Heuristic design and fundamentals of the Simulated Annealing

Kathryn A. Dowsland (1), Belarmino Adenso Díaz (2)

(1) ASAP Research Group School of Computer Science and Information Technology
University of Nottingham
Nottingham NG8 2BB, United Kingdom

(2) Escuela Politécnica Superior de Ingeniería de Gijón
Universidad de Oviedo
Campus de Viesques E-33204 Gijón, Spain

e-mail: kad@cs.nott.ac.uk, adenso@epsig.uniovi.es

Simulated Annealing is one of the most classic metaheuristics. Its simplicity and good results over a wide range of problems have made it a highly popular tool, with hundreds of applications in a great variety of fields. This article is a review of the fundaments underlying the method, and analyses the different decisions that the analyst has to take in the design phase as well as summarising the main recommendations that may be made when defining new algorithms. A review is likewise made of the most relevant implementations published in relation to this methodology.

Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial. No.19 (2003),pp. 93-102
ISSN: 1137-3601. © AEPIA (http://www.aepia.org/revista).

Diseño de Heurísticas y Fundamentos del Recocido Simulado

Kathryn A. Dowsland, Belarmino Adenso Díaz

ASAP Research Group
School of Computer Science and Information Technology
University of Nottingham
Nottingham NG8 2BB, Reino Unido
kad@cs.nott.ac.uk

Escuela Politécnica Superior de Ingeniería de Gijón
Universidad de Oviedo
Campus de Viesques
E-33204 Gijón, España
adenso@epsig.uniovi.es

Resumen

El Recocido Simulado es una de las metaheurísticas más clásicas. Su simplicidad y buenos resultados en numerosos problemas, la han convertido en una herramienta muy popular, con cientos de aplicaciones en los más variados campos. En este artículo se hace una revisión de los fundamentos del método, analizando las diferentes decisiones que el analista ha de tomar en la fase de diseño, y resumiendo las principales recomendaciones que cabe hacer a la hora de definir nuevos algoritmos. Se realiza asimismo un repaso de las implementaciones más relevantes referidas a esta metodología que han sido publicadas.

Palabras clave: Recocido Simulado, Metaheurísticas, Optimización Combinatoria

1. Introducción

Desde que Kirkpatrick *et al.* (1983) introdujeron hace veinte años el concepto de Recocido Simulado, RS, ("*Simulated Annealing*", SA, en inglés), esta metaheurística ha demostrado ser una herramienta muy exitosa para resolver una amplia gama de problemas de optimización combinatoria. El Recocido Simulado es una variante de la búsqueda local que permite movimientos ascendentes para evitar quedar atrapado prematuramente en un óptimo local. El nombre le viene de la idea en que está basado un algoritmo diseñado en los años 50 para simular el enfriamiento de material (un proceso denominado "recocido").

A menudo se dice que mientras que es muy fácil hacer que RS funcione, es difícil hacer que funcione bien. Esto es debido a que no es propiamente un algoritmo, sino una estrategia heurística que necesita de varias decisiones para que quede totalmente diseñado, las cuales tienen una gran influencia en la calidad de las soluciones generadas.

El objetivo de este tutorial es presentar una introducción a RS y repasar algunos de los factores que necesitan ser tenidos en cuenta para poder realizar una implementación exitosa para un problema particular. La siguiente sección introduce las ideas que están detrás de esta técnica y ofrece

Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial. No.20 (2001), pp. 34-52.
ISSN: 1137-3601. © AEPIA (http://www.aepia.dsic.upv.es/).

una definición formal de la estrategia del Recocido Simulado. Como se comentará, las decisiones que hay que tomar para el diseño pueden ser clasificadas en dos grupos: las genéricas y las específicas del problema, y ambas son tratadas en las dos siguientes secciones. Finalmente se presenta una selección de aplicaciones publicadas para resolver algunos problemas reales y otros problemas clásicos de optimización combinatoria, con el fin de ilustrar algunas de las variantes que se han aplicado para ofrecer buenas soluciones en esa amplia gama de tipos de problemas.

2. El Proceso de Recocido Simulado

Los algoritmos tradicionales de búsqueda local parten de una solución inicial que de modo paulatino es transformada en otras que a su vez son mejoradas al introducirles pequeñas perturbaciones o cambios (tales como cambiar el valor de una variable o intercambiar los valores que tienen dos variables). Si este cambio da lugar a una solución "mejor" que la actual, se sustituye ésta por la nueva, continuando el proceso hasta que no es posible ninguna nueva mejora. Esto significa que la búsqueda finaliza en un óptimo local, que no tiene por qué ser forzosamente el global.

Un modo de evitar este problema es permitir que algunos movimientos sean hacia soluciones peores. Pero por si la búsqueda está realmente yendo hacia una buena solución, estos movimientos "de escape" deben realizarse de un modo controlado. En el caso del RS, esto se realiza controlando la frecuencia de los movimientos de escape mediante una función de probabilidad que hará disminuir la probabilidad de esos movimientos hacia soluciones peores conforme avanza la búsqueda (y por tanto estamos previsiblemente más cerca del óptimo global).

La fundamentación de este control se basa en el trabajo de Metropolis *et al.* (1953) en el campo de la termodinámica estadística. Básicamente, Metropolis modeló el proceso de recocido mencionado anteriormente simulando los cambios energéticos en un sistema de partículas conforme decrece la temperatura, hasta que converge a un estado estable (congelado). Las leyes de la termodinámica dicen que a una temperatura t la probabilidad de un incremento energético de magnitud δE se puede aproximar por

$$P[\delta E]=\exp(-\delta E/kt) \qquad [1]$$

siendo k una constante física denominada de Boltzmann. En el algoritmo de Metropolis se genera una perturbación aleatoria en el sistema y se calculan los cambios de energía resultantes: si hay una caída energética, el cambio se acepta automáticamente; por contra, si se produce un incremento energético, el cambio será aceptado con una probabilidad dada por [1]. El proceso se repite durante un número predefinido de iteraciones en series decrecientes de temperaturas, hasta que el sistema esté "frío".

Simulación termodinámica	**Optimización combinatoria**
Estados del sistema	Soluciones factibles
Energía	Coste
Cambio de estado	Solución en el entorno
Temperatura	Parámetro de control
Estado congelado	Solución heurística

Tabla 1. Relación establecida entre los elementos de simulación termodinámica y los de optimización combinatoria

A principios de la década de los 80, publicaciones independientes de Kirkpatrick *et al.* (1983) sobre diseño de circuitos VLSI, y Cerny (1985) para el TSP, mostraron cómo este proceso podría ser aplicado a problemas de optimización, asociando conceptos clave del proceso original de simulación, con elementos de optimización combinatoria según se indica en la tabla 1.

```
Sea f(s) el coste de la solución s y sea N(s) su entorno.
Seleccionar una solución inicial s0;
Seleccionar una temperatura inicial t0 > 0;
Seleccionar una función de reducción de la temperatura α;
Seleccionar un número de iteraciones nrep;
Seleccionar un criterio_de_parada;
REPETIR
   REPETIR
      Seleccionar aleatoriamente una solución s ∈ N(s0);
      Sea δ = f(s) - f(s0);
      SI δ < 0 ENTONCES s0 = s
      SINO
         Generar aleatoriamente u ∈ U(0,1);
         SI u < exp(-δ/t) ENTONCES  s0 = s;
      FINSINO
   HASTAQUE cuenta_iteraciones = nrep
   t = α (t);
HASTAQUE criterio_de_parada = CIERTO.

La mejor solución visitada será la solución heurística dada
por el algoritmo
```

FIGURA 1. Algoritmo básico de Recocido Simulado para minimización

Por tanto, cualquier implementación de búsqueda local puede convertirse en una implementación RS al elegir elementos del entorno de modo aleatorio, aceptar automáticamente todos los movimientos hacia una mejor solución, y aceptar los movimientos a una solución peor de acuerdo con una probabilidad

dada por [1]. La constante de Boltzmann k en general no se considera, debido a que no tiene significado en los problemas de optimización. Por tanto, podemos definir un algoritmo básico de Recocido Simulado para problemas de minimización como se indica en la Figura 1.

El parámetro t es un parámetro de control denominado generalmente *temperatura*, siguiendo la analogía con el proceso de enfriamiento físico. Una solución que suponga un incremento δ en la función de coste se aceptará con probabilidad $exp(-\delta/t)$. Por tanto, si se permite que t alcance valores suficientemente pequeños, ya no habrá más movimientos a peores soluciones y la convergencia será a un óptimo local.

Varios estudios teóricos demuestran que si t decrece lo suficientemente lento, el proceso converge a la solución óptima. Sin embargo, una función de reducción de la temperatura α que garantizase esa convergencia al óptimo global, requeriría unos tiempos de cálculo prohibitivos. A pesar de todo, muchos trabajos empíricos que se han publicado usando programas de enfriamiento más rápidos demuestran que RS es una heurística eficiente para una gran variedad de problemas combinatorios.

Para poder implementar el algoritmo de la figura 1 para un problema concreto, es preciso tomar una serie de decisiones. En este artículo serán divididas en decisiones genéricas y específicas. Las genéricas se refieren principalmente a cómo controlar la temperatura (incluyendo la definición de su valor inicial t_0 y la función de decrecimiento α), el número de iteraciones *nrep* antes del decrecimiento de la temperatura, y las condiciones que nos permitirán considerar que el sistema está ya "frío". Las decisiones específicas comprenden la definición del espacio de soluciones y la estructura de entornos, la función de coste, y cómo se obtendrá la solución inicial s_0. Ambos tipos de decisiones hay que tomarlas con cuidado pues, como se comentó anteriormente, ejercen una gran influencia sobre la calidad de las soluciones.

Aunque no hay un conjunto concluyente de reglas que definan cómo tomar las mejores decisiones genéricas y específicas en un problema concreto, sí que hay algunos consejos que pueden ayudar en el proceso de elección de esos parámetros. Estos consejos son los que se tratan en las dos siguientes secciones.

3. Decisiones genéricas

Las decisiones genéricas están básicamente relacionadas con los parámetros que dirigen el programa de enfriamiento, incluyendo los valores máximo y mínimo que podrá tomar la temperatura, la velocidad a la que se reducirá y las condiciones de parada. Como antes se comentó, resultados teóricos basados en la teoría de las cadenas de Markov han demostrado que con un programa de enfriamiento infinitamente lento, el algoritmo de recocido convergería al óptimo global con probabilidad 1, según la temperatura tiende a 0. Los detalles pueden ser consultados en Van Laarhoven y Aarts (1988), y en Aarts y Korst (1989).

Sin embargo, no se puede garantizar esa convergencia en programas de enfriamiento finitos, y en ese sentido, Van Laarhoven y Aarts han demostrado que para garantizar llegar a una solución que esté a una distancia arbitrariamente próxima a la óptima, son necesarios tiempos de computación de tipo exponencial. Hajek (1988) demostró que si se usa un programa de enfriamiento dado por $t_k = \Gamma / \ln(1+k)$, siendo k el número de iteraciones y Γ la máxima profundidad necesaria (ver Figura 2) para escapar de cualquier óptimo local —no global—, está garantizada una convergencia asintótica al óptimo. Como por lo general no se conoce el valor Γ, este resultado no tiene uso práctico, pero da una idea de cómo la velocidad del programa de enfriamiento debe estar relacionada con las características del espacio de soluciones, definidas por la función de coste y la estructura de entornos elegida.

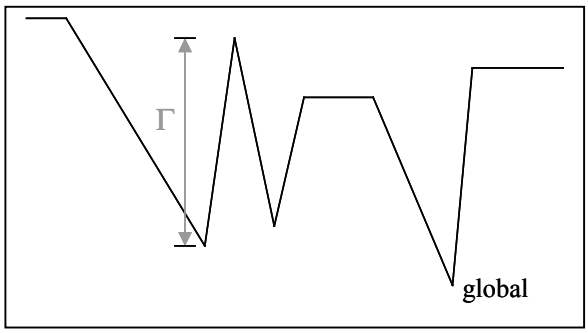

FIGURA 2. Explicación del valor Γ en el teorema de Hajek

Dado que uno de los objetivos de cualquier procedimiento robusto de búsqueda es que la calidad de la solución final sea independiente de la solución inicial de la que se parta, la temperatura inicial debería ser independiente de la solución inicial y lo suficientemente alta como para aceptar casi libremente las soluciones del entorno. En aquellos problemas que están bien estructurados, por lo general hay suficiente información como para estimar este valor. Por ejemplo, si el mayor incremento entre soluciones vecinas es conocido, sería posible calcular un valor t_0 que aceptase un movimiento con una determinada probabilidad.

Cuando este no sea posible, la razón entre movimientos aceptados y rechazados que dé lugar a un estado inicial aceptablemente volátil puede ser fijado de antemano, y entonces el sistema podría ser calentado rápidamente hasta que esa razón alcance el valor deseado. Ese sería entonces el valor elegido t_0 para la temperatura inicial a partir de la cual comenzaría el proceso de recocido.

La velocidad a la que se produce el enfriamiento es otro factor clave en el éxito de la estrategia. Viene determinado por una parte por el número de iteraciones *nrep* que se ejecutarán a cada temperatura, y por otro por la velocidad α a la que se realizará el enfriamiento. La teoría sugiere que se debería permitir que el sistema esté cerca de su estado estacionario correspondiente a la temperatura actual, antes de reducir ésta, y que además la temperatura vaya gradualmente acercándose al valor 0.

En los primeros años del RS se utilizaron diversos programas de enfriamiento, de los cuales dos son especialmente interesantes porque, por una parte, dominaron rápidamente las elecciones hechas por los investigadores que resolvieron problemas reales, y por otra, representan alternativas justamente opuestas. La primera, establecía una velocidad de enfriamiento de la temperatura de tipo geométrico, $t \rightarrow \alpha \cdot t$, con $\alpha < 1$. Las evidencias empíricas daban a entender que valores elevados de α, con valores entre 0.8 y 0.99 (correspondientes a velocidades lentas de enfriamiento) eran los que mejores resultados proporcionaban.

El número de iteraciones en cada temperatura se correspondía generalmente con el tamaño del problema, variando según descendía la temperatura. Por otro lado, es importante dedicar suficiente tiempo de búsqueda a temperaturas bajas para garantizar que se visita el óptimo local (es decir, aumentar el valor *nrep* según se reduce *t*).

Según el segundo programa de enfriamiento, propuesto por Lundy y Mees (1986), se ejecuta una sola iteración para cada temperatura, pero por el contrario la temperatura se reduce a una velocidad muy lenta según la fórmula $t \rightarrow t/(1+\beta t)$, siendo β un valor muy pequeño.

La evidencia empírica y los resultados teóricos dan a entender que los detalles concretos del programa de enfriamiento no son tan importantes como la velocidad a la que se reduce la temperatura, y que tanto el programa geométrico como el de Lundy y Mees darán resultados similares cuando el enfriamiento se produzca para el mismo rango de temperaturas y para un número total de iteraciones similares.

El valor final de *t* es también importante. Teóricamente, *t* debería reducirse hasta 0, pero en la práctica la búsqueda converge por lo general a su óptimo local final bastante antes de ese valor nulo de la temperatura. Por tanto, si la temperatura de parada se fija muy baja, gastaremos mucho tiempo de búsqueda en las fases finales, que seguramente sería mejor aprovechado en temperaturas superiores. Por el contrario, si la temperatura final se fija en un valor muy alto, es posible que la búsqueda no consiga alcanzar ningún óptimo local.

Como se cumple que $\text{Prob}(f(s_0)-f(s_{opt})<\varepsilon) > (|S|-1)\cdot\exp(-\varepsilon/t)$, siendo S el conjunto de soluciones, Lundy y Mees sugieren para llegar con probabilidad θ a una solución que esté a menos de una distancia ε del óptimo, parar en el momento en que $t \leq \varepsilon / \ln\left[{|S|-1}/{\theta}\right]$. Otros sugieren detener la búsqueda simplemente cuando se haya producido un número determinado de iteraciones sin ninguna aceptación.

Aunque los dos planes de enfriamiento son bastante sencillos de implementar, ambos requieren una labor de determinación (y por tanto de prueba y error) de los parámetros más adecuados. Por tanto, no es de extrañar que haya habido en todos estos años un gran interés por desarrollar programas de enfriamiento de modo automático. En muchas ocasiones, los fundamentos de las distintas soluciones aportadas se han basado en los propios orígenes del RC (es decir, la termodinámica estadística).

Por ejemplo, White (1984) propuso que la temperatura inicial debería ser lo suficientemente alta como para garantizar que la energía media (es decir, el coste) de las soluciones visitadas sea igual a la energía media de todo el sistema. Otros sugirieron sistemas realmente adaptativos que usan el concepto de equilibrio térmico para cada temperatura. Por ejemplo Huang *et al*. (1986) ajustan la temperatura según la ecuación $t \rightarrow t\exp(\frac{-\lambda t}{\sigma(t)})$ siendo $\sigma(t)$ la desviación típica de la energía para la temperatura *t*. Mirkin *et al*. (1993) hicieron experimentos con el programa de Huang obteniendo que los resultados son independientes del parámetro λ, pero que sin embargo λ tiene un gran impacto en el tiempo de cálculo.

Aunque la mayoría de los programas automáticos de enfriamiento resultan atractivos desde un punto de vista teórico, la evidencia empírica indica que su

uso es esporádico y que su éxito depende en gran medida del problema que se pretende resolver.

4. Decisiones específicas para cada problema

Las decisiones específicas se refieren sobre todo a la definición del espacio de soluciones, la estructura de los entornos y la función de coste, así como a la elección de la solución inicial. Los resultados de Hajek (1988) apoyan la idea de que el modo en que estas decisiones interactúan al definir el ámbito de la búsqueda, tiene un efecto muy significativo en la calidad de la solución final alcanzada.

Está generalmente aceptado que la solución inicial s_0 debe ser generada de modo aleatorio. Para otras alternativas, como ocurría con las decisiones genéricas, no hay reglas definitivas que seguir, aunque es posible apuntar algunos factores que deberían ser tenidos en cuenta.

Algunas de las primeras teorías sobre RS se basaban en entornos uniformes y simétricos, es decir, todos los entornos eran del mismo tamaño y si $i \in N(j) \Rightarrow j \in N(i)$. Sin embargo, hoy se sabe que es suficiente con exigir el cumplimiento de una condición más suave que exige que cualquier solución pueda alcanzarse desde cualquier otra a través de una cadena de movimientos válidos, usando los entornos. Es decir, es importante garantizar esta condición de "alcanzabilidad" (o "ergodicidad") al definir la estructura del espacio de soluciones y de los entornos.

Para garantizar que el tiempo de cálculo se usa de un modo efectivo, es importante verificar que las rutinas más frecuentemente usadas funcionen lo más rápido posible. Nótese que en cada iteración se llama siempre a dos funciones, la primera de las cuales es la generación aleatoria de una solución del entorno $N(s_0)$. Aunque este es un proceso bastante trivial para la mayoría de las definiciones de espacios de soluciones y entornos, se puede volver más difícil cuando los entornos son grandes y complejos, y cuando el espacio de soluciones está condicionado por muchas restricciones. Como luego veremos en un ejemplo de coloración de grafos, a veces es beneficioso relajar algunas de las restricciones en la definición del espacio de soluciones, y penalizar por el contrario mediante la función de coste las violaciones de las restricciones no consideradas.

La segunda función llamada en cada iteración es el cálculo del cambio en la función de coste. En este caso se utilizará más eficientemente el tiempo si este cambio se calcula usando información relativa a qué parte de la solución ha cambiado, sin proceder a evaluarla totalmente sin tener en cuenta la información previamente disponible.

Otro aspecto que mejora el tiempo de computación reside en el cálculo del factor de Boltzmann: Johnson *et al*. (1989), tras observar que en su problema la tercera parte del tiempo de computación se emplea en calcular las funciones exponenciales, proponen crear una tabla en la que miran directamente esos valores. El tamaño de la tabla ellos lo han fijado en 1000 posiciones, tras considerar que valores no automáticamente rechazables o aceptables son, para cada t, los $\delta \in [t/200 \ , \ 5t]$. Cada posición representa el $\delta = i \cdot (t/200)$, y por tanto, en la posición i se almacenará el valor $e^{-\delta/t} = e^{-i/200}$. El índice para mirar en esta tabla se obtiene calculando $\delta \cdot (200/t)$, redondeando y acotando los valores entre 1 y 1000.

Los resultados de Hajek indican que puede no ser una buena idea usar entornos que den lugar a espacios de búsqueda con muchos picos o con grandes y profundos valles. Es también evidente que la función de coste debería guiar la búsqueda hacia buenos mínimos locales, y que las grandes llanuras en las que todas las soluciones del entorno tienen el mismo valor, deberían ser evitadas. La eficiencia de la búsqueda en un número finito de iteraciones dependerá igualmente del tamaño del espacio de soluciones, y por tanto el objetivo debería ser mantenerlo lo más pequeño posible. Esto podría hacerse por ejemplo mediante un pre-procesado que reduzca el tamaño del espacio de soluciones. Sin embargo, debe tenerse cuidado que esto no viole las condiciones de alcanzabilidad, o que de lugar a superficies con picos que son más difíciles de explorar.

Al igual que con espacios de soluciones reducidos, hay ventajas al considerar entornos razonablemente reducidos, que aseguren que la búsqueda no converja hacia óptimos locales a temperaturas bajas. Sin embargo, esto debe ser compaginado con la facilidad que permiten los grandes entornos de que se produzcan mejoras importantes en un movimiento simple, y de facilitar rápidos movimientos de un área a otra del espacio de búsqueda. La evidencia empírica en un gran número de problemas parece indicar que las preferencias entre entornos pequeños y grandes son dependientes de cada problema.

Todas las anteriores ideas se ilustrarán ahora usando como ejemplo dos problemas clásicos: el problema del viajante y el de coloración de un grafo.

5. El problema del viajante

Antes de la aparición del Recocido Simulado, ya se habían definido numerosos algoritmos de búsqueda local para el problema del viajante. Como es sabido, para un problema con n ciudades, el conjunto de soluciones factibles puede ser representado como el conjunto de permutaciones de los números 1 a n. Por tanto, el espacio consiste en $n!$ soluciones. Un k-entorno de una solución se define como el conjunto de soluciones obtenidas al eliminar k tramos y reemplazarlos por otros k tramos diferentes. Para k = 2, sólo hay un modo de reconectar las ciudades tras haber quitado k tramos. Si v_i representa la ciudad en la posición i en una solución, si se eliminan los tramos (v_i, v_{i+1}) y (v_j, v_{j+1}), el único modo de reconectar el viaje es unir v_i con v_j, y v_{i+1} con v_{j+1}. En un problema simétrico el tamaño del entorno es $O(n^2)$, y tiene la gran propiedad de que es de varios órdenes de magnitud menor que el espacio de soluciones. Es sencillo explorar uniformemente el entorno generando un entero en el intervalo 1 a n, y el cambio en la función de coste se puede calcular sin re-evaluar todo el viaje, como $d(v_i,v_j) + d(v_{i+1},v_{j+1}) - d(v_i,v_{i+1}) - d(v_j,v_{j+1})$

6. El problema de coloración de un grafo

El problema de coloración de los vértices de un grafo consiste en asignar un color a cada vértice usando el mínimo número posible de colores, y de modo que dos vértices adyacentes nunca tengan asignado el mismo color. Al contrario que el problema del viajante, éste no fue un candidato obvio para ser resuelto mediante búsqueda local, aunque demostró ser buen candidato para su resolución mediante RS. Sin embargo, las decisiones específicas deben ser tomadas con cuidado, y fueron definidas de modos muy distintos por distintos investigadores.

En este caso, la definición natural de espacio de soluciones es el conjunto de coloraciones, lo cual se puede representar por un conjunto de particiones de los vértices en subconjuntos, de modo que no haya vértices adjuntos en cada subconjunto. El coste de cada partición es el número de subconjuntos que contiene.

La definición de entorno más simple es mover uno o más vértices de un subconjunto a otro, pero aquí surge el problema de hacerlo de modo que se siga manteniendo la factibilidad. Es decir, el conjunto de potenciales movimientos es reducido, lo que supone dos problemas: por un lado, si escogemos uniformemente un vértice y luego uniformemente entre el conjunto de movimientos factibles para ese vértice, no estaremos explorando uniformemente todo el entorno; en segundo lugar, es fácil encontrar ejemplos donde la condición de alcanzabilidad se vulnera y no existe un camino de movimientos válidos desde la solución inicial a la solución óptima.

Un modo de obviar esta dificultad es usando definiciones de entorno más complejas, en los cuales se mueven de un subconjunto a otro cadenas de vértices (cadenas de Kempe), manteniendo siempre la factibilidad: al cambiar un nodo a otro subconjunto, todos los de su cadena se cambian al contrario del que estuvieran (Figura 3) . Esto tiene la desventaja de que el conjunto de cadenas factibles debe ser actualizado tras cada movimiento, y como las cadenas serán de diferentes tamaños, algunos vértices tendrán una mayor probabilidad de ser movidos que otros.

Otra segunda probabilidad es relajar la condición de factibilidad y permitir en el espacio de soluciones cualquier partición de nodos. En este caso debería añadirse al coste una penalización igual al número de arcos existentes entre los vértices que estén en un mismo subconjunto.

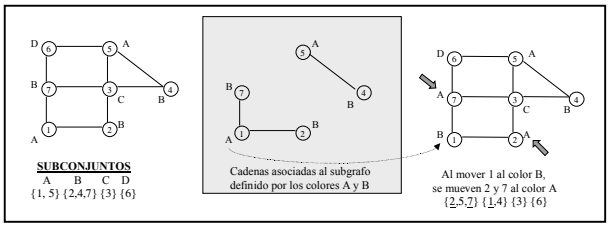

FIGURA 3. Ejemplo de movimiento usando cadenas, en el problema de coloración

La definición natural de coste como el número de colores (o subconjuntos usados) no es el más adecuado desde el punto del Recocido Simulado, ya que no variará el coste con un movimiento, salvo que éste dé lugar a que un subconjunto se quede vacío. Esto dificulta la guía de la búsqueda entre las soluciones factibles. Lo que sería necesario es una función de coste que premie soluciones en las que haya algunos subconjuntos grandes junto con otros pequeños, y no aquéllas en las que los vértices están particionados de un modo muy uniforme. Por ejemplo, Johnson *et al*. (1991) usaron una función de coste de la forma $-\sum_{i=1}^{Q}|C_i|^2 + \sum_{i=1}^{Q} 2|C_i||E_i|$, donde Q es el número de subconjuntos usados, y C_i

y E_i son respectivamente el conjunto de vértices y de arcos en el subconjunto i. Esta función no garantiza que su mínimo se corresponda con una coloración con el mínimo número de colores, pero sí asegura que todos los óptimos locales serán coloraciones factibles.

Una alternativa que evita el problema de tener que encontrar los pesos correctos para las dos partes del objetivo es fijar el valor Q, y minimizar simplemente $\sum_{i=1}^{Q} |Ei|$. Una vez que se haya alcanzado el valor óptimo de cero, se puede reducir Q y repetir de nuevo el proceso. Esto presenta la dificultad de que la búsqueda debe converger al valor óptimo para cada Q, pero por contra, tiene la ventaja de que este valor es conocido, por lo que no se pierde tiempo buscando una solución mejor. Johnson *et. al*. (1989) y Johnson *et al*. (1991) describen un conjunto de experimentos usando diferentes implementaciones para diferentes problemas (incluido el problema de coloración de un grafo), dando útiles ideas de cómo las decisiones específicas y genéricas influyen en la calidad de las soluciones.

7. Aplicaciones

Los dos anteriores ejemplos abordan dos problemas clásicos. Los problemas reales, al ser por lo general muy grandes, suelen ser especialmente difíciles, complicándose a menudo al incorporar múltiples y antagónicos objetivos, y una mezcla de restricciones fuertes y débiles. De cualquier modo, hay muchos ejemplos en los que, tras considerar los aspectos anteriormente discutidos, implementaciones simples de RS han demostrado ser muy eficientes.

Ejemplos recientes incluyen los trabajos de Lučić and Teodorović (1999) para la planificación de los turnos de una compañía aérea, D'Amico *et al*. (2002) para el problema de dividir una región en distritos para la asignación de patrullas de policía, o Antunes y Peeters (2001) para la solución de un problema de localización multi-período relativo a la reorganización de la escolarización elemental en Portugal.

Sin embargo, para otros problemas las soluciones simples pueden no ser tan eficientes por diversos motivos. En algunos casos, el problema puede que no sea buen candidato a ser resuelto mediante RS, sino por otras formas de modificación de las soluciones, debido a las características particulares del problema. Discutir todas estas posibilidades en detalle sería muy largo como para presentarlo aquí, pero los ejemplos que más abajo se indican dan una idea de las variantes que han demostrado ser exitosas.

Una queja bastante común relativa a RS es que esta metodología puede necesitar mucho tiempo de cálculo cuando los problemas son grandes y complejos. Como el tiempo de cálculo está directamente relacionado con el tamaño del espacio de soluciones, el modo más obvio de reducir el tiempo es intentar reducir el espacio de soluciones. Muchos problemas clásicos de optimización pueden reducirse a través de un pre-procesamiento mediante un conjunto de reglas de reducción. Este es el enfoque elegido por Bornstein y Azlan (1998) en su solución RS para el problema de localización de una planta con restricciones de capacidad, y por Dowsland (1993) en su solución para problemas bidimensionales de empaquetado rectangular.

Cuando esto no sea posible, una alternativa podría ser reducir el espacio de soluciones para parte de la búsqueda. Lutfiyya *et al.* (1992) resuelven un problema de empaquetado irregular donde las soluciones están representadas por las posiciones en un mallado de la piezas empaquetadas. Un mallado más denso implica una representación más exacta, y por tanto una mejor calidad del óptimo, pero tiene el inconveniente de suponer un mayor tamaño del espacio de búsqueda. Los autores han afrontado este problema manejando un mallado menos fino durante las etapas iniciales del algoritmo permitiendo así que la búsqueda encuentre áreas prometedoras. Posteriormente el mallado se vuelve más fino conforme se reduce la temperatura, facilitando una búsqueda más exacta conforme el algoritmo empieza a converger.

Otra posibilidad es reducir la proporción del entorno que puede ser explorado en cada iteración. Por ejemplo, para problemas del tipo de creación de horarios o de secuenciación donde el objetivo es hallar una solución factible, la función de coste puede representar el número de restricciones no satisfechas siendo el objetivo reducir ese número a cero. En este caso, es habitual considerar exclusivamente movimientos que contribuyan a reducir el coste. Tanto Lutfiyya *et al.* (1992) como Sechen *et al*. (1988) usan entornos reducidos para evitar perder tiempo de cálculo a temperaturas bajas. Ambos indican que movimientos grandes son muy raramente aceptados cuando t es pequeña, y por tanto reducen el tamaño del entorno conforme el sistema se enfría, evitando así una pérdida de tiempo explorando y rechazando saltos grandes en las últimas iteraciones del algoritmo.

En algunas ocasiones es posible que no podamos definir una adecuada combinación de parámetros

específicos que aseguren una superficie "suave" (sin picos ni valles profundos) del espacio de soluciones. En esos casos, una alternativa útil podría ser hacer un "recalentamiento", el más común de los cuales consiste en hacerlo muy esporádicamente cuando apreciemos signos de que nos hemos quedado atrapados o que no nos movemos durante un elevado número de iteraciones de una determinada solución.

Sin embargo, otras estrategias han sido también exitosas. Por ejemplo Dowsland (1993) usó para un problema de empaquetamiento bidimensional un programa de enfriamiento ondulatorio que enfría cuando se acepta un movimiento, y se recalienta a un ritmo mucho menor cada vez que se rechaza un movimiento, evitando así quedarse atrapado en un óptimo local profundo. Connolly (1990) afirma para un problema de asignación cuadrática que "recocer" a una temperatura constante le dio mejores resultados que el recocido clásico. Ross (2000) llevó a cabo unos experimentos detallados usando estrategias convencionales y de recalentamiento en un problema de diseño de redes de distribución, concluyendo que resulta interesante considerar alguna forma inteligente de recalentamiento.

También pueden causar problemas la necesidad de fijar un conjunto adecuado de pesos en problemas multiobjetivo, o cuando necesitamos penalizar en la función de coste las restricciones que han sido relajadas en la definición del espacio de soluciones. Dige *et al.* (1993) tuvieron serios problemas usando estos últimos pesos para obtener soluciones factibles en su implementación RS para el problema de elaboración de horarios escolares. Wright (1991) afirma que los pesos deberían ser fijados de acuerdo con la dificultad para poder satisfacer las restricciones, y no proporcionalmente a su importancia, obteniendo muy buenos resultados al aplicar estas ideas a diferentes problemas de secuenciación relacionados con el mundo del cricket.

Thompson and Dowsland (1998) abordaron problemas similares en la implementación de una solución para el problema de secuenciación de exámenes, resolviéndolo en dos fases: en la primera se relajó el espacio de soluciones y la búsqueda se concentró en encontrar una solución factible; una vez hallada, se eliminan del espacio de búsqueda todas las infactibles y se amplía el entorno para eliminar todo potencial problema de alcanzabilidad; la búsqueda continúa concentrándose en los objetivos secundarios y las restricciones "blandas".

Otros (por ejemplo Stern (1992)) han obtenido buenos resultados con pesos que son dependientes de la temperatura. Incluso en aquellos casos en los que todos los elementos de la función de coste son objetivos auténticos, puede ser que el equilibrio óptimo entre los distintos objetivos no sea constante durante todo el período de búsqueda, por lo que pesos constantes no serán completamente exitosos. Wright (2001) propone un modo de evitar esto, que denomina "búsqueda de subcoste guiada". En ella la función de aceptación se define que tal modo que permitirá un mayor grado de empeoramiento global en la función de coste, siempre que dé lugar a una mejora notable en al menos un objetivo.

8. Conclusiones

Los anteriores ejemplos son simplemente una pequeñísima muestra del gran número de problemas que han utilizado con éxito el Recocido Simulado. Otros ejemplos aparecen en artículos de revisión tales como Díaz *et al.* (1996) (donde se presentan numerosas aplicaciones en los campos del diseño VLSI, telecomunicaciones, grafos, física, biología, etc), Dowsland (1993), Aarts and Lenstra (1997) o Koulamas *et al.* (1994). Cualquier búsqueda en una base de datos científica proporciona docenas de citas cubriendo un amplio espectro de problemas teóricos y prácticos. Hay también numerosos ejemplos de hibridación entre RS y otras técnicas heurísticas, particularmente con algoritmos genéticos.

Una de las atracciones del Recocido Simulado es que resulta muy fácil trabajar con él en cualquier problema que presente una estructura de entorno natural, como ocurre en muchos problemas de optimización combinatoria (aunque algunos como el de *Number Partitioning* parecen presentar especial resistencia a su resolución mediante RS). Sin embargo, incluso para los más simples problemas se debe tener cuidado a la hora de definir las decisiones genéricas y las específicas del problema. En muchos casos eso es suficiente para obtener un buen algoritmo. Otras veces, debe ponerse especial atención en definir un programa de enfriamiento bien afinado, o en aprovechar el conocimiento que se tenga del problema específico para definir una buena estructura de entornos, espacio de soluciones o función de coste.

Referencias

Aarts, E.H.L., Korst, J.H.M.: Simulated Annealing and Boltzmann Machines, Wiley, Chichester, 1989.

Aarts, E.H.L., Lenstra, J.: Local Search in Combinatorial Optimisation. Wiley, Chichester 1997.

Antunes, A., Peeters, D.: On solving complex multi-period location models using simulated annealing. European Journal of Operational Research 130: 190-201, 2001.

Bornstein, C.T., Azlan, H.B.: The use of reduction tests and simulated annealing for the capacitated plant location problem. Location Science 6: 67-81, 1998.

Cerny, V.: A thermodynamical approach to the travelling salesman problem: an efficient simulation algorithm. J. of Optimization Theory and Applic.,45: 41-55, 1985.

Connolly, D.T.: An improved annealing scheme for the QAP. European Journal of Operational Research 46: 93-100, 1990.

D'Amico, S.J., Wang, S-J, Batta, R., Rump, C.M.: A simulated annealing approach to police district design. Computers & Operations. Research 29: 669-684, 2002.

Díaz, A. (coordinador), *Optimización heurística y Redes Neuronales en Dirección de Operaciones e Ingeniería*, Paraninfo, Madrid, 1996.

Dige, P., Lund, C., Ravn, H.F.: Timetabling by simulated annealing. Applied Simulated Annealing – Lecture Notes in Economic and Mathematical Systems 396, R.V.V. Vidal (ed), 151-174, Springer-Verlag, 1993.

Dowsland, K.A.: Some experiments with simulated annealing techniques for packing problems. European Journal of Operational Research 68: 389-399, 1993.

Dowsland, K.A.: Simulated Annealing. in Modern Heuristic Techniques for Combinatorial Problems, C.R.Reeves (ed), Blackwell, Oxford, 1993.

Hajek, B.: Cooling schedules for optimal annealing. Mathematics of Operations Research.,13: 311-329, 1988.

Huang, M.D., Romeo, F., Sangiovanni-Vincentelli, A.: An efficient general cooling schedule for simulated annealing. Proc. IEEE International Conference on Computer Aided design, Santa Clara. 381-384, 1986.

Johnson, D.S., Aragon, C.R., McGeoch, L.A., Schevon, C.: Optimization by simulated annealing: an experimental evaluation; Part I Graph partitioning. Operations Research 37: 865-892, 1989.

Johnson, D.S., Aragon, C.R., McGeoch, L.A., Schevon, C.: Optimization by simulated annealing: an experimental evaluation; Part II Graph coloring and number partitioning. Operations Research 39: 378-406, 1991.

Kirkpatrick, S., Gellat, C.D., Vecchi, M.P.: Optimization by simulated annealing. Science, 220: 671-680, 1983.

Koulamas, C., Antony, S.R., Jane, R.: A survey of simulated annealing applications to operations research problems. OMEGA 22: 41-56, 1994.

Lučić, P., Teodorović, D.: Simulated annealing for the multi-objective aircrew rostering problem. Transportation Research (A) 33: 19-45, 1999.

Lundy, M., Mees, A.: Convergence of an annealing algorithm. Math. Prog. 34: 111-124, 1986.

Lutfiyya, H., McMillin, B., Poshyanonda, P., Dagli, C.: Composite stock cutting through simulated annealing. Mathematical. Computational Modelling 16: 57-74, 1992.

Metropolis, N., Rosenbluth, A.W., Rosenbluth, M.N., Teller, A.H., Teller, E.: Equation of state calculation by fast computing machines. Journal of Chemistry Physics, 21: 1087-1091, 1953.

Mirkin, G., Vasudevan, K., Cook, F.A.: A comparison of several cooling schedules for simulated annealing implemented on a residual static problem. Geophysical Research Letters 20: 77-80, 1993.

Ross, A.D.: A two-phased approach to the supply network re-configuration problem. European Journal of Operational Research 122: 18-30, 2000.

Sechen, C., Braun, D., Sangiovanni-Vincentelli, A.: Thunderbird: a complete standard cell layout package. IEEE Journal of Solid-State Circuits, SC23: 410-420, 1988.

Stern, J.M.: Simulated annealing with a temperature dependant penalty function. ORSA Journal on Computing, 4: 311-319, 1992.

Thompson, J.M., Dowsland, K.A.: A robust simulated annealing based examination timetabling system. Computers & Operations Research 25: 637-648, 1998.

Van Laarhoven, P.J.M., Aarts, E.H.L.: Simulated Annealing – Theory and Applications. Kluwer, Dordrecht, 1988.

White, S.R.: Concepts of scale in simulated annealing. Proc. IEEE, International Conference on Computer Design, 646-651, 1984.

Wright, M.: Scheduling English cricket umpires. Journal of the Operational Research Society 42: 447-452, 1991.

Wright, M.: Subcost-guided search – experiments with timetabling problems. Journal of Heuristics, 7: 251-260, 2001.