Mejoras en la calidad de las soluciones de *simulated annealing* mediante preservación de menores costos

Gastón Sivori[1], Claudio Verrastro[1,2,3], Juan Carlos Gómez[1,2,4]

[1] Universidad Tecnológica Nacional, Facultad Regional Buenos Aires, Departamento de Ingeniería Electrónica, Av. Medrano 951 (C1179AAQ) Ciudad Autónoma de Buenos Aires, Argentina
[2] Universidad Tecnológica Nacional, Grupo de Inteligencia Artificial y Robótica (GIAR), Facultad Regional Buenos Aires, Av. Medrano 951 (C1179AAQ) Ciudad Autónoma de Buenos Aires, Argentina
[3] Comisión Nacional de Energía Atómica (CNEA), Centro Atómico Ezeiza, Instrumentación y Control, Camino Real Presbítero González y Aragón 15, (B1802AYA), Buenos Aires, Argentina
[4] Instituto Nacional de Tecnología Industrial (INTI), Instrumentación y Control, Electrónica e Informática, Avenida General Paz 5445, Edificio 42, (B1650WAB) San Martín, Buenos Aires, Argentina

gast.sivori@gmail.com

Recibido el 10 de agosto de 2017, aprobado el 19 de diciembre de 2017

Resumen

Simulated Annealing (SA) o Metropolis es un algoritmo de búsqueda de soluciones que emplea meta-heurística para problemas de optimización global donde el objetivo es encontrar buenas aproximaciones al valor óptimo de una función en un espacio de búsqueda grande. Su implementación estándar presenta dispersión en la calidad de las respuestas encontradas debido a que explora el espacio de soluciones en forma estocástica. En este trabajo, se presenta una implementación del algoritmo de SA que utiliza la preservación de menores costos en problemas de optimización global y permite obtener soluciones mejores, con menor dispersión y manteniendo tiempos de convergencia finitos. La preservación de los menores costos se realiza durante el proceso de generación aleatoria (perturbación) de la nueva solución a partir de la solución corriente. La modificación se hace conservando una parte de la solución elegida con una probabilidad inversamente proporcional a su costo parcial. La implementación del algoritmo se llevó a cabo en Matlab® y se comparó contra el recocido simulado estándar en problemas TSP simétricos de la librería TSPLIB obteniendo soluciones excelentes para problemas de n < 200 nodos.

PALABRAS CLAVE: SIMULATED ANNEALING - PROBLEMAS COMBINACIONALES - TSP SIMÉTRICO - PRESERVACIÓN DE CAMINOS - OPTIMIZACIÓN

Abstract

Simulated Annealing (SA) or Metropolis is a search algorithm that uses meta-heuristics for solve global optimization problems, where the goal is find-out good approximations to the optimal value of a function in a large search space. Its standard implementation has a high degree of dispersion in the solutions quality due to the stochastic nature of the search procedure. In this paper, an implementation of the SA algorithm that promote short paths (SPP) in global optimization problems yields better solutions keeping finite convergence times. Preservation of short paths is performed during the random generation process of a new solution (perturbation). The random selected partial path is preserved in the new solution with an inversely probability to its cost. The implementation of this algorithm was carried out in Matlab®, the results are evaluated comparing with the standard implementation in symmetrical TSP problems from the TSPLIB, the performance obtained was promising for problems of n < 200 nodes.

KEYWORDS: SIMULATED ANNEALING - COMBINATIONAL PROBLEMS - SYMMETRICAL TSP - PATH PRESERVATION - OPTIMIZATION

Introducción

En problemas de ingeniería, las soluciones a problemas de optimización global, en especial aquellos de optimización combinatoria, se busca obtener resultados óptimos o sub-óptimos y que además se haga de manera eficiente. El problema del viajante de comercio (TSP, por sus siglas en inglés), es un problema arquetípico, donde se busca el ciclo hamiltoniano más corto en una secuencia de n ciudades. El viajante visita cada ciudad una única vez y retorna al punto de partida. Además, desde cualquier ciudad se puede ir a cualquiera de las restantes.

El estudio de soluciones a dicho problema ha tenido repercusión para investigadores en varios campos como la investigación operativa, inteligencia artificial, biología, matemática, física y se han publicado muchos artículos referidos a este problema por su gran importancia como problema combinacional (Lawler, 1991). Esto es porque el tiempo de resolución a través de métodos de exploración exhaustiva es una función No determinista Polinómica (NP) respecto de la complejidad del problema que aumenta según n!. En otras palabras, TSP representa un excelente modelo para resolver problemas prácticos de la ciencia e ingeniería tales como los aplicables a, investigación operativa, programación lineal, los rayos X, la cristalografía, el mapeo de rutas, el diseño de circuitos impresos, vehículos autónomos, logística y distribución, etc., porque se ha demostrado que TSP es NP completo (Teorema de Cook-Levin, 1971) y las soluciones pueden ser extrapoladas a otros problemas NP.

Hay muchos algoritmos de aproximación para resolver problemas combinacionales como lo son A*, greedy, vecino más cercano, métodos de particiones, mínima arborecencia, etc. (Held y Karp, 1969)

También se han ensayado variantes del recocido simulado estándar en forma paralelizada (Sohn, 1994; Sohn *et al.*, 1996), basándose en algoritmos genéticos (Francisco, 2010) y en redes neuronales (Scott, 1996). Asimismo, se han propuesto métodos sumamente rápidos en la obtención de soluciones (Ingber, 1989; Taghi, 2008; Hong, 2006), y métodos más exhaustivos que mejoran la calidad de las soluciones obtenidas (Keikha, 2011).

Este trabajo concentra sus esfuerzos sobre problemas de TSP que tienen la característica de ser simétricos. Esto significa que, independientemente del sentido de recorrido, el costo de viajar entre dos ciudades es el mismo.

$$d_{ij} = d_{ji} \qquad (1)$$

Con el interés de mejorar la calidad de soluciones que ofrece el recocido simulado estándar, se implementó un algoritmo de perturbación que permite mejorar la calidad de soluciones obtenidas a partir de la preservación de los caminos más cortos que presenta el problema. Dicho trabajo se llevó a cabo en Matlab® y, para compararlo con la implementación estándar, se utilizaron problemas que posee la librería TSPLIB.

En la siguiente sección se explica el marco teórico referido al recocido simulado en su implementación estándar. En la tercera sección se expone la implementación desarrollada en este trabajo y en la cuarta sección se expresan los resultados de contrastar dicha implementación contra el estándar para problemas de la librería TSPLIB. En la última sección se expresan las conclusiones y sugerencias de los autores.

Breve resumen de *Simulated Annealing* (SA)

Al aumentar el número *n* de ciudades que se visitan la cantidad de soluciones posibles se incrementa exponencialmente. Esto implica que encontrar la solución óptima conlleva resolver un problema con una complejidad de n!. Encontrar esta solución mediante fuerza bruta requiere evaluar todas las soluciones posibles. Esto no es práctico y en muchos casos no es posible en tiempo finito.

El recocido simulado es un algoritmo de búsqueda meta-heurístico donde se desea encontrar una buena aproximación al valor óptimo de una función en un espacio de búsqueda grande. Su principio de funcionamiento se inspira en el tratamiento térmico de recocido del acero y otros materiales, cuya finalidad es liberar las tensiones internas del material después del moldeo, la forja o la soldadura. El calentamiento produce que los átomos dentro de la estructura cristalina del material adquieran una distribución aleatoria, que con el enfriamiento van ordenándose hasta lograr una estructura cristalina más regular. Es

en dicho punto donde se alcanza la mínima energía del sistema y el material obtiene su máxima tenacidad.

En 1953, Metropolis (Metropolis, Rosenbluth y Teller, 1953) desarrolló un algoritmo inspirado en este proceso de recocido, partiendo de una solución inicial cualquiera, a cada paso del algoritmo se provoca una perturbación aleatoria y se evalúa la nueva solución. Si este cambio (ΔE), asimilado a la energía, resulta en una mejora se acepta la nueva solución. De lo contrario, si ΔE > 0, se acepta el desplazamiento con probabilidad según la expresión (2).

$$e^{-\frac{\Delta E}{T.K}} \qquad (2)$$

en donde, K es la constante de Boltzman y T la temperatura del sistema.

Parámetros de SA

Una de las desventajas de este algoritmo es la dificultad en el ajuste de sus parámetros y su influencia según la complejidad del problema. Estos parámetros han sido estimados en (Bruce y David, 1991; Shojaee *et al.*, 2010; Deutsch y Wen, 1998). Y estudiados rigurosamente en (Maribel *et al.*, 2010). A continuación se detallan los parámetros fundamentales del recocido simulado:

- Esquema de enfriamiento (CS): representa el decrecimiento gradual de la temperatura. Debe ser lo suficientemente lento para que el algoritmo explore satisfactoriamente el espacio de soluciones pero, al mismo tiempo, lo suficientemente rápido para que la solución converja en un tiempo finito.
- Temperatura inicial (IT): representa la temperatura inicial de recocido simulado. Puede establecerse manualmente o estimarse, minimizando el tiempo de ejecución del algoritmo.
- Número de iteraciones (NI): representa la cantidad de iteraciones que generará el algoritmo para reemplazar la solución actual en pos de una solución de mayor calidad. Dadas las características del algoritmo, es imprescindible tomar de vez en cuando soluciones peores que permitan mayor exploración del espacio de soluciones.
- Número de cambios (NC): representa el criterio de salida del algoritmo. Define la cantidad máxima de intentos permitidos en los que la solución no mejoró.

Parte Experimental

Solución inicial

La solución inicial en el recocido simulado se genera mediante una secuencia aleatoria de los n nodos a visitar que componen un *tour*, donde el primer y último elemento se corresponden con la primera ciudad que es origen y finalización del *tour*. Dado que los problemas evaluados son simétricos, la ciudad de origen no afecta a la solución final.

Implementación estándar de SA

En la Figura 1 se puede apreciar el código de la implementación estándar del recocido simulado (SA).

Perturbación de la solución actual

En la implementación estándar, la perturbación de la solución se genera mediante una ventana de longitud L generado aleatoriamente posicionada en el índice del nodo i y que cumplen con las Ecuaciones 3 y 4.

$$1 \leq i \leq n-2 \qquad (3)$$

$$2 \leq L \leq n-i \qquad (4)$$

Este tipo de perturbación se denomina reversión de bloque, donde para generar una nueva solución, se invierte el orden de L posiciones de la solución actual. El ejemplo 1 muestra este tipo de perturbación para *n=5, i=2 y L=3.*

Estado Inicial	1 2 3 4 5 1
Ventana	0 0 1 1 1 0
Estado Perturbado	1 2 5 4 3 1

Esquema de enfriamiento

Asimismo, el enfriamiento del sistema se define a partir de factores de enfriamiento geométricos como el de la Ecuación 5 o el factor de Lundy y Mess (Lundy y Mess, 1986) en la Ecuación 6.

$$T_n = \alpha * T_{n-1} \qquad (5)$$

donde $\alpha < 1$, típicamente es 0,9.

```
% Genero una solución inicial
s = fInitialState(n);
Num_Intentos = 0;
Iteraciones = 0;
while Iteraciones < NI
% Genero un estado vecino al actual
   sNew = fPerturbate(s);
% Calculo el costo del estado actual y del
vecino obtenido
   Costo_Actual = fCostCalculator(s,dmat,n);
   Costo_Nuevo =
fCostCalculator(sNew,dmat,n);
% Evaluación de la perturbación
   if Costo_Actual <= Costo_Nuevo
      s = sNew;
      Num_Intentos = 0;
   else
% Aceptación del estado con cierta
probabilidad
      if fEvaluate (Costo_Actual,
Costo_Nuevo, Temp)
         s = sNew;
         Num_Intentos = 0;
      else
         Num_Intentos = Num_Intentos + 1;
      end
   end
% Condición de salida
   if Num_Intentos > NC
      break
   end
   Iteraciones = Iteraciones + 1;
% Decremento la temperatura del recocido
simulado.
   Temp = fTempDecay(Temp,Parametro);
end
```

Ejemplo 1: Perturbación por reversión de bloque de longitud L

$$T_n = \alpha * T_{n-1} \qquad (6)$$

donde $\beta > 0$, típicamente 0,1.

En general, el factor de enfriamiento permite mayor exploración del espacio de soluciones del problema si se garantiza que la temperatura del recocido simulado decrezca lentamente. Si decrece demasiado lentamente, los tiempos de convergencia serán altos.

Temperatura Inicial

La temperatura inicial del recocido simulado es un parámetro sumamente importante para disminuir los tiempos de convergencia de las soluciones. Su valor depende pura y exclusivamente de la complejidad de problema a tratar y de la cantidad de nodos del mismo. Si la temperatura inicial es demasiado alta, la probabilidad de aceptación se mantendrá alta por una gran cantidad de iteraciones lo que significa que se desaprovechará tiempo de ejecución del algoritmo, mientras que valores bajos hacen que el algoritmo arroje respuestas mediocres o de mala calidad debido a la pobre exploración del espacio de soluciones. Para impedir esto, la temperatura inicial se suele estimar y para ello existen varios métodos (Shakouri *et al.*, 2009; Santa Chávez, *et al.*, 2014; Ben-Ameur, 2004). En este trabajo se han estimado las temperaturas iniciales de cada problema según (Santa Chávez *et al.*, 2014) y se calcula al comienzo de cada ejecución de la siguiente forma:

. Se genera una alternativa inicial con una función de evaluación asimilable a la energía del sistema E_1.

. Se generan k perturbaciones aleatorias, dónde k es dependiente de la cantidad de ciudades del problema (se utilizó $k = n^2$ en la práctica), y se calcula la función de evaluación E_j de cada una.

. Se obtiene el valor promedio del ΔE (diferencia de energía entre el estado inicial y el estado perturbado) usando la Ecuación 7.

$$\Delta E = \sum_{j=1}^{k} \frac{|E_j - E_1|}{k} \qquad (7)$$

Se elige una tasa de aceptación τ_0=0,5 para que la probabilidad de saltar a un peor estado (mayor energía) al comienzo sea aproximadamente del 50%.

Utilizando la expresión (2) se puede escribir la tasa de aceptación como en la Ecuación 8.

$$\tau_0 = e^{-\frac{\Delta E}{T_0 * K}} \qquad (8)$$

Se despeja la temperatura inicial T_0 y se obtiene de la Ecuación 9.

$$T_0 = -\frac{\sum_{j=1}^{k} \frac{|E_j - E_1|}{k}}{\ln(\tau_0) * K} = \frac{\sum_{j=1}^{k} \frac{|E_j - E_1|}{k}}{0{,}69 * K} \qquad (9)$$

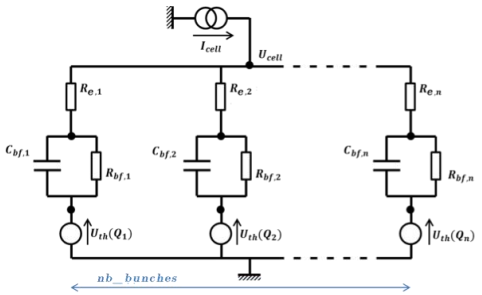

Fig. 2. Gráfico de respuesta de SA para el problema eil51. La temperatura inicial utilizada es de 1000 °C pero la temperatura estimada es de 27,26°C, trazo azul. La línea roja punteada indica la mejor solución conocida hasta el momento.

A modo de ejemplo, en la Figura 2 se puede observar la respuesta del algoritmo de recocido simulado estándar para una temperatura inicial de 1000 °C en la ejecución del problema eil51 de la librería TSPLIB. Asimismo, se calculó la temperatura inicial del problema con la Ecuación (9) (T_0=27,26°C) observándose que esta temperatura T_0 es adecuada ya que empezando en 1000 ºC, se llevaron a cabo aproximadamente 30000 iteraciones innecesarias, en las cuales el algoritmo se mantuvo aceptando soluciones sin progreso en la optimización global.

Por este motivo, la estimación de la temperatura inicial se utilizará en las dos implementaciones del algoritmo de recocido simulado que se comparan en este trabajo.

Preservación de caminos cortos (PCC) en SA

Una de las grandes desventajas que presenta el recocido simulado estándar es que la calidad de las soluciones encontradas posee mucha dispersión. A la vez que es necesario ejecutarlo varias veces dada su naturaleza estocástica. En general, si el problema no presenta grandes complejidades, las soluciones son muy buenas. Pero a medida que la cantidad de ciudades aumenta, la calidad de las soluciones empeora alcanzando errores altos.

Para enfrentar estos problemas se implementó un algoritmo que utiliza la preservación de caminos cortos (PCC) (Verrastro, 1991). La idea detrás de esto es proveer al recocido simulado de un tipo de perturbación distinta a la estándar que guíe la exploración en el espacio de soluciones. En el recocido simulado estándar, la perturbación es generada de manera aleatoria generando reversión de bloques que para problemas de TSP simétricos sólo afectarán a las conexiones con dichos bloques. Con PCC, se desea mantener la componente aleatoria de la perturbación pero preservando los bloques que poseen soluciones parciales de longitud L de menor costo.

Dada una solución S(t) se desea generar una nueva solución S(t+1) para que sea evaluada por el algoritmo de recocido simulado y sea aceptada o no según el criterio de SA.

Antes de comenzar el bucle principal del recocido simulado, se genera una matriz de orden de mérito OM donde cada elemento contiene el orden de mérito del camino C_{xy}. Donde C_{xy} es el costo de un camino de longitud 1 desde el nodo x al nodo siguiente y.

Esta matriz tiene dimensiones de n*n y sus valores son inversamente proporcional a la longitud del camino C_{xy} con respecto a los nodos x e y, se calculan según la Ecuación 10.

$$OM(x,y) = \frac{Tx - C_{xy}}{Tx} + \frac{Ty - C_{xy}}{Ty} \qquad (10)$$

donde Tx y Ty son las sumatorias de los caminos que convergen al nodo x e y respectivamente. De este modo, los valores de OM(x,y) quedan restringidos al intervalo:

$$0 \leq OM(x,y) \leq 2 \qquad (11)$$

Para privilegiar la preservación de caminos parciales cortos se procede de la siguiente manera:

1. Para una solución S(t) compuesta por n nodos C(i), para i de 1 a n, que se desea perturbar. Se calcula el vector de mérito VM(i) de esa solución, donde cada elemento contiene el orden de mérito del camino parcial de longitud L contenido en la solución S(t). Cada elemento

del vector se calcula como la sumatoria de L valores de OM(x,y) de los caminos que componen la solución a partir del nodo i

$$VM(i) = \sum_{j=i}^{i+L-1} OM(C(j), C(j+1)) \quad (12)$$

2. Se genera un vector P(i) de n elementos donde cada elemento es la suma acumulada del elemento anterior con el correspondiente elemento de VM(i)

3. Se genera un número aleatorio entre 0 ≤Rand≤P(n)

4. Se selecciona como inicio del bloque a invertir al nodo iésimo cuya

P(i) ≤Rand≤P(i+1).

5. Se toma como nueva solución S(t+1) a evaluar por el algoritmo de recocido simulado a aquella secuencia donde se invirtió el bloque desde el nodo i al nodo i+L, de la misma forma como se describe en el Ejemplo 1.

La nueva solución será aceptada o rechazada según los criterios del recocido simulado. De esta manera, se prioriza conservar los caminos más cortos que convergen a cada nodo.

La implementación de SA con PCC comienza por preservar longitudes de segmento de L = n – 2. Conforme se obtienen mejores soluciones y la temperatura decrece, la longitud de segmentos a preservar disminuye según un criterio de cambio que se describe en el párrafo siguiente.

En la Figura 3 se presenta el código para la implementación antedicha donde L representa la longitud del segmento a preservar, NC el contador de iteraciones fallidas y K es un factor de ajuste que se estableció en K = 10 y se incrementa en 1 conforme se disminuye la longitud L. Toda vez que se modifica la longitud del segmento L se lleva a cabo un cambio de época donde, se asigna como solución inicial de la próxima época, a la mejor solución obtenida en la época anterior.

Finalizado el recocido simulado, la mejor solución obtenida puede introducirse como solución inicial de una nueva ejecución del recocido

```
% Genero una solución inicial
s = fInitialState(n);
Mejor_Solucion = s;
% Genero la matriz OM
OM = fGenerateOMxy(dmat,n);
while Ejecuciones < NS

% Inicialización del Annealing
   s = Mejor_Solucion;
   L = round((n-2)/(1+Ejecuciones));
   Num_Intentos = 0;
   K = 10;

   while Iteraciones < NI
      OMs = fCalcOMVector(OM,s,n,L);
      Ps = fCalcPVector(OMs,n);
      sNew = fPCCPerturbate(s,n, Ps, L);

% Calculo el costo del estado actual y del vecino obtenido
      Costo_Actual = fCostCalculator(s,dmat,n);
      Costo_Nuevo = fCostCalculator(sNew,dmat,n);

% Evaluación de la perturbación
      if Costo_Nuevo <= Costo_Actual
         s = sNew;
         Num_Intentos = 0;
      else
% Aceptación del estado con cierta probabilidad y
disminución de la longitud del segmento
         if fEvaluate (Costo_Actual, Costo_Nuevo, Temp)
            s = sNew;
            Num_Intentos = 0;
         else
            Num_Intentos = Num_Intentos + 1;
         end

         if Num_Intentos > round(n*K/L)
            if L > 1
               L = L - round(log(n));
            end
            K = K + 1;
            CambiarEpoca = 1;
            Num_Intentos = 0;
         end
         if L <= 1
            break
         end
      end

% Guardado de la mejor solución
      [Mejor_Solucion, Costo_mejor_solucion] =
fSaveBest(s,dmat,n);

% Disminución de la temperatura
      Temp = fTempDecay(temp,alpha);
   end

% Evaluación de la mejor solución hasta el momento
   Costo_final = fCostCalculator(s,dmat,n);
   if  Costo_final >= Costo_mejor_solucion
      Temp = Temp_Inicial;
      Ejecuciones = Ejecuciones + 1;
   else
      Ejecuciones = 0;
      err_ant = err;
   end
```

Fig. 3. Código de la implementación de recocido simulado con preservación de caminos cortos

simulado que utilice una longitud L inicial según la Ecuación 13.

$$L = round(\frac{n-2}{1+Ejecuciones}) \quad (13)$$

donde las Ejecuciones representan la cantidad de veces que se repetirá el recocido simulado en pos de mejorar la solución final hasta una cantidad máxima de ejecuciones NS.

Resultados

Para ensayar el algoritmo de recocido simulado con PCC se lo compara con la implementación estándar del mismo. Para este análisis se usaron problemas de la librería TSPLIB. Los problemas de esta librería indican en su nombre el valor de la cantidad de ciudades o nodos n del problema. En particular, se ensayaron: gr17, gr21, gr24, fri26, att48, eil51, eil76, kroB100, kroB150, d198 y kroB200.

Las ejecuciones se ensayaron para los mismos parámetros del recocido simulado. En particular, se utilizó un esquema de enfriamiento según la Ecuación 5 con α=0,999, una cantidad máxima de cambios NC=10000 y la temperatura inicial para cada ejecución fue estimada según la Ecuación 9 en ambas implementaciones.

Se realizaron 10 ejecuciones consecutivas y como factor de calidad, se obtuvieron los errores promedios porcentuales (referidos a la mejor solución conocida). Además se calculó el tiempo de convergencia total y se obtuvo el desvío estándar de los resultados para cada problema.

Respecto del parámetro que requiere la implementación de SA con PCC, se seleccionó la cantidad de ejecuciones deseadas NS=5 de modo que la condición de finalización del algoritmo fue cuando la solución no mejoró luego de 5 ejecuciones.

Los resultados de dichas ejecuciones se pueden observar en las Figuras 4 a 7.

Se observan excelentes soluciones para los problemas ensayados. En particular, los problemas gr17, gr21, gr24 y fri26 se encontró la mejor solución conocida con PCC, en las 10 ejecuciones de cada problema. Los tiempos de ejecución en estos casos fueron variados ya que la componente aleatoria del algoritmo permite que a veces se explore rápidamente el espacio de soluciones y otras veces se deba realizar varias ejecuciones para mejorar la solución.

Respecto a los problemas de mayor complejidad, se observa que a medida que n aumenta, los tiempos de convergencia se incrementan de forma exponencial y la calidad de soluciones empeora alcanzando prácticamente el mismo desempeño que la implementación estándar para problemas de n=200.

A pesar de esto, el desvío estándar de los errores obtenidos (ver Figura 5) resulta menor en la implementación de SA con PCC.

Conclusiones

En este trabajo, se propuso una implementación de recocido simulado que utiliza un mecanismo de perturbación diferente a la implementación estándar del mismo. Esta implementación explota la idea de favorecer la preservación de caminos parciales de bajo costo durante la ejecución del algoritmo.

En rigor, el recocido simulado implementa el enfriamiento del sistema como un decrecimiento de la probabilidad de aceptación de soluciones peores a medida que explora el espacio de soluciones. Aceptar soluciones peores es una propiedad fundamental en la meta-heurística porque permite la exploración del espacio de soluciones.

Valiéndose de este concepto, la implementación propuesta pretende guiar la búsqueda de soluciones mediante la preservación de los caminos de menor costo a partir de una matriz de valores de mérito que caracteriza todas las conexiones entre nodos del problema.

Puede observarse que la implementación propuesta permite obtener mejores soluciones y con menor dispersión respecto de la implementación estándar del recocido simulado. Sin embargo, como contrapartida, el tiempo de ejecución es mayor.

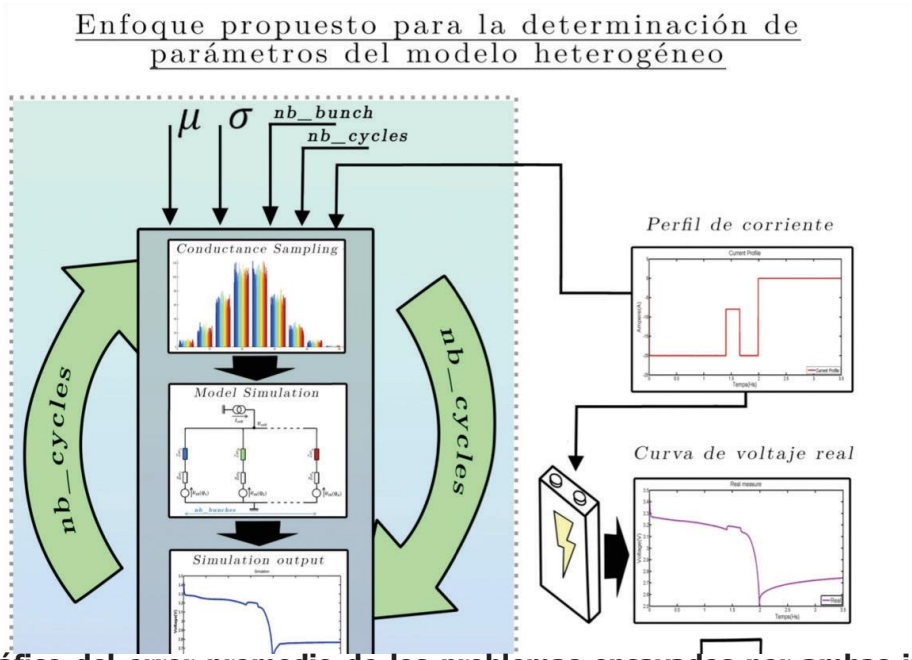

Fig. 4. Gráfico del error promedio de los problemas ensayados por ambas implementaciones. Las líneas punteadas corresponden con la aproximación polinómica de los errores promedio.

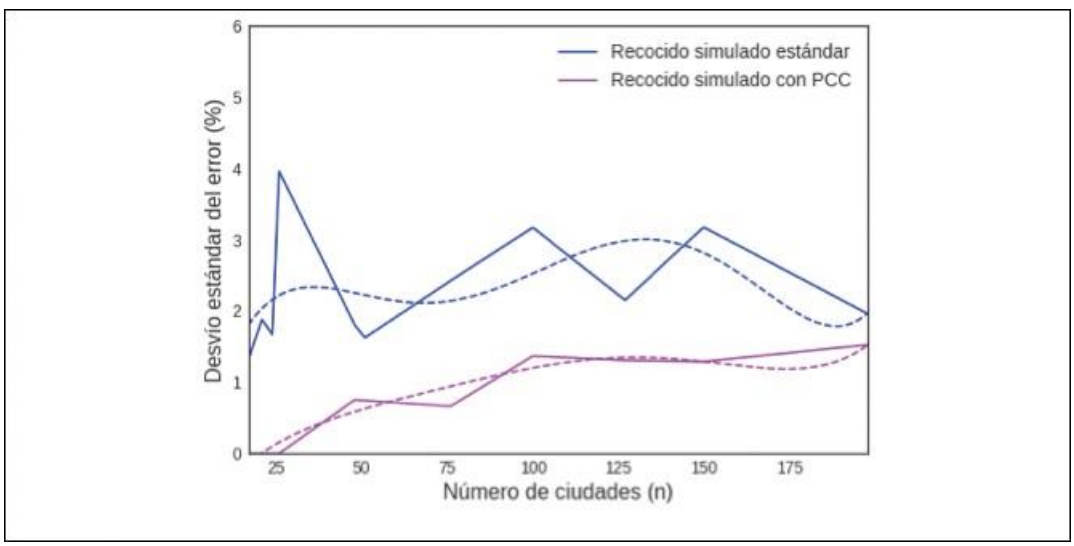

Fig. 5. Gráfico del desvío estándar del error obtenido de los problemas ensayados por ambas implementaciones. Las líneas punteadas corresponden con la aproximación polinómica de los mismos.

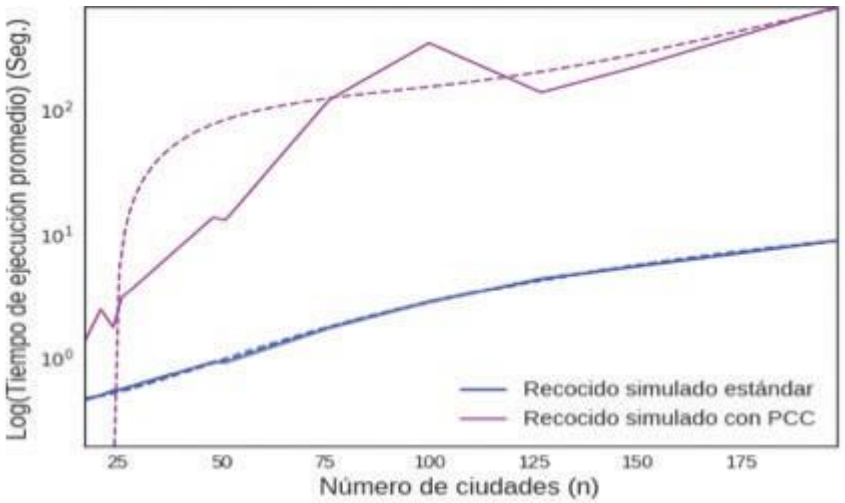

Fig. 6. Gráfico logarítmico de los tiempos de ejecución (en segundos) de los problemas ensayados por ambas implementaciones. Las líneas punteadas corresponden con la aproximación polinómica de los mismos.

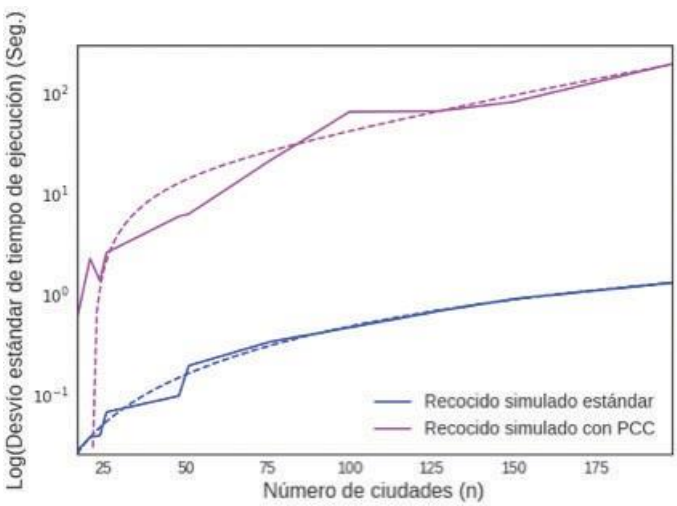

Fig. 7. Gráfico logarítmico del desvío estándar de los tiempos de ejecución (en segundos) de los problemas ensayados por ambas implementaciones. Las líneas punteadas corresponden con la aproximación polinómica de los mismos.

Referencias

ARENAS, M. G.; LAREDO, J. L.; CASTILLO, P. A.; SANCHEZ, P. G.; MORA, A. M.; PRIET A.; MERELO, J. (2010) "Statistical analysis of the parameters of the simulated annealing algorithm", IEEE Congress on Evolutionary Computation (CEC).

BEN-AMEUR, W. (2004) "Computing the Initial Temperature of Simulated Annealing", Computational Optimization and Applications, 29, 369–385.

DEUTSCH, C. V.; WEN, X. H. (1998) "An Improved Perturbation Mechanism for Simulated Annealing Simulation", Mathematical Geology, Vol. 30, No. 7.

HELD, M. and KARP, R.M. (1969) "The Traveling-Salesman Problem and Minimum Spanning Trees" pp 1138 - 1162 Operations Research

HONG, Y.; ZHIPING, L. (2006) "Speed-up simulated annealing by parallel coordinates", European Journal of Operational Research, Vol. 173, No. 1, pp. 59–71.

INGBER, L. (1989) "Very Fast Simulated Annealing", Math. Computational Modeling Vol. 12, No. 8, pp. 967-973.

LAWLER, E. L.; LENSTRA, J. K.; RINNOOY KAN A. H. G.; SHMOYS D. B. (1991) "The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization"

LUNDY, S.; MESS, A. (1986) "Convergence of an annealing algorithm", Mathematical programming 34(1): 111124.

METROPOLIS, N.; ROSENBLUTH, A.; ROSENBLUTH, M.; TELLER, A.; TELLER, E. (1953) "Equations of state calculations by fast computing machines". J. Chem. Phys., 21(6): 1087–1092.

MOHAMMAD, M. K. (2011) "Improved Simulated Annealing using Momentum", Second International Conference on Intelligent Systems, Modeling and Simulation.

RODRIGUEZ DIAZ, F. J.; GARCIA MARTINEZ, C.; LOZANO, M. (2010) "A GA-Based Multiple Simulated Annealing", IEEE Congress on Evolutionary Computation (CEC).

SANTA CHAVEZ, J. J.; PEÑUELA, C. A. M.; GRANADA, M. E. (2014) "Algoritmo de recocido simulado aplicado al problema de secuenciamiento regular", AVANCES Investigación en Ingeniería, Vol. 11, No. 1.

SCOTT ACTON, T.; BOVIK, A. C. (1996) "Generalized Deterministic Annealing", IEEE Transactions on neural networks, Vol. 1, No. 3.

SHAKOURI, H. G.; SHOJAEE, K.; BEHNAM, M. T. (2009) "Investigation on the choice of the initial temperature in the Simulated Annealing: A Mushy State SA for TSP", 17th Mediterranean Conference on Control & Automation.

SHOJAEE, K. G.; BEHNAM, M. T.; SHAKOURI, H. G.; REZAEI, M. (2010) "Enhancement of SA Algorithm by Intelligent Time Schedule", Proceedings of the 29th Chinese Control Conference.

SOHN, A. (1994) "Parallel Speculative Computation of Simulated Annealing", International Conference on Parallel Computing.

SOHN, A.; WU Z.; JIN, X. (1996) "Parallel Simulated Annealing by Generalized Speculative Computation", Annals of Operations Research, Vol. 63, No. 1, pp. 29–55.

STUCKMAN, B.; JETT, D. (1991) "A Bayesian Approach to Parameter Selection for Simulated Annealing", IEEE Congress on Decision Aiding for Complex Systems, Conference Proceedings.

TAGHI, M.; BAGHMISHEH, V.; NAVARBAF A. (2008) "A Modified Very Fast Simulated Annealing Algorithm", International Symposium on Telecommunications, Conference Proceedings.

TSPLIB https://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/ (última visita mayo de 2017)

VERRASTRO C. (1991) "Hacia un solución polinómica del problema del viajante" II Simposio de Inteligencia Artificial y Robótica" UNLu, 51-59