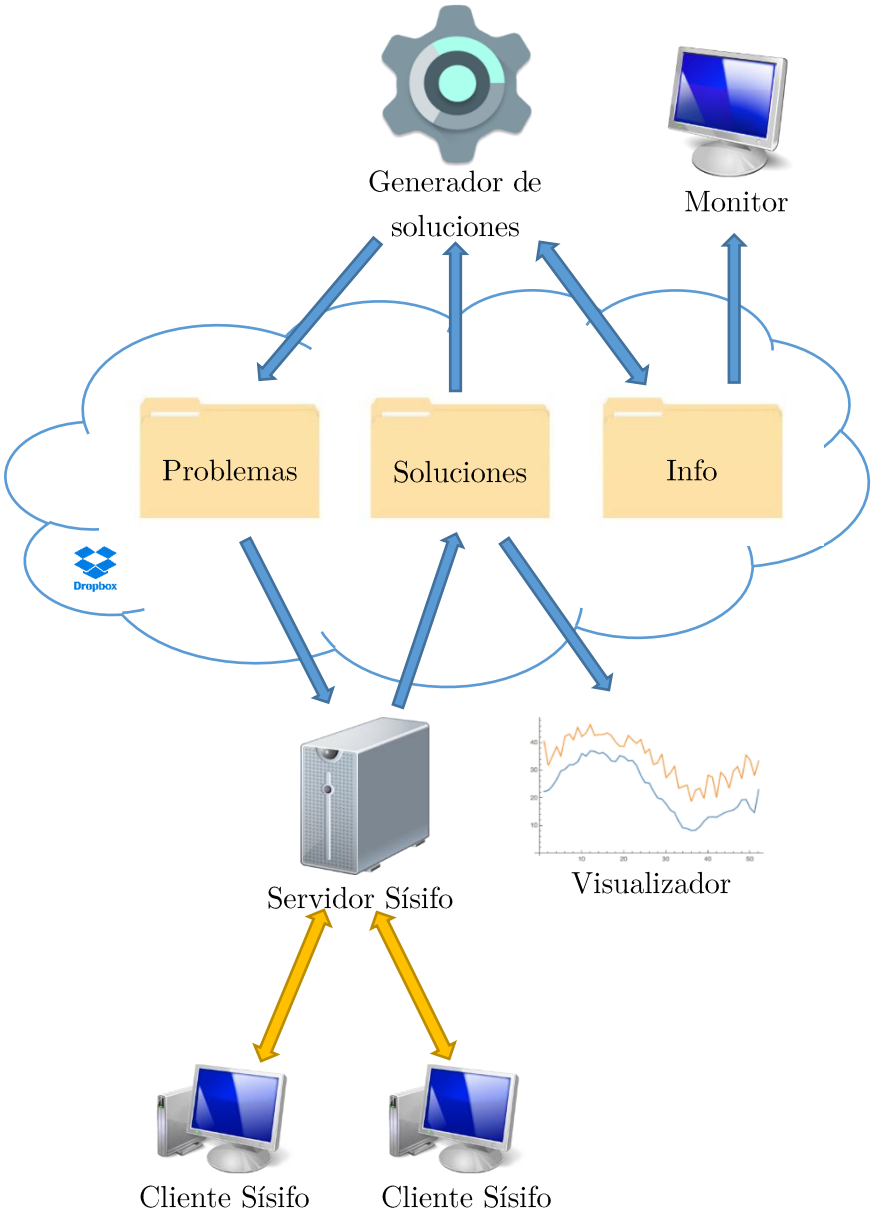


Figura 12. Arquitectura del sistema

4.2 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. Veamos cómo funciona:

- En primer lugar, el GS carga su configuración del fichero config.txt (ver apartado 4.2.1). En caso de no existir, se inicializa con la configuración por defecto;
- Se genera el número de partículas especificadas en la configuración que hemos cargado en el paso anterior;
- Se inicializan las velocidades de las partículas acorde a la ecuación 7;
- Se generan las fichas de problemas (ver apartado 4.2.2) en el directorio de problemas. Para los nombres de las fichas se toma el instante en el que se va a generar el fichero (día, mes, hora, minutos, segundo y milésimas), y el orden de generación de éste con la extensión .pro. Este nombre es único y lo distinguimos añadiendo un prefijo. Por ejemplo: P_26_02_28_15_069000_53.pro;
- El generador se queda a la espera de que se le devuelvan las soluciones de las partículas generadas, que estarán en el directorio soluciones con el nombre del fichero original pero con la extensión .sol (Ver apartado 4.2.3). Si hay ficheros nuevos, se calcula su ajuste tal como se ha descrito en el apartado 3.1.Cuando todas las partículas están analizadas continuamos con otra nueva iteración;
- Con esta primera iteración ya tenemos un conjunto de partículas en el frente de Pareto. Entonces se eligen, aleatoriamente, un mejor global para cada partícula y se actualizan las partículas. Esta actualización se hace 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 hemos elegido aleatoriamente;
- A continuación entramos en el bucle principal de las iteraciones donde se generan las nuevas fichas, esperamos a que estén resueltas

las simulaciones y las analizamos, actualizamos las nuevas velocidades y las nuevas partículas (véanse las ecuaciones 6 y 7);

- Llegados a este punto todo el proceso vuelve a empezar hasta alcanzar la condición de parada, que será un número de iteraciones fijo. En nuestro caso hemos utilizado 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. Este fichero se guarda en la carpeta Info. Además verifica que config.txt sigue intacto, si detecta cambios cargará la nueva configuración.

4.2.1 Configuración

La configuración del generador de soluciones viene dada por parámetros que el algoritmo MOPSO necesita para comenzar con el proceso de generación de parámetros y ajuste del modelo. Son los siguientes:

- **ITMAX**: Número máximo de iteraciones del algoritmo;
- **Partículas**: Número de partículas que el algoritmo va a generar en cada una de las iteraciones;
- **Nrepositorio**: Número máximo de partículas que vamos guardando como las mejores;
- **Tasa_mut**: Es la probabilidad de mutación que se aplica a cada partícula;
- **Dist_mut**: Es la distribución de la mutación;
- **Aleat_elec_global**: Aleatoriedad en la elección del mejor global;
- **Nodos**: Numero de nodos de la red (visto en el apartado 3);
- **Turnos**: Número de turnos a simular (visto en al apartado 3).

Todos estos parámetros se encuentran en un fichero de texto llamado config.txt, que se almacena en la carpeta “Info” del directorio compartido.

En caso de no existir un fichero config.txt, el generador crea uno con una configuración por defecto para cada uno de los parámetros aquí mencionados.

En la Figura 13 podemos ver un ejemplo de fichero de configuración.

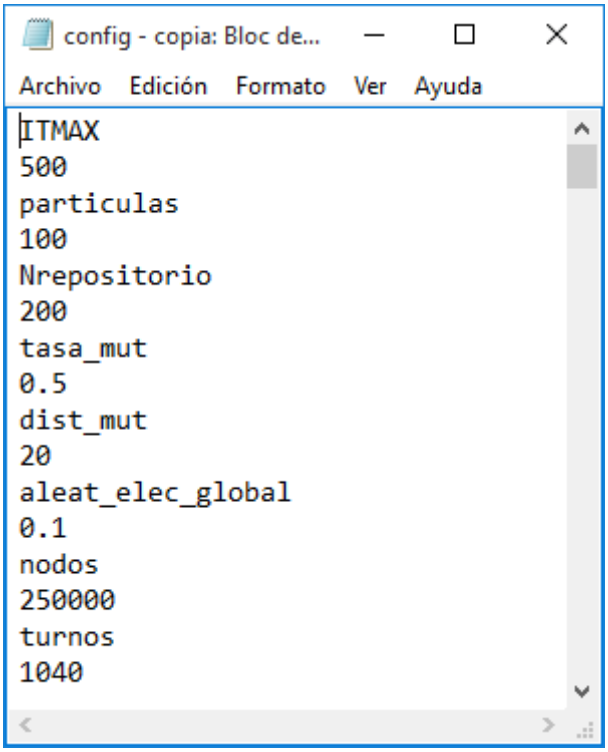


Figura 13. Ejemplo de configuración

4.2.2 Fichas de problemas

Las fichas de problemas son unos ficheros de texto. Cada uno de estos contiene todos los parámetros que el modelo de varicela necesita para simular la enfermedad (capítulo 3 de la memoria). Estas fichas son leídas por el modelo de varicela y, con ellas, el modelo puede simular el comportamiento del virus. Podemos ver un ejemplo de una ficha de problemas en la Figura 14.

Los campos que tiene cada ficha son:

- **identificador_de_prueba:** Identificador que hace única a cada ficha de problemas;
- **generador_de_red**: Es la semilla del generador de números pseudoaletorio; para este modelo es cero;
- **turnos_a_simular:** Define los turnos que van a ser simulados por el modelo de varicela. El valor es fijo para este modelo, 1040, que representa semanas por lo que equivale a 20 años;

- **Probabilidad_Recuperado_a_Latente**: Es la probabilidad que tiene un individuo que está recuperado de varicela de ser de nuevo susceptible a ésta;
- **numero_de_nodos**: Número de nodos existentes en la red. Es un número fijo igual a 250000;
- **Cobertura_vacunal_primer_anyo**: Parámetro para introducir la vacunación en el primer año. Ahora mismo este parámetro se ignora;
- **Cobertura_vacunal_segundo_anyo**: Igual que la cobertura del primer año;
- **Cobertura_vacunal_tercer_anyo**: Igual que la cobertura del primer año;
- **RESERVADO_1 – RESERVADO_6**: Estos 6 parámetros están reservados para futuros propósitos, no entran en juego en nuestros cálculos ni son evaluados;
- **turno_inicio_vacuna_1**: Como no hay vacunación en el modelo, al igual que la cobertura se ignora;
- **turno_inicio_vacuna_2**: Como no hay vacunación en el modelo, al igual que la cobertura se ignora;
- **turno_inicio_vacuna_3**: Como no hay vacunación en el modelo, al igual que la cobertura se ignora;
- **grado_medio_de_los_nodos**: Este parámetro indica la conectividad media entre los nodos de la red;
- **Alfa1_6_a_12_meses**: Probabilidad de contagio en el rango de edad de 6 a 12 meses;
- **Alfa2_1_a_4_anyos**: Probabilidad de contagio en el rango de edad de 1 a 4 años;
- **Alfa2_5_a_9_anyos**: Probabilidad de contagio en el rango de edad de 5 a 9 años;
- **Alfa2_10_a_14_anyos**: Probabilidad de contagio en el rango de edad de 10 a 14 años;
- **Alfa2_15_o_mas_anyos**: Probabilidad de contagio en el rango de edad de 15 o más años;

- **Contagio_S1** – **Contagio_S52**: probabilidad de infección dependiendo de la semana del año (semana 1 a semana 52 del año).

```
identificador_de_prueba
P_03_20_03_59_025000_1
generador_de_red
0
turnos_a_simular
1040.0
Probabilidad_Recuperado_a_Latente
0.00173807447076
numero_de_nodos
250000.0
grado_medio_de_los_nodos
20.0609048611
Cobertura_vacunal_primer_anyo
0
Cobertura_vacunal_segundo_anyo
0
Cobertura_vacunal_tercer_anyo
0
RESERVADO_1
0
RESERVADO_2
0
turno_inicio_vacuna_1
99999999
turno_inicio_vacuna_2
99999999
turno_inicio_vacuna_3
99999999
RESERVADO_3
0
RESERVADO_4
0
RESERVADO_5
0
RESERVADO_6
0
Alfa1_6_a_12_meses
0.0285
Alfa2_1_a_4_anyos
7.96977997032
Alfa2_5_a_9_anyos
1.55473471409
Alfa2_10_a_14_anyos
0.0598797795658
Alfa2_15_o_mas_anyos
0.0688057579027
Contagio_S1
12.1000158726
Contagio_S2
5.89400814807
```

Figura 14. Ficha de problemas

En la Figura 14 faltarían los parámetros desde Contagio_S3 a Contagio_S52 pero por extensión de la imagen no se incluyó.

4.2.3 Ficheros solución

El resultado de la simulación de cada modelo tiene la forma de un fichero de texto con una extensión que lo identifica como tal (.sol). Dicho resultado contiene tres partes claramente diferenciadas: parámetros del problema evaluado, resultado de la simulación de dicho problema (la "solución" propiamente dicha), y un conjunto de valores auxiliares que se usan para información accesoria. El tamaño del fichero de texto depende del valor del parámetro turnos, en nuestro caso son 1040. La Figura 15 muestra un ejemplo de la estructura del fichero solución.

```
problema =
{P_09_01_41_31_043000_53,0,1041,0.000298798361568,250000,26.5190201137,0
,0,0,0,0,99999999,99999999,99999999,0,0,0,0,0,0,0,0,8.68016837751,0.0678
748192061,0.0475,0.0438708446003,0.0511880040255};

solucion = {{ {2976, 1898, 1872, 408, 401, 427, 408, 400, 410, 421, 408,
416, 135, 134, 158, 139, 128, 140, 124, 129, 126, 135, 146, 140, 120,
117, 117, 141, 117, 118, 149, 108, 145, 141, 122, 155, 135, 148, 135,
127, 118, 124, 129, 126, 127, 131, 141, 119, 100, 137, 152, 152, 133,
126, 124, 111, 141, 129, 128, 134, 123, 118, 124, 131, 136, 133, 130,
127, 110, 126, 137, 104, 108, 107, 105, 98, 110, 94, 107, 119, 90, 87,
89, 81, 77, 79, 49, 55, 51, 37, 44, 42, 32, 24, 21, 6, 5, 9, 2, 3, 10,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}, {5, 32, 12, 9,
12, 5, 7, 2, 8, 8, 9, 5, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1,
0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
2, 2, 2, 1, 3, 1, 2, 0, 2, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}, {34,
1071, 1114, 2589, 2593, 2574, 2591, 2602, 2588, 2575, 2589, 2583, 2871,
2871, 2846, 2866, 2877, 2864, 2880, 2874, 2877, 2867, 2855, 2860, 2879,
2882, 2881, 2856, 2879, 2878, 2846, 2886, 2848, 2851, 2869, 2835, 2853,
2839, 2850, 2856, 2863, 2855, 2847, 2847, 2842, 2833, 2820, 2837, 2850
```

Figura 15. Muestra de un fichero solución

Los valores del problema son una lista numérica de elementos separados por comas y llaves, con la información propia de la prueba simulada (número de turnos, nodos, probabilidad de contagio, etc.) ya descritos en capítulos anteriores (primeras líneas de la figura 15).

Los valores de la "solución" son una lista numérica de valores separados por comas y llaves, para cada uno de los 1040 turnos. Estos valores numéricos son en concreto la cantidad de infectados por semana.

La parte "auxiliar" (que no aparece en la figura) contiene en nuestro caso el tiempo total de ejecución de la simulación y una comprobación de integridad de los datos.

Así, por cada una de las simulaciones realizadas, tendremos un fichero de solución que hemos de evaluar a fin de descubrir cuál de ellos reproduce de manera más fiel el comportamiento ideal del modelo.

4.3 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 conjunto de ordenadores heterogéneos (Figura16). Éste asigna tareas a un conjunto de PCs, espera a que las tareas terminen, y recoge los resultados para su posterior análisis. Sísifo está pensado para poder instalarse y configurarse en una red local en pocos minutos y apenas necesita mantenimiento, a cambio de ofrecer limitado control de acceso y seguridad, por lo que es apto en la práctica sólo en entornos controlados como una intranet.

Cada cliente Sísifo tiene corriendo el modelo de varicela, de modo que cuando el servidor encuentra problemas sin resolver, éste se lo asigna a uno de sus clientes libres para que lo resuelva. Cuando el fichero está resuelto, le devuelve la solución al servidor y éste se encarga de almacenar dicha solución debidamente.

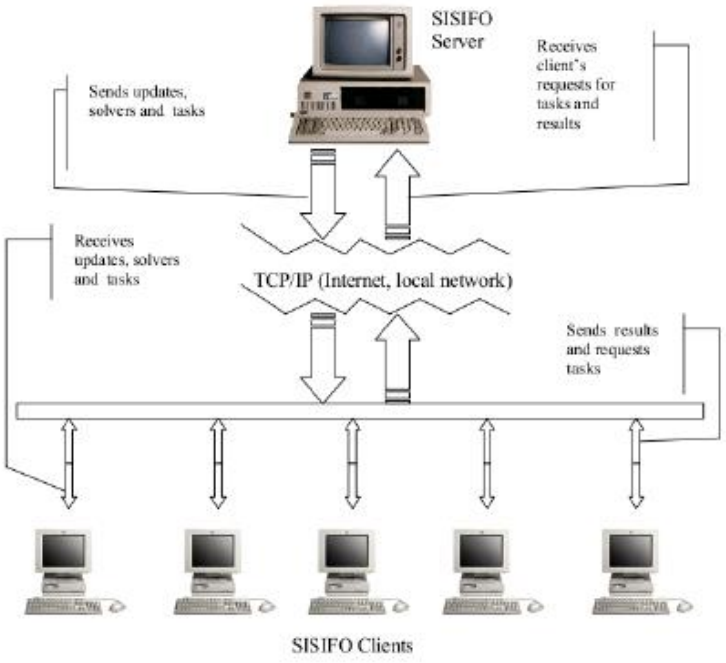


Figura 16. Arquitectura de Sísifo

Respecto a las 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.

4.4 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. Dropbox es un software que enlaza todas las computadoras mediante una sola carpeta, lo cual constituye una manera fácil de respaldar y sincronizar los archivos. El servicio cliente de Dropbox permite a los usuarios dejar cualquier archivo en una carpeta designada. Ese archivo es sincronizado en la nube y en todas las demás computadoras del cliente de Dropbox.

De este modo, Dropbox contiene un directorio compartido el cual está compuesto por distintas carpetas/directorios:

- **Problemas**: En esta carpeta se guardan todos los problemas pendientes de resolver, es decir, aquí el GS deja sus fichas generadas y Sísifo las recoge;
- **Soluciones**: En esta carpeta se almacenan todos los ficheros solución. El servidor de Sísifo deja el fichero .sol y es el GS quien lo toma para analizarlo debidamente;
- **Info**: En este directorio se almacenan los ficheros de información o configuración anteriormente mencionados:
    - **Config.txt;**
    - **ficherosRestantes.txt;**
    - **Monitor.txt.**

El utilizar un servicio de almacenamiento en la nube, como Dropbox, proporciona un muy bajo acoplamiento entre los sistemas que lo utilizan, ya que ninguno de estos sabe sobre la existencia del otro y además los permite operar en redes completamente diferentes, lo ofrece una gran ventaja: se pueden reemplazar sin que afecte al resto del sistema.

4.5 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. Este monitor se convierte en los ojos del investigador mientras todo el proceso de ajuste está en marcha ya que, gracias a éste, podemos ver qué está ocurriendo actualmente en los cálculos.

La arquitectura del monitor lo componen los siguientes elementos:

- Servlets;
- JSP;
- Explorador web;
- Directorio compartido Info.

La Figura 17 muestra cómo interactúan los elementos mencionados.

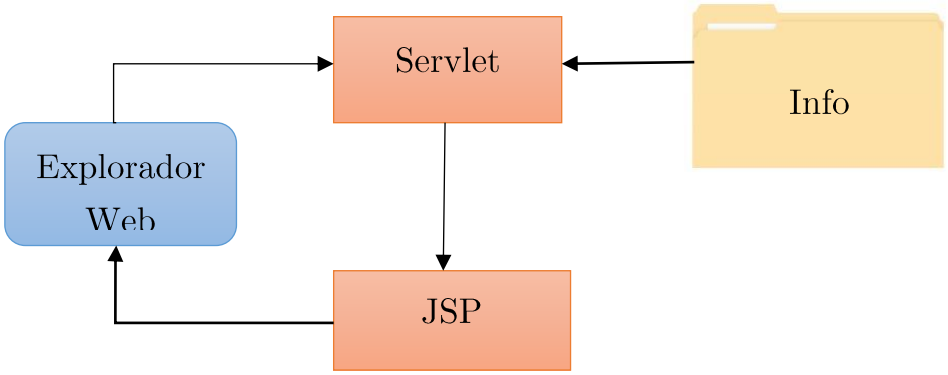


Figura 17. Arquitectura del monitor

Para que el monitor pueda cumplir su función sólo necesita leer del directorio Info los ficheros: monitor.txt y ficherosRestantes.txt. Esta información es procesada por la arquitectura mostrada en la Figura 17 y el explorador web muestra al usuario la siguiente información:

- **Ficheros restantes**: Número de ficheros que quedan por resolver en la iteración actual;
- **Iteraciones:** Número actual de iteración respecto al total;

- **Parámetros de configuración de MOPSO**
  - Nparticulas;
  - Nrepositorio;
  - Tasa_mut;
  - Dist_mut;
  - Aleat_elec_global
- **Ficheros en el frente de Pareto**: Soluciones que se encuentran actualmente en el frente de Pareto, además también visualiza los errores en ambos objetivos.
- **Parámetros de la red**: Para cada fichero en el frente de Pareto se muestran los parámetros que configuran la red del modelo además de su valor en cada uno de ellos.

Para acceder al monitor basta con abrir un explorador cualquiera de internet y acceder a la url: varicela.ddns.net y si el servidor Apache Tomcat está corriendo podremos visualizar el monitor tal como muestra la Figura 18.

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.

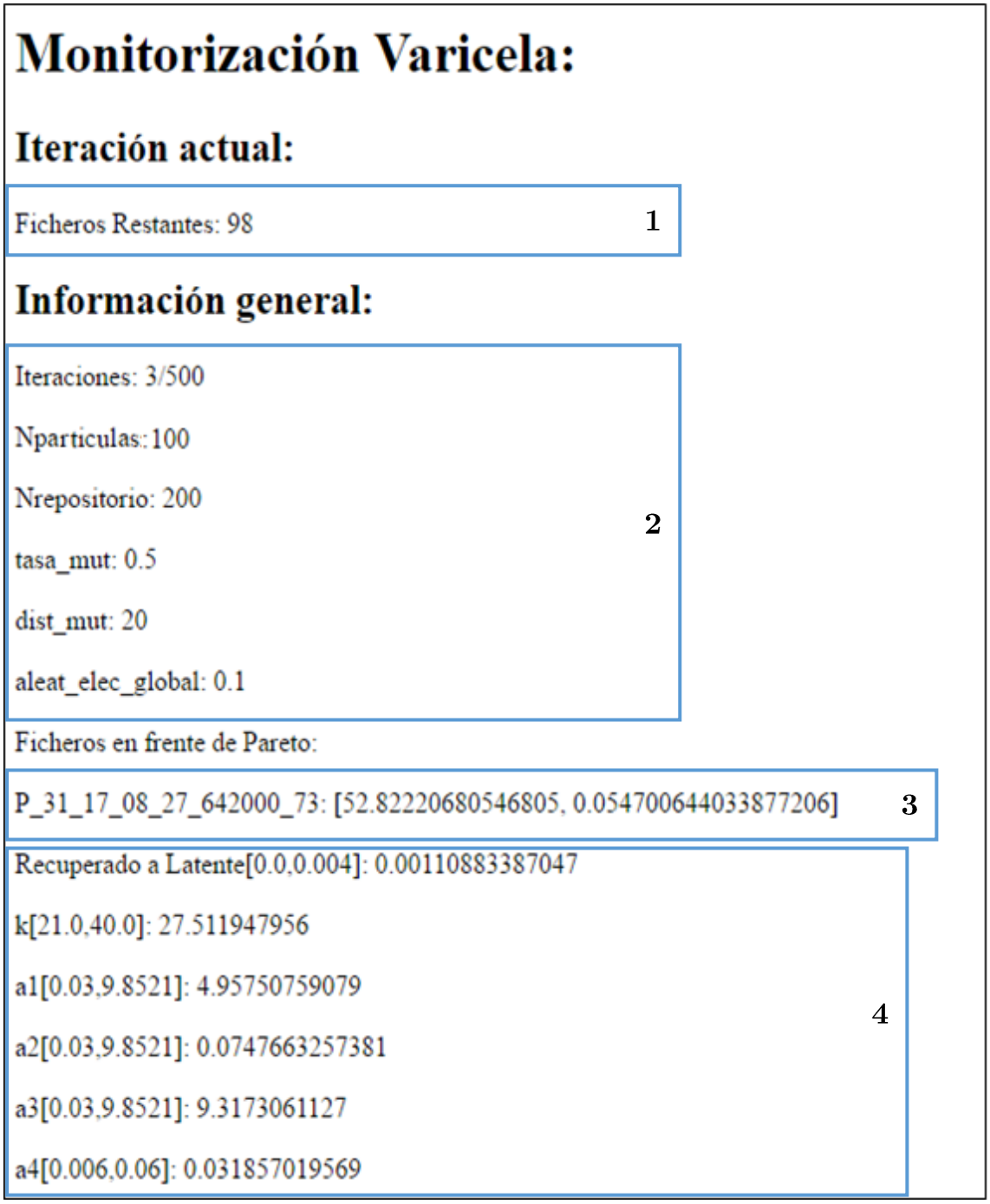


Figura 18. Monitor del sistema

Los elementos de la Figura 18 son:

1) Ficheros restantes que quedan por resolver en la iteración actual, en la imagen quedarían 98;
2) Información de configuración de MOPSO;
3) Lista de ficheros en el frente de Pareto . Cada entrada está compuesta por:
    a. El nombre del fichero P_31_17_08_27_6420000_73;
    b. RMSE de cada objetivo: [RMSE del objetivo 1, RMSE del objetivo2].
4) Valores de los parámetros de la red. Nótese que faltaría la probabilidad de contagio de cada semana (1-52), pero por extensión de la imagen se incluye.

4.6 Visualizador de soluciones

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

Este visualizador tiene acceso al directorio compartido soluciones, para que éste pueda procesarlas cuando sea oportuno. Con este visualizador se muestran los resultados obtenidos en el apartado 5. Cuando ejecutamos el visualizador vemos una imagen como la de la Figura 19.

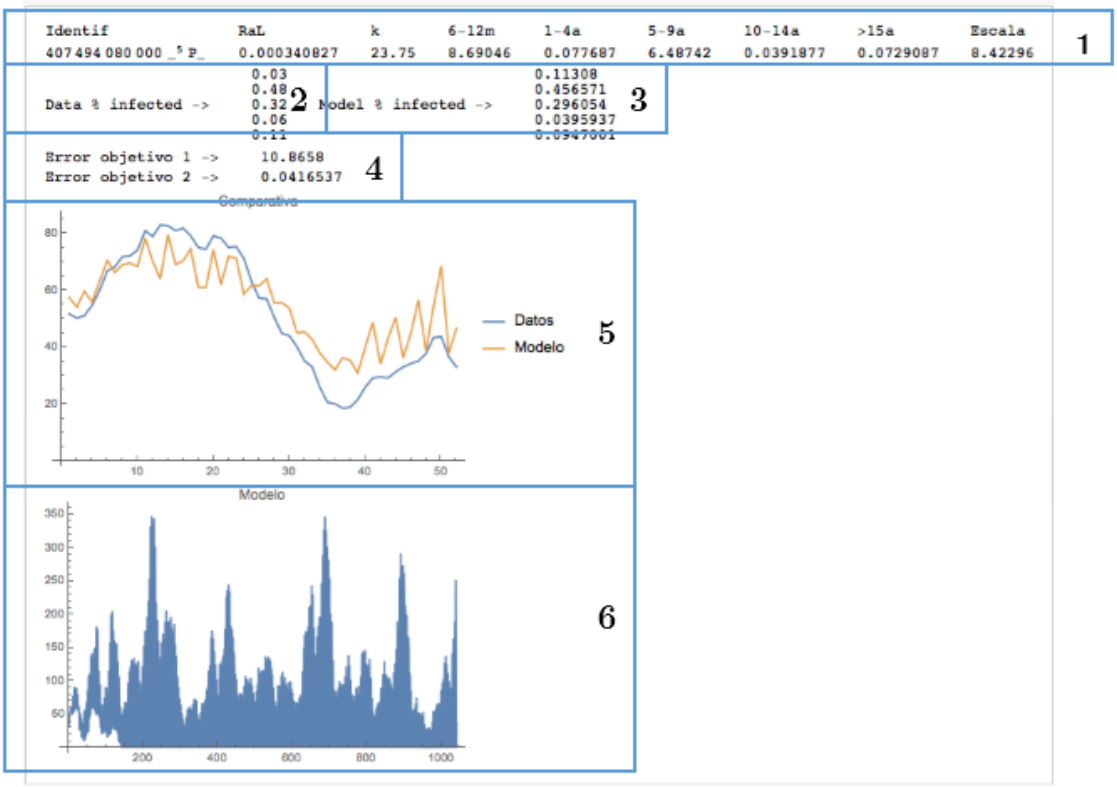


Figura 19. Visualización de una solución

En esta figura aparecen:

1) Parámetros de interés del modelo;
2) Porcentaje de infectados por grupo de edad;
3) Porcentaje de infectados por grupo de edad calculados por el modelo;
4) Errores en los objetivos:
    a. Error objetivo 1: RMSE entre las gráficas Datos (en azul) y Modelo (amarilla) que aparecen en el bloque 5;
    b. Error objetivo 2: RMSE entre los datos reales (bloque 2) y datos calculados por el modelo (bloque 3).
5) Muestra el ajuste del primer objetivo:
    a. La línea que figura como modelo es la media de los últimos 10 años de los infectados totales de varicela que devuelve el modelo;
    b. La línea que figura como datos son los casos reportados (datos).

6) Si vemos actividad en esta gráfica nos asegura que la enfermedad no ha desaparecido y por lo tanto lo obtenido es coherente.

4.7 Funcionamiento del sistema

Una vez explicado el funcionamiento de cada elemento del sistema se va a hacer una descripción del ciclo de funcionamiento del proceso completo.

El sistema se pone en funcionamiento cuando arranca el generador de soluciones. Éste carga la configuración con la cuál va a trabajar e inicializa los ficheros restantes por resolver en el directorio Info. Una vez configurado e inicializado debidamente, este crea N fichas de problemas y las almacena en el directorio de Problemas. Acto seguido se comienza a monitorizar el directorio de soluciones para ver si las fichas de problemas que inicialmente se crearon han sido resueltas.

Entonces el servidor Sísifo entra en juego monitorizando el directorio de Problemas, con el fin de resolver las nuevas fichas que el generador ha creado. Cuando el servidor de Sísifo detecta nuevos problemas que resolver, comienza a repartir dichas fichas entre sus clientes. Cada cliente tiene en ejecución el modelo de varicela. Una vez que un cliente termina de resolver la tarea que el servidor le asignó previamente, éste devuelve el problema resuelto. Cada vez que el generador detecta una solución nueva, actualiza en el directorio Info el fichero llamado, ficherosRestantes.txt con el número de ficheros restantes que quedan por analizar. Seguidamente se realiza el ajuste correspondiente a los ficheros que han sido resueltos. Una vez tenemos todas las soluciones y el generador ha realizado el ajuste para cada uno de ellos, hemos completado una iteración.

Cuando acaba una iteración comienza a repetirse dicho proceso, el cuál vuelve a iniciarse cuando el generador crea nuevas fichas de problemas. Este proceso se repite hasta alcanzar las iteraciones inicialmente programadas.

Debemos recordar que este proceso puede ser monitorizado por los dos elementos anteriormente descritos, el monitor del sistema y el visualizador de soluciones.

En la Figura 20 podemos ver el flujo de funcionamiento del sistema.

Las líneas azules indican el recorrido de las fichas de problemas y las líneas naranjas el recorrido de las fichas solución

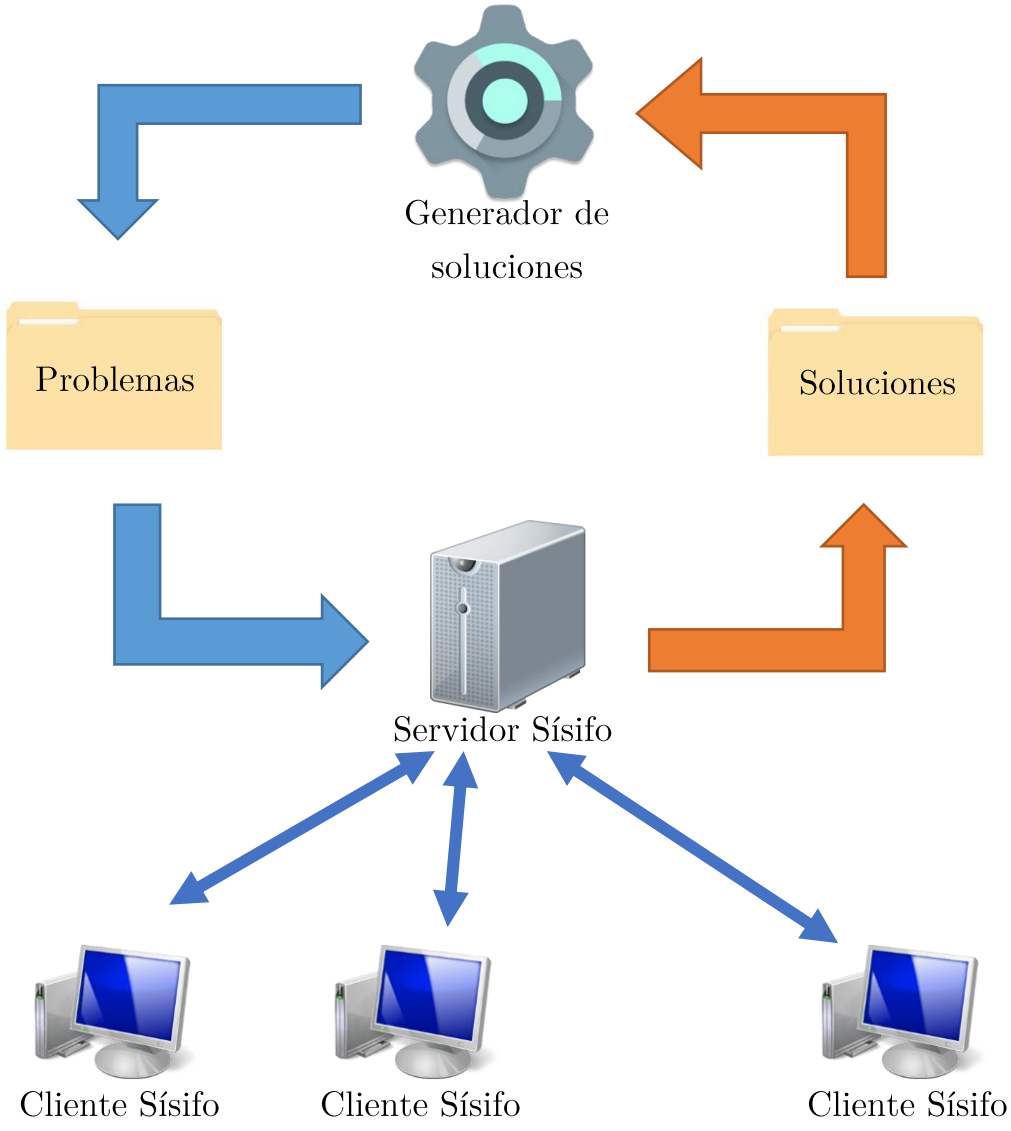


Figura 20. Flujo de funcionamiento del sistema

Capitulo 5:RESULTADOS

En este apartado se va a presentar el mejor resultado obtenido hasta la fecha utilizando todo lo descrito en la presente memoria. Además se presentará la configuración utilizada del algoritmo MOPSO y la configuración inicial de la red.

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

Parámetros	**Valores**
ITMAX	500
Partículas	100
Nrepositorio	200
Tasa_mut	0.5
Dist_mut	20
Aleat_elec_global	0.1

Tabla 2. Configuración del GS

Se ha elegido un número máximo de 500 iteraciones porque tras varias pruebas se demuestra que con este número siempre se obtienen soluciones con un RMSE menor que diez en cada uno de los objetivos.

El número de partículas es 100 porque tras las pruebas realizadas se asegura que este número es suficiente para obtener la suficiente variedad en los parámetros de los problemas generados, y así evitar una convergencia prematura del algoritmo y poder obtener el RMSE requerido en cada objetivo.

Para el resto de parámetros se han elegido los valores típicos en cada uno de ellos y no nos hemos visto obligados a modificarlos en ninguno de los casos ya que han funcionado correctamente.

La red del modelo se ha construido en base a los siguientes parámetros:

Parámetros	**Valores**
Número de nodos	250000
Turnos	1040

Tabla 3. Configuración base de la red del modelo

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 hay que tener en cuenta que a mayor número de nodos el tiempo de cálculo en simular dicha red es también mayor. En nuestras pruebas el equilibrio se ha encontrado en 250000 ya que el ajuste de las soluciones es igual de bueno que con 300000, pero además ahorramos tiempo de cálculo de manera significativa.

El valor del número de turnos, como se ha comentado anteriormente, equivale a semanas, por lo tanto al escoger 1040 como valor estamos simulando 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 podamos ajustar los últimos 10 debidamente.

Con la configuración descrita se ha obtenido el siguiente ajuste:

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

Tabla 4. RMSE en objetivos

Hemos obtenido lo que consideramos 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.

Como se ha comentado en apartados anteriores nos ayudamos de un visualizador para ver gráficamente las soluciones obtenidas. En la Figura 21, podemos ver la representación gráfica de la mejor solución.

```
                          0.03                                0.103644
                          0.48                                0.474265
Data % infected ->        0.32    Model % infected ->         0.275758
                          0.06                                0.0235278
                          0.11                                0.122805

Error objetivo 1 ->     8.38863
Error objetivo 2 ->     0.0422087
```

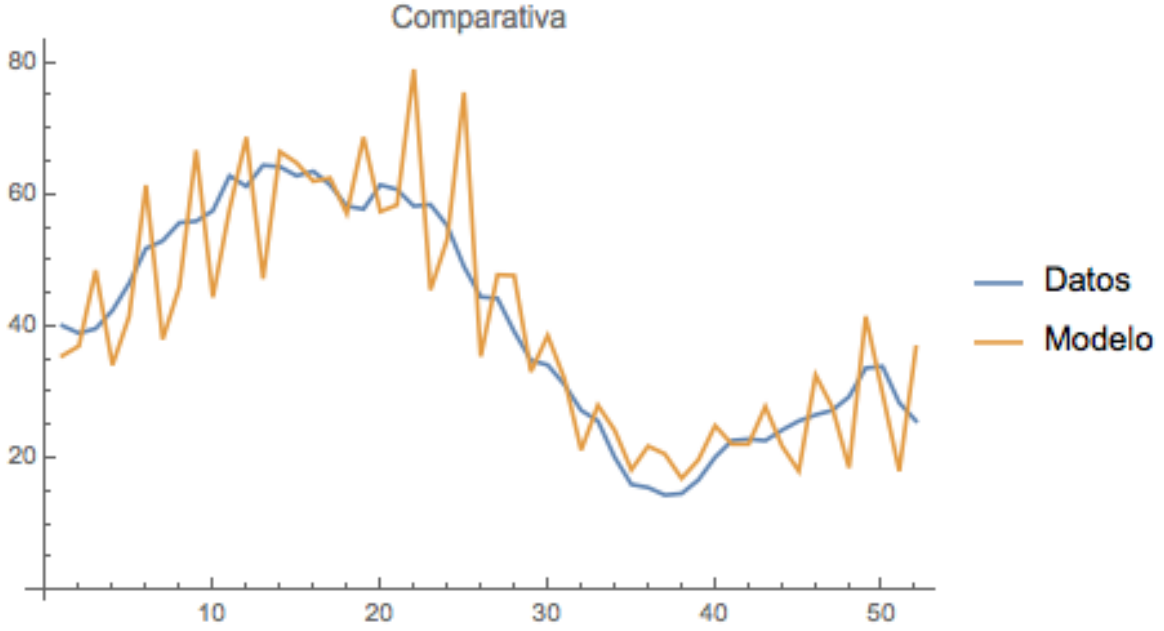


Figura 21. Mejor solución obtenida

En cuanto a tiempo de cálculo haremos una comparativa del empleado usando MOPSO y computación distribuida frente a una estimación realizada para otros métodos más tradicionales, como la búsqueda exhaustiva de los parámetros.

El modelo de varicela tarda de media en resolver un problema 15min; calculando que se necesiten 120000 problemas para obtener al menos una solución igual de buena que la obtenida por MOPSO, hace un total de 1250 días para un sistema de búsqueda exhaustiva.

Si a la búsqueda exhaustiva le añadimos computación distribuida, en nuestro caso Sísifo donde tenemos disponibles 15 clientes: 1250 días / 15 clientes = 83.3 días.

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

Búsqueda exhaustiva	**Búsqueda exhaustiva y computación distribuida**	**MOPSO y computación distribuida**
1250 días	83.3 días	12 días

Tabla 5. Tiempos de cálculo

Como se puede observar, el introducir un entorno de cálculo distribuido en un método clásico de búsqueda mejora enormemente el tiempo de cálculo invertido, pero en ocasiones, como es nuestro caso, deja de 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, obteniendo un tiempo de cálculo abordable en un corto periodo de tiempo.

En la Figura 22 podemos ver gráficamente que la cantidad de problemas que necesita resolver MOPSO es muchísimo menor a la que necesita un método de búsqueda clásica.

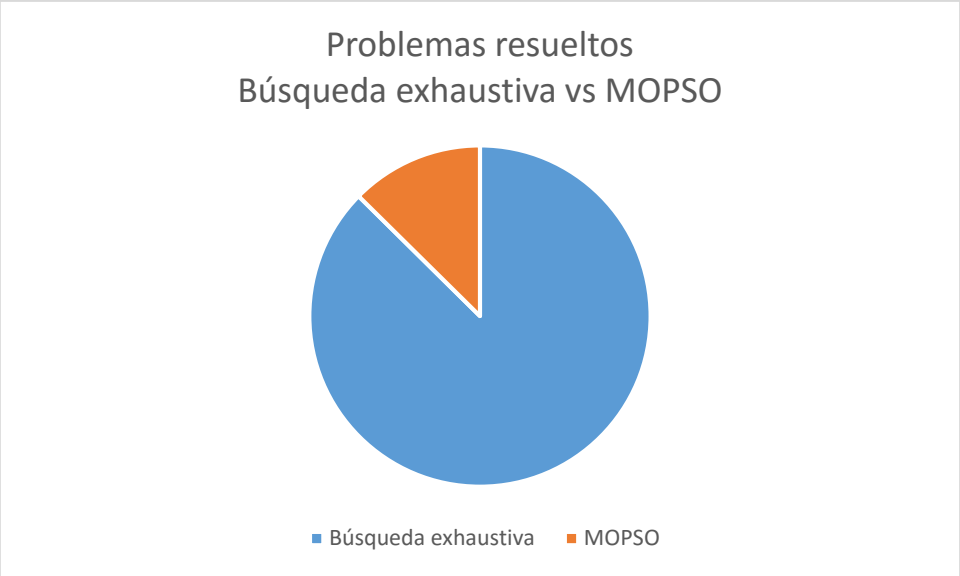


Figura 22. Problemas resueltos

Además se ha podido comprobar, tras variar pruebas, que podemos reducir el número de iteraciones ya que se obtiene al menos una solución equivalente a la presentada entre las iteraciones 173 y 231.



Capitulo 6: CONCLUSIONES Y TRABAJOS FUTUROS

En el presente trabajo hemos abordado la problemática de ajustar los parámetros de un modelo de redes aleatorias para estudiar el contagio de una enfermedad infectocontagiosa, muy común en ciertos grupos de edad, como es la varicela.

El ajuste de un modelo como el de varicela es muy costoso computacionalmente -más de tres años si usamos metodologías clásicas como búsqueda exhaustiva-, lo que nos ha llevado a usar computación distribuida como primera mejora. Aun así, los tiempos son elevados (más de 83 días). Esto hace plantearnos una alternativa que permita reducir en gran medida el tiempo de cálculo y hacerlo abordable a corto plazo, sustituyendo el método de búsqueda exhaustiva por un algoritmo metaheurístico como MOPSO. Gracias a la incorporación de este tipo de algoritmos hemos conseguido un buen ajuste (RMSE menor a 10 en cada uno de los parámetros analizados) 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. Hacer uso de esta arquitectura nos ha permitido introducir almacenamiento en la nube, lo que permite que sistemas completamente desconocidos entre sí colaboren para un mismo fin sin que los componentes de cada uno sepan de la existencia de los elementos que conforman el sistema completo. Esto proporciona al sistema una alta cohesión y a su vez un muy bajo acoplamiento, pues los elementos pueden sustituirse sin que sea agresivo para la propia arquitectura.

Además hemos podido implementar e introducir diversos elementos: Un monitor de soluciones que ayuda al investigador a seguir la traza tanto a los cálculos del modelo como al ajuste en tiempo real, y un visualizador de soluciones que interpreta las soluciones obtenidas y permite mostrar gráficamente el ajuste obtenido para una determinada solución.

6.1 Trabajos futuros

- Con respecto al modelo de varicela, realizar diversas simulaciones para determinar las posible estrategias de vacunación;
- Realizar un estudio del 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 validarla por un lado y, por otro, hacerla completamente general, lo que abriría un amplio abanico de posibles aplicaciones.

BIBLIOGRAFÍA

[1] Acedo, L., Moraño, J. A., & Díez-Domingo, J. (2010). Cost analysis of a vaccination strategy for respiratory syncytial virus (RSV) in a network model.Mathematical and Computer Modelling, 52(7), 1016-1022.

[2] Acedo, L., Moraño, J. A., Villanueva, R. J., Villanueva-Oller, J., & Díez-Domingo, J. (2011). Using random networks to study the dynamics of respiratory syncytial virus (RSV) in the Spanish region of Valencia.Mathematical and Computer Modelling, 54(7), 1650-1654.

[3] Acedo, L., Morano, J. A., Villanueva, R. J., Villanueva-Oller, J., & Dıez-Domingo, J. (2010). Seasonal Respiratory Syncytial Virus Epidemic in a Random Social Network. MODELLING FOR ADDICTIVE BEHAVIOUR, MEDICINE AND ENGINEERING 2010, 1.

[4] Acedo, L., Villanueva-Oller, J., Morano, J. A., & Villanueva, R. J. (2013, January). The Neurona@ Home project: simulating a large-scale cellular automata brain in a distributed computing environment. In PHYSICS, COMPUTATION, AND THE MIND-ADVANCES AND CHALLENGES AT INTERFACES: Proceedings of the 12th Granada Seminar on Computational and Statistical Physics (Vol. 1510, No. 1, pp. 267-270). AIP Publishing.

[5] 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.

[6] 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.

[7] Álvarez, E., Donado-Campos, J., & Morilla, F. (2015). New coronavirus outbreak. Lessons learned from the severe acute respiratory syndrome epidemic. Epidemiology and infection, 143(13), 2882-2893.

[8]Barabási, A. L., & Albert, R. (1999). Emergence of scaling in random networks.science, 286(5439), 509-512.

[9] Bar-Yam, Y. (1997). Dynamics of complex systems(Vol. 213). Reading, MA: Addison-Wesley.

[10] 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 Symposium Workshops & PhD Forum (IPDPSW), 2012 IEEE 26th International (pp. 507-518). IEEE.

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

[12] 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.

[13] Dos Santos, R. Z. (1999). Immune responses: getting close to experimental results with cellular automata models. Ann. Rev. Comp. Phys, 6, 159.

[14] Eberhart, R. C., & Shi, Y. (2000). Comparing inertia weights and constriction factors in particle swarm optimization. In Evolutionary Computation, 2000. Proceedings of the 2000 Congress on (Vol. 1, pp. 84-88). IEEE.

[15] Edelstein-Keshet, L. (1988). Mathematical models in biology (Vol. 46). Siam.

[16] Gil-Niala, M.A., Morilla, F., Dormido, S., & Donado, J. (2010). Entorno de simulación de enfermedades transmisibles. XXVIII Reunión científica de la sociedad española de epidemiología.

[17] Hershberg, U., Louzoun, Y., Atlan, H., & Solomon, S. (2001). HIV time hierarchy: winning the war while, loosing all the battles. Physica A: Statistical Mechanics and its Applications, 289(1), 178-190.

[18] Hethcote, H. W. (2000). The mathematics of infectious diseases. SIAM review,42(4), 599-653.

[19] Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. InNeural Networks, 1995. Proceedings., IEEE International Conference on (Vol. 4, pp. 1942-1948). IEEE.

[20] Kennedy, J., Kennedy, J. F., Eberhart, R. C., & Shi, Y. (2001). Swarm intelligence. Morgan Kaufmann.

[21] Kermack, W. O., & McKendrick, A. G. (1927, August). A contribution to the mathematical theory of epidemics. In Proceedings of the Royal Society of London A: mathematical, physical and engineering sciences (Vol. 115, No. 772, pp. 700-721). The Royal Society.

[22] Leij, M. J. (2006). The economics of networks: Theory and empirics. Erasmus School of Economics (ESE).

[23] 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.

[24] Murray, J. D. (2002). Mathematical biology I: an introduction, Vol. 17 of interdisciplinary applied mathematics.

[25] Podestá, L. (2011). Exantemas. Arnoldo Quezada. Pediatría Ambulatoria.

[26] Proulx, S. R., Promislow, D. E., & Phillips, P. C. (2005). Network thinking in ecology and evolution. Trends in Ecology & Evolution, 20(6), 345-353.

[27] Sánchez, V., Santos, M., & Villanueva, J. (2014). Estudio de la dinámica de transmisión del meningococo C mediante redes aleatorias.

[28] Santonja, F. J., Lombana, I. C., Rubio, M., Sánchez, E., & Villanueva, J. (2010). A network model for the short-term prediction of the evolution of cocaine consumption in Spain. Mathematical and Computer Modelling, 52(7), 1023-1029.

[29] Shi, Y., & Eberhart, R. (1998, May). A modified particle swarm optimizer. InEvolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on (pp. 69-73). IEEE.

[30] Traud, A. L., Mucha, P. J., & Porter, M. A. (2012). Social structure of Facebook networks. Physica A: Statistical Mechanics and its Applications,391(16), 4165-4180.

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

[32] Watts, D. J. (1999). Small worlds: the dynamics of networks between order and randomness. Princeton university press.

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

ABREVIATURAS

CPU	Central Processing Unit
ER	Erdős y Rényi
GS	Generador de soluciones
JSP	JavaServer Pages
MOPSO	Multi Objective Particle Swarm Optimization
RMSE	Root-Mean-Square Error
PC	Personal Computer
PSO	Particle Swarm Optimization
VRS	Virus Respiratorio Sincitial
VZV	Virus Varicela-Zóster