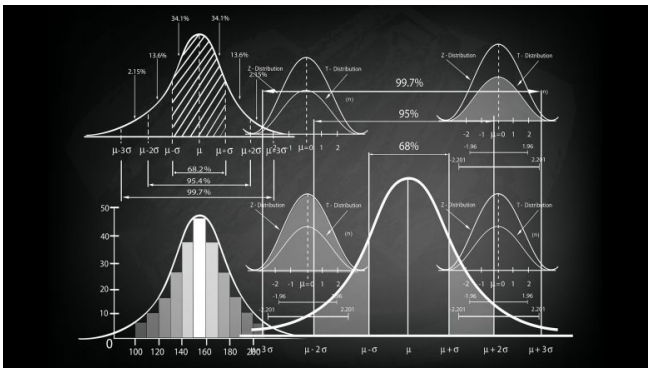

Métodos de Simulación.

Memoria Enunciado 7.

Ejercicio 3: Aplicación de simulación: Juego de cifras. Recocido simulado.

Máster Universitario en Inteligencia Artificial

Universidad Politécnica de Madrid

Javier Cardeñosa Alabau.

Adrián Michelena Sanz.

Alberto Miño Calero.

Índice

1. Descripción del problema

El problema a resolver parte de una secuencia de 8 números enteros: 75, 3, 1, 4, 50, 6, 12, 8; y un valor objetivo: 852. Se facilita también una lista de operadores aritméticos: +, -, *, /.

1.1 Objetivo del problema.

El objetivo del problema es obtener la secuencia de operadores aritméticos que, respetando el orden de los números enteros dados, se aproxime lo máximo posible a un valor objetivo. Es decir, el objetivo es que se cumpla una igualdad matemática.

2. Resolución del problema

Para resolver este problema haremos uso de la metaheurística conocida como "recocido (enfriamiento) simulado". Ésta se introduce para evitar el inconveniente de otras heurísticas de quedarse atrapado en óptimos locales. Para ello, esta metaheurística ofrece la posibilidad de moverse a soluciones peores con cierta probabilidad, basada en una temperatura, de forma similar a como funcionan ciertos procesos en la industria siderúrgica. Esto permite explorar mejor el espacio de soluciones, haciendo más diversificada la búsqueda para evitar esos mínimos locales. Una vez la metaheurística comienza la búsqueda, la temperatura fijada como inicial comenzará a decrecer cada cierto número de pasos, haciendo la búsqueda más selectiva a la hora de aceptar nuevas soluciones. Finalmente, si no se cumple antes ninguna condición de parada, la temperatura será tan reducida que sólo se aceptarán soluciones mejores a la actual.

El recocido simulado no escoge la mejor solución de un entorno, sino que escoge una al azar y en el caso de que sea mejor a la solución actual, se modifica. Es decir, que trabaja con única solución, que puede ir cambiando al largo del proceso de búsqueda, y se almacena también la mejor (óptima) encontrada. En caso de que las soluciones alcanzadas aleatoriamente sean peores, habría dos alternativas:

- La solución nueva se convierte en solución actual con una probabilidad $p(i)$ en función de una temperatura T.
- La solución anterior se mantiene como solución actual con una probabilidad complementaria de $1 - p(i)$.

Para dictar si una solución es mejor que otra, debemos definir una función fitness que nos indique numéricamente si se mejora o no. En nuestro caso, nuestra función fitness estaría definida de la siguiente forma:

$$f = |valor\ objetivo - solución\ obtenida|$$

El fitness obtenido nos dirá cuán cerca ha estado la solución actual de acercar el óptimo. Se ha usado el valor absoluto ya que no importa que la distancia sea positiva o negativa. Si esta función devuelve un 0, significará que hemos alcanzado un óptimo global, que no se puede asegurar que sea único.

2.1 Algoritmo

El algoritmo de Recocido simulado sigue la siguiente estructura:

1. Inicialización: Se escoge una solución inicial $x_1 \in X$. Hacemos $x^* = x_1, f^* = f(x_1), i = 0$, y escogemos $T_1 > 0$.
   Donde x^* es la solución óptima y T_1 la temperatura inicial.

2. Generamos aleatoriamente $y \in E(x_i)$.

   a. Si $f(y) < f(x_i)$, hacemos $x_{i+1} = y$, si $f(y) < f^*$, hacemos $x^* = y, f^* = f(y)$.

   b. En caso contrario, $f(y) \geq f$, generamos un valor $u \sim U(0,1)$.

      i. Si $p(i) = e^{-\left(\frac{f(y)-f(x_i)}{T_i}\right)} > u$ *(Boltzman),* hacemos $x_{i+1} = y$.
      ii. En caso contrario, $p(i) \leq u$, hacemos $x_{i+1} = x_i$.

   c. Actualizamos T_i.

3. Hasta satisfacer el criterio de parada

3. Diseño del algoritmo

Para inicializar este algoritmo necesitamos fijar una serie de parámetros. Estos van a determinar la convergencia de nuestro algoritmo.

3.1 Codificación de las soluciones

En este problema, se tienen dos componentes relaciones: una lista de valores numéricos, y una lista de operadores. Según la definición del problema, los valores numéricos permanecen inalterables, y lo que se busca es la lista de operadores que, aplicados en orden a dicho valores, se acerque más a una igualdad matemática cierta. Por ello, la solución vendrá dada únicamente por la lista de operadores. Dado que un operador se aplica entre dos valores numéricos, y que se deben operar la totalidad de ellos, la lista de operadores tendrá un tamaño igual a la de valores menos 1.

3.2 Definición del entorno

El primer elemento a diseñar para resolver el problema es el entorno en el que el algoritmo va a buscar y evaluar las soluciones. Sobre la evaluación, ya se ha comentado en el apartado sobre la resolución teórica del problema, que consiste en calcular el valor dada la lista de número y la de operadores. Teniendo como valores una lista de números y como solución a evaluar una lista de operadores, la función que calcula los pasos aleatorios durante las iteraciones del algoritmo funciona de la siguiente forma:

1. Se escoge de manera aleatoria una posición de la lista que representa la solución, en donde se almacenan los operadores.
2. Se elimina el operador seleccionado de la lista de operadores posibles, para evitar que se vuelva a introducir el mismo.
3. Luego se escoge aleatoriamente un operador de todos los posibles definidos en el problema.
4. Se sustituye el operador de la posición elegida de la solución por el escogido aleatoriamente de la lista de operadores.

De esta forma, se asegura que cada paso esté dentro de un entorno cercano, puesto que solo se modifica un elemento de la solución, un operador. Además, se asegura que siempre se haga un cambio efectivo y que no se pueda volver a introducir el mismo que había previamente en la posición a modificar.

3.3 Cálculo del fitness

Se hace tal y como se ha comentado en la sección anterior: para calcular el valor del fitness de una solución, se aplican los operadores a la lista de valores. A partir de este valor, se calcula la distancia al valor objetivo en valor absoluto, y ese resultado será el fitness de la solución. En el caso de este problema, se conoce cual será el óptimo global: cuando la distancia sea 0. Por tanto, se puede considerar esto como una condición de parada.

3.4 Temperatura Inicial

Para la temperatura inicial debemos tomar un valor tal que la probabilidad de moverse a soluciones peores durante el número de pasos L inicial sea siempre, como mínimo, 0,9. El motivo es que, si escogemos un valor muy cercano a 0, no se posibilita el movimiento a soluciones peores y eso hace que el algoritmo tienda a caer en óptimos locales. Esto se hace mediante la ejecución de pruebas piloto.

En el caso del problema a resolver, existe un inconveniente, y es que dependiendo del paso aleatorio que se dé cuando se hace cada paso aleatorio en cada iteración, puede existir una distancia, en términos de calidad de la solución o fitness, muy grade. Estos saltos tan grandes están relacionados con el operador de multiplicación y, en menor medida, con el de división. Esto provoca que puedan existir diferencias muy elevadas entre el fitness de la solución actual en cada paso, y

la nueva que se calcula aleatoriamente. Debido a estas diferencias tan elevadas, la determinación de la temperatura inicial da lugar a temperaturas muy elevadas.

La temperatura inicial que hemos usado ha sido: 10.000.000. Con esta temperatura hemos observado que las probabilidades de moverse a una solución peor durante las L primeras iteraciones eran siempre mayores a 0,9.

3.5 Actualización de la temperatura

La temperatura irá actualizándose cada L iteraciones, de forma que irá decreciendo, multiplicándose por $0 < \alpha < 1$. Si el decrecimiento es muy lento, el algoritmo será más lento. Por el contrario, si es demasiado rápido, cabe la posibilidad de quedar atrapado en un óptimo local. Un valor típico es $\alpha = 0{,}95$. *(Hajek, 1988).*

El factor de decrecimiento que hemos usado en la simulación de nuestro problema ha sido de $\alpha = 0{,}95$. El valor de L que hemos usado es de $L = 50$. Por tanto, cada 50 iteraciones, la temperatura será multiplicada por $\alpha = 0{,}95$.

3.6 Criterio de parada

Para evitar que el algoritmo se haga muy largo, es necesario fijar criterios de parada. Generalmente se usan los siguientes:

- Parar si f^*no ha mejorado al menos $\varepsilon_1\%$, tras k_1 iteraciones de L pasos.
- Parar si el número de transiciones aceptadas es menor que el $\varepsilon_2\%$ de L tras k_2 iteraciones de L pasos.

Para la resolución de nuestro problema, se ha trabajado con la primera de estas dos condiciones de parada, teniendo también en cuenta como condiciones de parada si se alcanza un óptimo global o se alcanza un límite de iteraciones.

Los valores de ε_1 y k_1 han sido los siguientes:

- $\varepsilon_1 = 0{,}000001$
- $k_1 = 200$

El número máximo de iteraciones que se permiten es de 100.000.

3.7 Otros parámetros del algoritmo

Los valores utilizados han sido: [75, 3, 1, 4, 50, 6, 12, 8]. El valor objetivo a alcanzar tras las diferentes operaciones es: 852.

Para inicializar el algoritmo partimos de una solución inicial cuya secuencia de operadores aritméticos es la siguiente: [*, -, *, /, +, -, +]

4 Resultados

4.1 Problema Simple

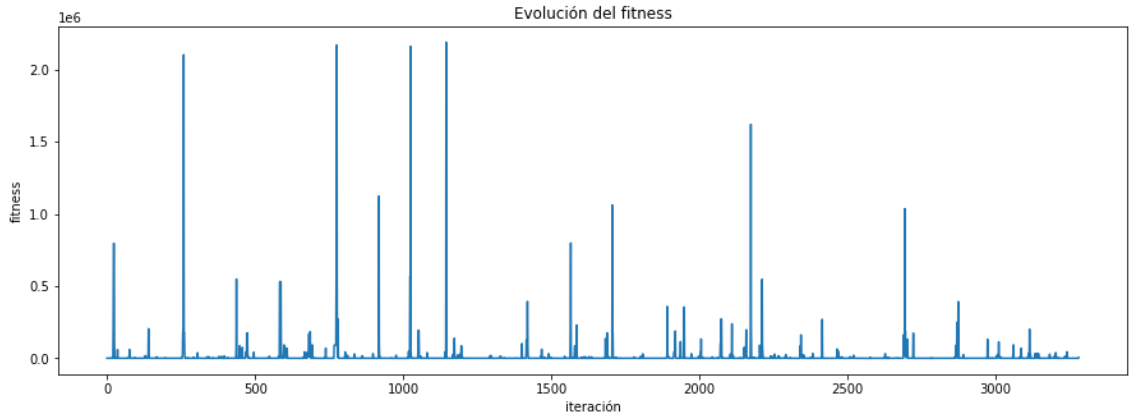

Figura 1 Evolución del fitness en cada iteración

En la Figura 1 se muestra cómo evoluciona el fitness durante la ejecución del algoritmo. Al alcanzar el óptimo de forma rápida, tanto en tiempo como en número de iteraciones, el aspecto de la gráfica no representa el comportamiento general del recocido simulado, y no se puede apreciar cómo, a medida que el algoritmo avanza, la temperatura decrece y se aceptan menos soluciones peores, provocando una intensificación de la búsqueda y disminuyendo la exploración.

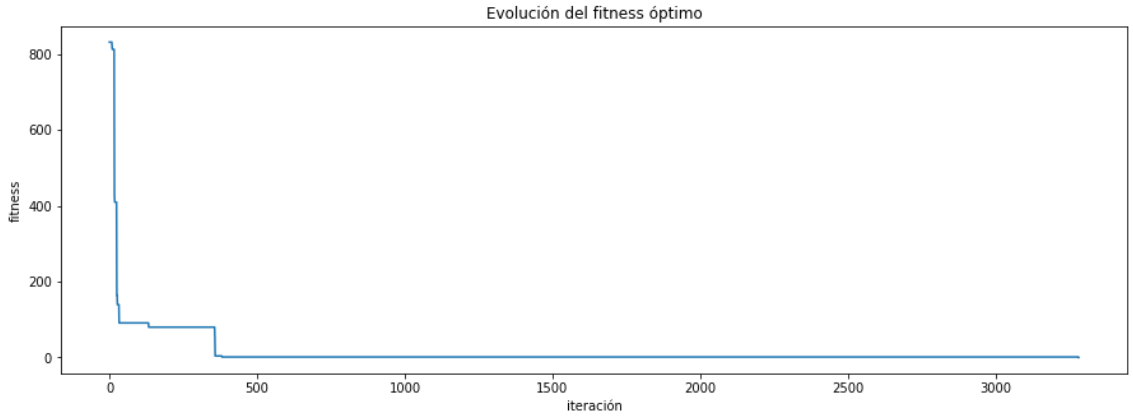

Figura 2 Evolución del fitness óptimo

En la Figura 2 se contempla la evolución de la solución óptima encontrada. En este caso, sí que se aprecia como al principio, durante las primeras 500 iteraciones, se encuentran varias soluciones óptimas que, además están cercanas al óptimo global. Luego, el algoritmo entra en una etapa en donde necesita un mayor número de pasos hasta encontrar el óptimo global.

Podría considerarse que, dado que ha encontrado un óptimo que puede considerarse bueno, no tiene sentido dejar que siga funcionando, y que por tanto se

debería endurecer las condiciones de parada. Sin embargo, lo cierto es que el tiempo de ejecución es realmente bajo, y por eso se ha considerado más adecuado utilizar los parámetros de condición de parada tal y como se han fijado.

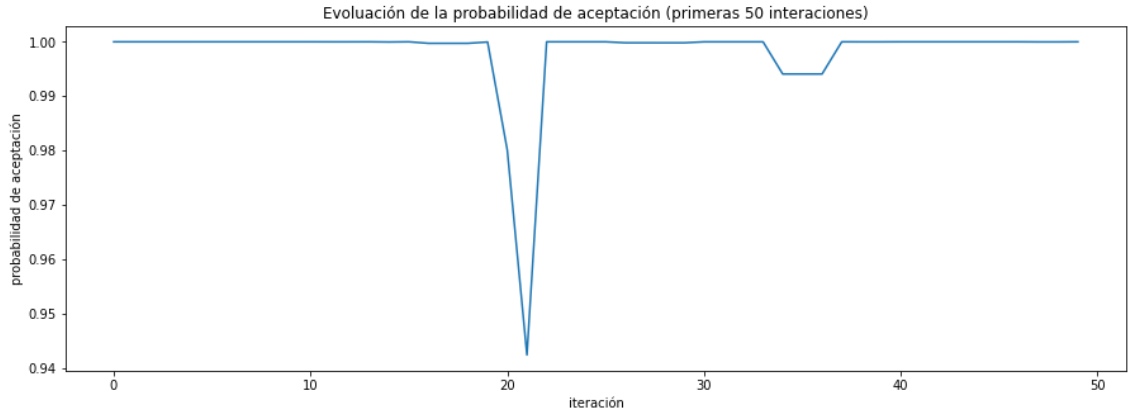

Figura 3 Evolución de la probabilidad de aceptación en las 50 primeras iteraciones

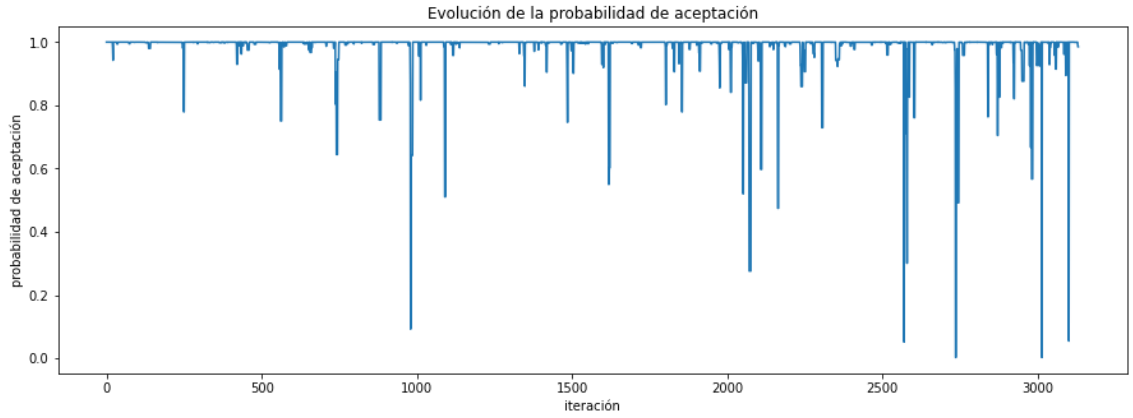

Figura 4 Evolución de aceptación en todas las iteraciones

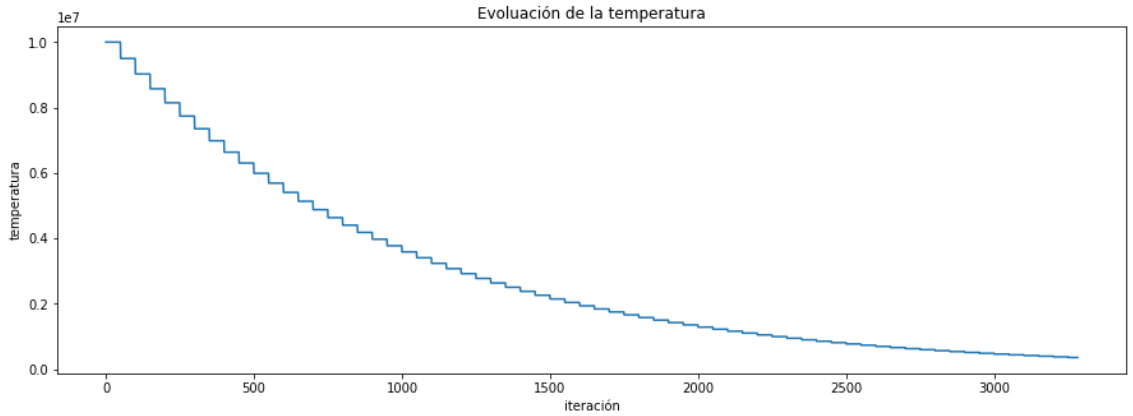

Figura 5 Evolución de la temperatura

Para el ejercicio propuesto, la metaheurística encuentra la solución óptima muy rápido. En este caso, encuentra el óptimo en la iteración 3282 en un tiempo de ejecución 0,04s. Para comprobar que no se trataba de un caso aislado, realizamos 100 iteraciones y vimos que el 92% de las iteraciones para ya que encontraba el óptimo. El tiempo de ejecución 4,57s.

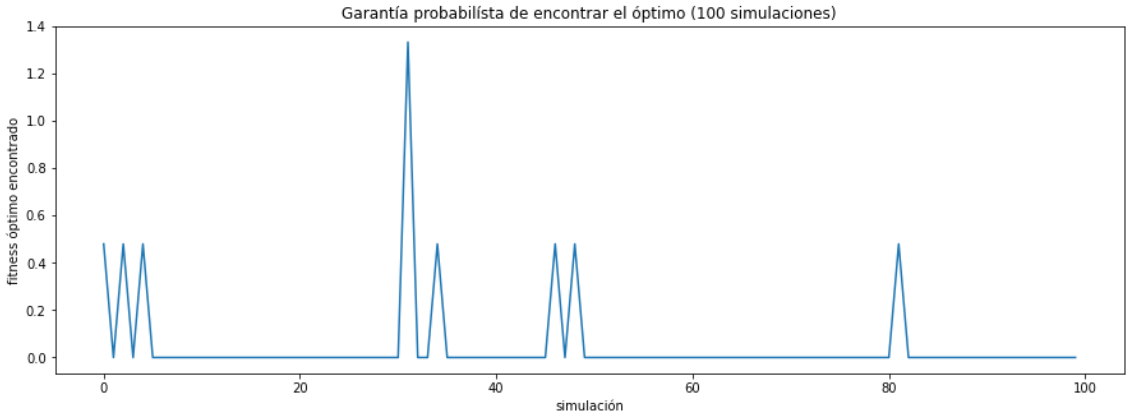

Figura 6 Garantía probabilística de encontrar el valor óptimo en 100 iteraciones

4.2 Problema Modificado

Como hemos observado que el problema anterior era demasiado simple, hemos decidido complicar el problema aumentando el número de valores iniciales a 15. El valor objetivo los hemos mantenido en 852.

Otra forma de complicar el problema es añadir otros operadores, como el operador potencia, sin embargo se descartó porque provocaba un desbordamiento e interrumpía la ejecución.

Los parámetros usados en este caso han sido:

- Temperatura inicial: 50.000.000.000.
- $\alpha = 0{,}90$
- $L = 100$
- $k_1 = 200$
- $\varepsilon_1 = 0{,}00001$
- Solución inicial: [+, *, /, -, +, +, /, /, -, +, *, +, -, +]
- Valores iniciales: [72, 1, 61, 15, 7, 19, 4, 5, 1, 2, 28, 12, 17, 13, 44]

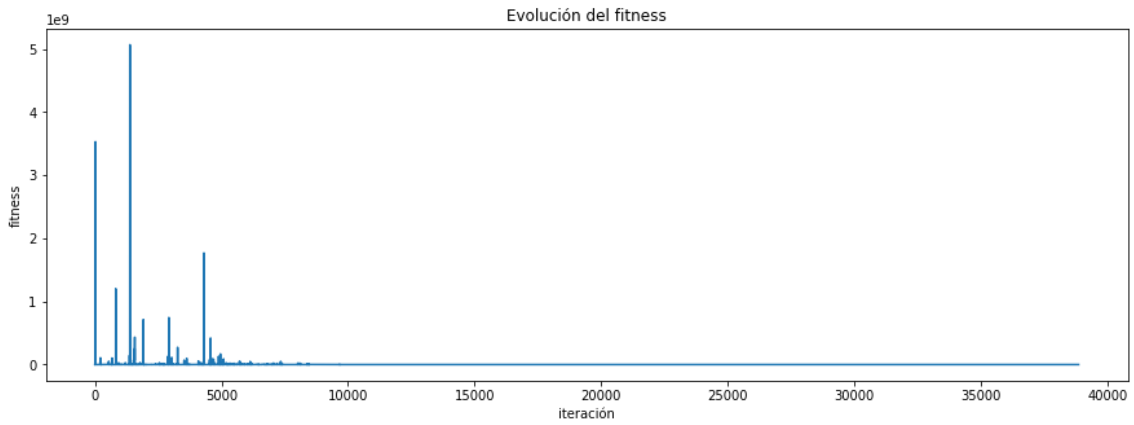

Figura 7. Evolución del fitness.

En la Figura 7. Evolución del fitness. Figura 7, observamos la evolución del fitness para el problema complejo. La escala de la imagen es demasiado alta (10^9). No ha sido posible cambiar la escala, aunque cabe mencionar que, aunque parezca que los valores son 0 a partir de la iteración 10.000, pueden ser valores muy superiores. En la Figura 8 se ve la evolución de las soluciones óptimas que encuentra el algoritmo.

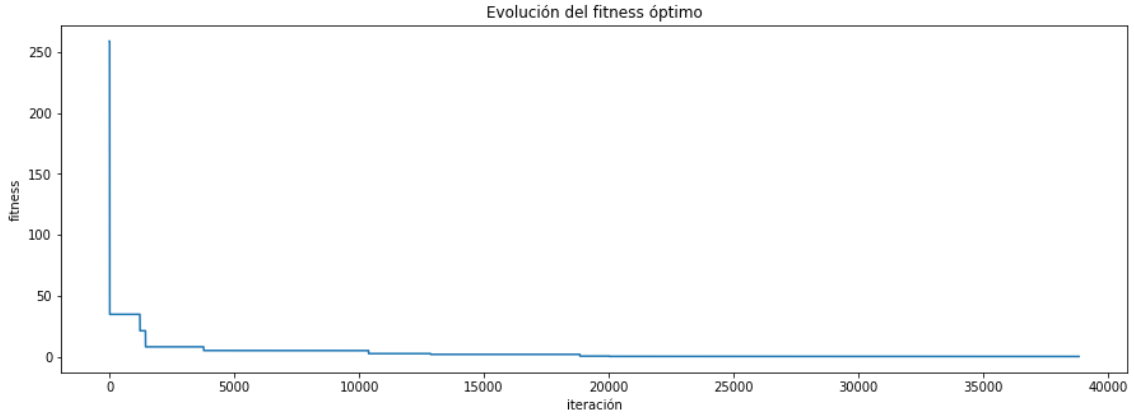

Figura 8. Evolución del fitness óptimo

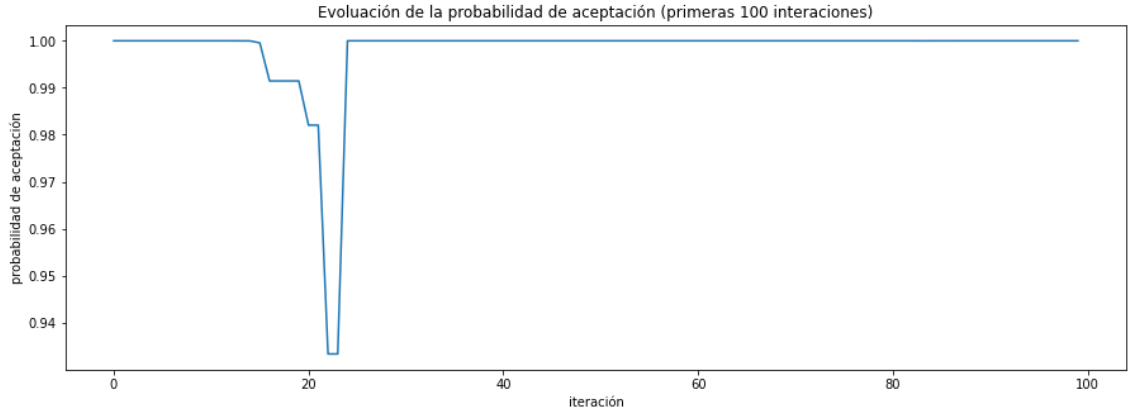

Figura 9. Evolución de la probabilidad de aceptación en las 100 primeras iteraciones

En la Figura 9, podemos observar que la probabilidad de moverse a soluciones peores es siempre mayor a 0.9 durante las primeras 100 iteraciones, por lo que la temperatura inicial se puede considerar adecuada.

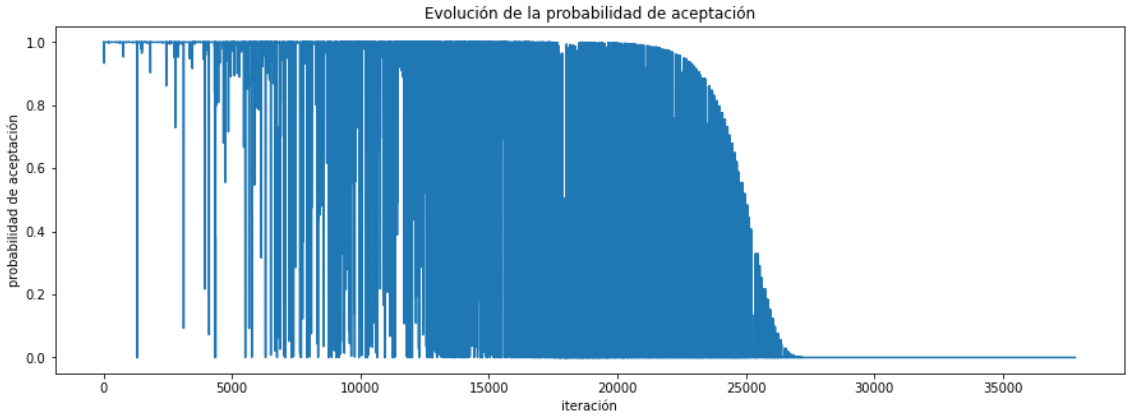

Figura 10. Evolución de aceptación en todas las iteraciones.

En la Figura 10, vemos que la probabilidad de aceptación al principio es muy alta debido a que el algoritmo hace una búsqueda muy diversificada. Esta probabilidad decrece conforme avanzan las iteraciones, haciendo que sólo se permitan movimientos hacia soluciones mejores.

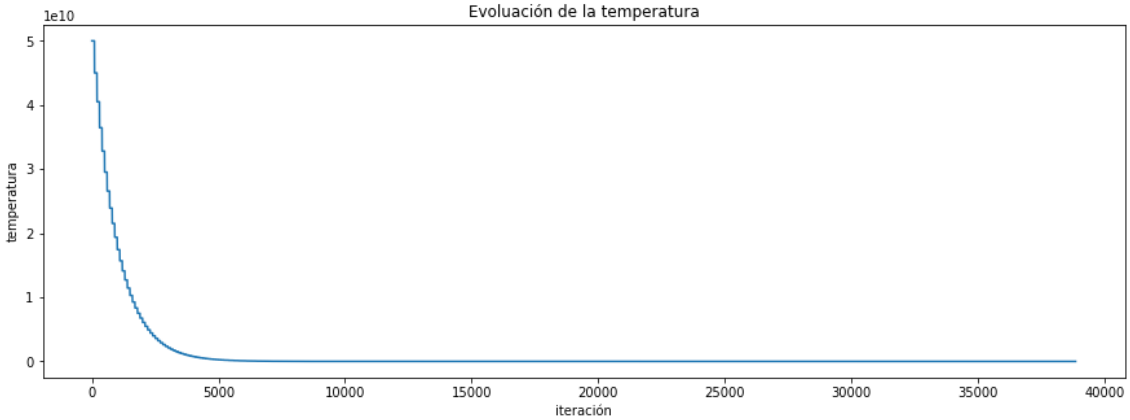

Figura 11. Evolución de la temperatura.

En la Figura 11, ocurre lo mismo que ocurría en la Figura 7 con la escala. La temperatura parece llegar a 0 en muy pocas iteraciones, pero no es así debido a que se encuentra en una de escala 10^{10}. De hecho, la temperatura en la iteración final es de $\sim 1{,}69$.

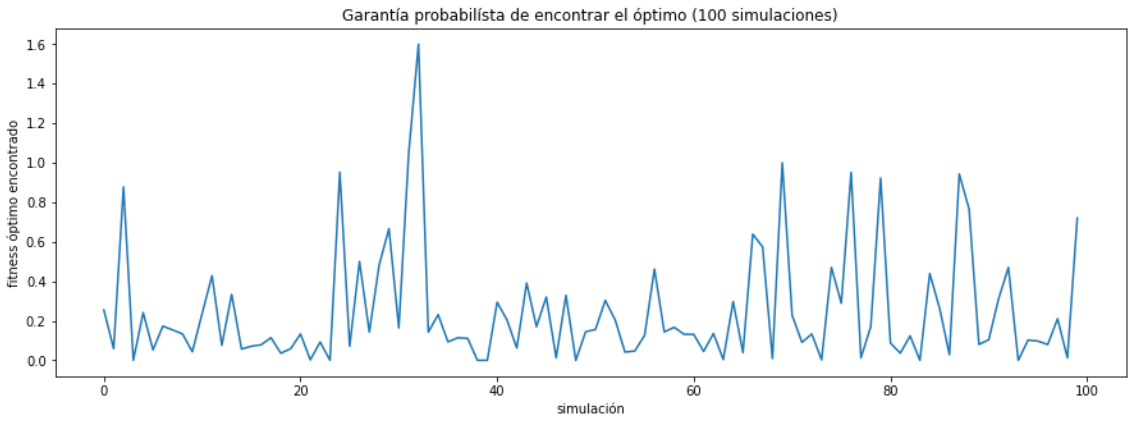

Figura 12 Garantía probabilística del problema modificado

Y para finalizar, en la Figura 12 se puede comprobar como la modificación del problema lo ha complicado, y ya no ocurre que sea capaz de alcanzar una solución óptima global en la mayoría de los casos, quedándose en estas 100 simulaciones en un 7% de éxito. Aunque también hay que añadir que las soluciones están muy cerca del óptimo, por lo que se puede decir que son buenas soluciones. Los parámetros escogidos para la ejecución son los mismos que se enumeraron en el problema modificados, que fueron, de las configuraciones que se comprobaron, los que mejor resultado dieron a la hora de resolver el problema.

5 Conclusiones

La metaheurística del recocido simulado se puede utilizar para resolver el problema que ha planteado, siendo éste bastante simple. Presenta un inconveniente relacionado con el operador de multiplicación, como se ha comentado, y es que provoca saltos muy grandes en el fitness de las soluciones, lo cual hace que la temperatura inicial deba ser lo suficientemente elevada como para compensar este valor para que la probabilidad de moverse a soluciones peores sea mayor de 0,9. En el caso de este problema inicial planteado, se encuentra la solución en el 92% de los casos.

En cambio, en el caso del problema modificado, la ampliación del espacio de búsqueda hace al algoritmo mucho más difícil encontrar soluciones óptimas, consiguiéndolo solo en el 7% de las 100 simulaciones. Se puede decir que este problema es más adecuado para revisar las gráficas e ilustrar el comportamiento del algoritmo, especialmente la referente a la evolución de las probabilidades de movimientos a soluciones peores. Sin embargo, esos saltos del valor de fitness que se comentaron se acentúan aún más, con el consiguiente aumento de la temperatura inicial. Y es este valor tan grande de la temperatura junto con los saltos tan grandes en el fitness lo que provoca que las gráficas de evolución de ambas no sean una representación tan adecuada para representar la evolución de estos valores a lo largo de las ejecuciones. Por último, hay que añadir que, en el caso de este problema más complejo, la ejecución del algoritmo suele finalizar porque las mejoras no

superan el porcentaje fijado en las condiciones de parada, y una relajación de las mismas aumenta en gran medida el tiempo de ejecución sin que se mejoren significativamente los resultados.