Resolución del problema del viajante de comercio mediante métodos de inteligencia artificial

Juan Ignacio Medved, Tomás Bautista Ordóñez

Campus Universitario, Buenos Aires, Argentina
Dto Electrónica UTN FRBA
jmedved@frba.utn.edu.ar
toordonez@frba.utn.edu.ar

***Resumen*— Se proponen dos alternativas para la resolución del problema del viajante de comercio basadas en métodos de búsqueda heurísticos (A*) y probabilísticos (Recocido simulado).**

***Palabras Clave*— TSP, A Estrella, Recocido Simulado, Python**

I. INTRODUCCIÓN

El problema del viajante de comercio o TSP por sus siglas en inglés (*travelling salesman problem*) es un problema de tipo NP-Hard dentro de la optimización combinatoria, muy importante en investigación operativa y en ciencias de la computación. Tiene diversas aplicaciones en el campo de la planificación, logística y fabricación de circuitos electrónicos [1][2].

El TSP consiste en la problemática de encontrar el camino con menor costo que une un número determinado de ciudades sin repetirlas, sin omitir ninguna y volviendo a la ciudad de origen.

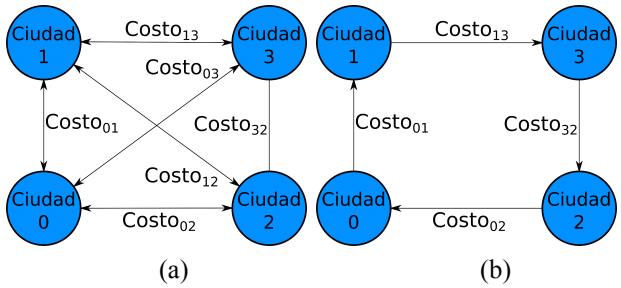

Fig. 1 Representación gráfica del escenario que plantea el problema (a), camino óptimo (b).

Debido a la simplicidad de su enunciado y reglas resulta de gran interés para el desarrollo y/o verificación de algoritmos de búsqueda de distintos tipos [3].

En este trabajo se realiza la resolución del TSP utilizando el algoritmo "A Estrella" y "Recocido simulado" (SA, por sus siglas en inglés) [4][5] para mapas de diferentes características.

A. *Descripción del algoritmo A estrella*

El algoritmo A estrella es un método de búsqueda informado heurístico ampliamente utilizado en problemas de grafos. Se destaca por su precisión (alcanza el resultado óptimo) y eficiencia (utiliza una estimación para acelerar el proceso).

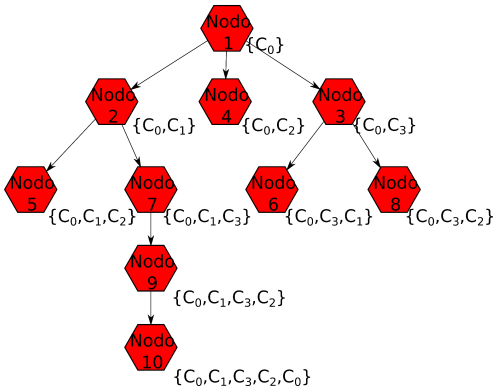

Fig. 2 Representación típica en grafos de algoritmos A aplicados en TSP.

El método consiste en la exploración de un árbol de nodos, como el de la Fig. 2. En el caso del TSP se le llama nodo a un camino válido entre ciudades. Los caminos válidos contienen ciudades sin repetir, comienzan en la ciudad de origen y en el caso de que sea un camino completo (nodo meta) debe finalizar con la ciudad de origen, ver nodo 10 de Fig. 2.

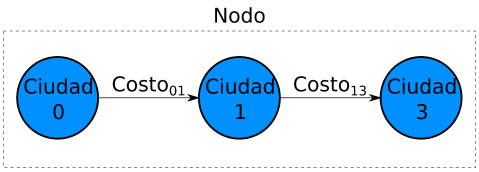

Fig. 3 Nodo no meta.

Cada nodo tiene un padre, a excepción del primero, y por lo menos un hijo, a excepción de los nodos meta. Los nodos hijos contienen el mismo camino que su padre con el agregado de una ciudad al final del mismo.

Los nodos meta son el último nodo de cada rama del árbol (como el nodo 10 de la Fig. 2), estos contienen una solución al problema, es decir tienen un camino completo. El fin del algoritmo es encontrar el nodo meta de menor

costo, o sea el camino óptimo.

Cada nodo tiene un costo total *f(i)* (donde *i* es el número de nodo) compuesto por dos partes. La primera, *g(i)*, representa el costo de ir de la primera a la última ciudad del nodo a través del camino que el mismo propone. La segunda, *h(i)*, es una estimación de $h^*(i)$ que equivale al costo óptimo restante para llegar a la ciudad meta a través de un camino válido.

$$f(i) = g(i) + h(i) \quad (1)$$

Estas dos componentes tienen efectos opuestos en la evolución del algoritmo, la función *g(i)* prioriza la exploración de nodos de menor costo acumulado, esto involucra un comportamiento de apertura horizontal en el árbol de búsquedas, ralentizando el algoritmo y exigiendo mayor almacenamiento de datos. Por otro lado, la función *h(i)* prioriza los nodos más cercanos al objetivo, haciendo que el árbol se explore verticalmente.

Para garantizar la solución óptima la función h(i) debe ser una heurística admisible, esto es que siempre sea menor o igual al costo óptimo restante, ergo, que no sobreestime. Este comportamiento se representa en la Ec. 2. A este requisito se lo conoce como condición minorante.

$$h(i) \leq h^*(i) \quad (2)$$

Por otro lado, si la heurística es demasiado mala, o sea una subestimación grosera (caso extremo h(i)=0), la componente de apertura en anchura *g(i)* será predominante, haciendo que la complejidad y almacenamiento utilizado crezca de manera exponencial. De esta manera se puede concluir que la mejor estimación es la mayor subestimación que se pueda conseguir [5].

El algoritmo hace uso de dos listas de nodos llamadas "abierta" y "cerrada". En la lista abierta se encuentran nodos que no han sido explorados, mientras que en la lista cerrada ocurre lo opuesto.

En el inicio la lista abierta contiene únicamente el nodo inicial (compuesto solo por la ciudad de origen) y la lista cerrada se encuentra vacía.

Su funcionamiento se basa en un lazo de iteración en el que se extrae de la lista abierta al nodo de menor costo total *f(i)*, en caso de que sea un nodo meta, se termina la búsqueda con éxito y si no lo es, se mueve el nodo a la lista cerrada y se agregan todos sus hijos a la lista abierta.

Este proceso continúa hasta encontrar un nodo meta o que la lista abierta quede vacía, en cuyo caso se sale del algoritmo con código de error.

Muchas de las formas de hacer más eficiente a este método se centran en insertar nodos a la lista abierta de manera eficiente. Este trabajo propone utilizar una estimación grosera del costo óptimo (obtenida de otro método de búsqueda no óptimo pero veloz) y utilizarla como costo máximo a partir del cual los nodos hijos se descartan y no se agregan a la lista abierta, esto toma el nombre de condición mayorante.

B. Descripción del algoritmo Recocido simulado

SA es un algoritmo probabilístico inspirado en el proceso de recocido del acero que consiste en calentar y luego enfriar lentamente el material. El calor causa que los átomos aumenten su energía y puedan desplazarse de sus posiciones iniciales, mientras que un enfriamiento lento aumenta las probabilidades de finalizar el proceso en una configuración con menor energía que la inicial.

El objetivo del algoritmo es encontrar una buena aproximación al valor óptimo de una función en un espacio de búsqueda grande, evitando caer en un mínimo local de la función.

Se parte de una solución inicial (SI) cualquiera y en cada iteración del algoritmo se provoca una perturbación aleatoria en dicha solución que es evaluada (SE) y aceptada en el caso de ser superadora a la inicial. En el caso contrario, la nueva solución es aceptada con una probabilidad dependiente de la temperatura (T) y la diferencia de costo (ΔE = SI - SE) según la ecuación:

$$e^{\frac{-\Delta E}{T}} \quad (3)$$

En cada iteración se disminuye la temperatura del sistema, disminuyendo la probabilidad de aceptación de la nueva solución.

La perturbación eventualmente puede destruir una muy buena solución, por lo que para evitar esta pérdida eventual se puede resguardar el costo mínimo global obtenido durante toda la simulación y finalmente informar dicho resultado.

II. RESOLUCIÓN CON A ESTRELLA

Los costos entre ciudades pueden representarse en formato matricial, donde cada ciudad representa una fila y una columna.

Las características de los mapas utilizados en este trabajo hacen que las matrices sean simétricas (el camino A→B es igual al B→A) y con diagonal principal nula.

TABLA I
MATRIZ DE COSTOS G

	Costos				
Ciudades	c_0	c_1	c_2	···	c_{n-1}
c_0	0	$g_{0,1}$	$g_{0,2}$	···	$g_{0,n-1}$
c_1	$g_{1,0}$	0	$g_{1,2}$	···	$g_{1,n-1}$
c_2	$g_{2,0}$	$g_{2,1}$	0	···	$g_{2,n-1}$
⋮	⋮	⋮	⋮	⋮	⋮
c_{n-1}	$g_{n-1,0}$	$g_{n-1,1}$	$g_{n-1,2}$	···	0

Tal como se mencionó anteriormente la característica

principal de este algoritmo es la heurística, los mapas utilizados para probar este algoritmo no cumplían con la geometría euclidiana, por lo que la propuesta de estas funciones tiene cierta dificultad. Sin embargo se plantearon tres alternativas (Ec. 4, Ec. 5 y Ec. 6), todas ellas tienen la propiedad de ser minorantes.

$$h_0 = 0 \tag{4}$$

$$h_3 = min\{G\}(n + 1 - n_{recorridas}) \tag{5}$$

$$h_4 = min\{G_n\}(n + 1 - n_{recorridas}) \tag{6}$$

En todas las ecuaciones n representa la cantidad de ciudades que tiene el mapa, $n_{recorridas}$ la cantidad de ciudades recorridas que tiene el nodo bajo análisis, *G* es la matriz de costos entre ciudades y G_n es la matriz de costos entre ciudades sin recorrer, es decir, es la matriz *G* cuyas filas y columnas correspondientes a la ciudades recorridas han sido eliminadas.

La Ec. 4 representa una estrategia "abrir primero"[5] haciendo que se exploren muchos caminos, la Ec. 5 plantea un escenario demasiado ideal en el que el camino pendiente está compuesto por costos iguales al mínimo costo existente en la matriz. La Ec. 6 surge como una mejora ya que contempla las ciudades recorridas, evitando repetir costos ya utilizados.

Como se mencionó en la descripción del algoritmo, una mejora a este es incorporar una restricción en la etapa de almacenamiento de nodos en la lista abierta. Este "umbral" puede ser el costo de un nodo meta encontrado por otro método de búsqueda o un nodo meta aleatorio válido.

Esto reduce la cantidad de nodos en la lista abierta y por consiguiente el tiempo que conlleva el ordenamiento de la misma y el almacenamiento en memoria.

El valor más simple que se puede conseguir es el costo del camino de ciudades ordenadas alfabéticamente (diagonal adyacente a la diagonal principal de la matriz de costos), es decir,

$$e_0 = \sum_{j=0}^{n-2} g_{j,j+1} + g_{n-1,0} \tag{7}$$

El segundo umbral utilizado se obtiene como resultado del algoritmo recocido simulado,

$$e_1 = SA\{G\} \tag{8}$$

El cuarto y quinto ensayo involucra estas mejoras en combinación con la heurística h_4.

III. RESOLUCIÓN CON RECOCIDO SIMULADO

Para probar el algoritmo SA se utilizó la base de datos TSPLIB [6] que contiene distintos escenarios para probar el TSP y problemas relacionados. En este trabajo se utilizaron los mapas con distancia euclidiana en 2 dimensiones, con un número de ciudades entre 48 y 200.

A. Parámetros Iniciales

Los parámetros iniciales para la resolución del algoritmo son la temperatura inicial (TI), el factor de disminución de la temperatura (F), el número de iteraciones (NI) máximas del programa, la cantidad de iteraciones a igual temperatura (IIT) y el número de iteraciones sin cambios (NC) utilizado como variable de salida.

Para estimar una temperatura inicial, se realizó una corrida del algoritmo con una cantidad de iteraciones k

$$k = n^2 \tag{9}$$

Donde n es la cantidad de ciudades del problema y se obtuvo el promedio de la diferencia de costo. Finalmente se elige una probabilidad de aceptación inicial del 50% y de la expresión

$$P_{inicial} = e^{\frac{-\Delta E}{T_{inicial}}} \tag{10}$$

Se despeja la variable de temperatura inicial, resultando en la siguiente ecuación:

$$T_{inicial} = \frac{-\Delta E}{ln\,(P_{inicial})} \tag{11}$$

La simulación termina una vez que se alcanza el número máximo de iteraciones,NI, o si no ocurren cambios en el costo del camino en un determinado número de ciclos y se iguala la variable NC.

B. Funciones de Perturbación

Para la función de perturbación se realizaron pruebas con 2 funciones diferentes. La primera función utilizada fue la simple permutación entre 2 ciudades de la lista.

La segunda función utilizada fue la inversión. Dicha función fue implementada mediante una ventana de longitud aleatoria que provoca una reversión de bloque, generando una nueva solución.

La diferencia que se observa es que mientras la función de permutación potencialmente realiza un cambio de 4 caminos, la función de inversión solamente genera un cambio de 2 caminos ya que es un mapa reversible.

C. Perfiles de Temperatura

En el caso de la disminución de temperatura se utilizó la siguiente función:

$$T_n = \alpha \ * T_{n-1} \tag{12}$$

Donde el valor α utilizado para las simulaciones fue de

0,995 logrando un enfriamiento lo suficientemente lento para garantizar mayor exploración de las soluciones del problema, teniendo en cuenta que los tiempos de ejecución no sean excesivamente altos.

Asimismo, para evaluar el impacto de realizar iteraciones a la misma temperatura, se incorporó el parámetro IIT.

IV. RESULTADOS

A. Resultados de A Estrella

Realizando una comparativa de las distintas opciones de heurísticas y estimaciones planteadas, en la tabla II y tabla III, se observa una clara superioridad de la Ec. 5 y Ec. 6, los tiempos de ejecución y nodos abiertos son reducidos considerablemente a comparación de no usar heurística, es decir la Ec. 4.

TABLA II
TIEMPOS DE EJECUCIÓN

	Tiempo				
Mapa	h_0	h_3	h_4	h_4 y e_0	h_4 y e_1
1	1,31 ms	202 us	1,41 ms	918 us	1,08 ms
2	2,59 ms	506 us	2,41 ms	2,53 ms	2,17 ms
3	6,81 ms	1,09 ms	5 ms	4,42 ms	4,8 ms
4	15,9 ms	10 ms	33,6 ms	33,9 ms	32,6 ms
5	1,5 h	1,2 m	41,8 s	53,5 s	4,41 s
6	617 ms	93,3 ms	145 ms	144 ms	90,6 ms
7	3 m	1,26 ms	5,71 ms	5,06 ms	5,08 ms
8	5,2 h	3,5 h	3,9 m	4 m	39,8 s
9	1,88 s	1,15 ms	5,89 ms	5,28 ms	7,44 ms
10	4,56 m	2,35 s	293 ms	400 ms	174 ms
11	N/A	3,12 ms	16,6 ms	20,4 ms	15,0 ms
12	N/A	64,8 ms	17,4 ms	18,4 ms	14,4 ms

TABLA III
NODOS ABIERTOS DURANTE EJECUCIÓN

	Nodos abiertos				
Mapa	h_0	h_3	h_4	h_4 y e_0	h_4 y e_1
1	22	8	8	8	8
2	49	13	14	14	14
3	89	34	34	34	34
4	327	208	327	327	327
5	65034	9187	6607	6607	6607
6	1608	553	406	406	406
7	18583	19	20	20	20
8	348923	239175	27410	27410	27410
9	2483	19	20	20	20
10	24693	2693	572	572	572
11	N/A	29	30	30	30
12	N/A	29	30	30	30

La mejora por condición mayorante con recocido simulado se hace notable al observar el tiempo de ejecución y el almacenamiento utilizado a lo largo del programa (Fig. 4, Fig. 5 y Fig. 6).

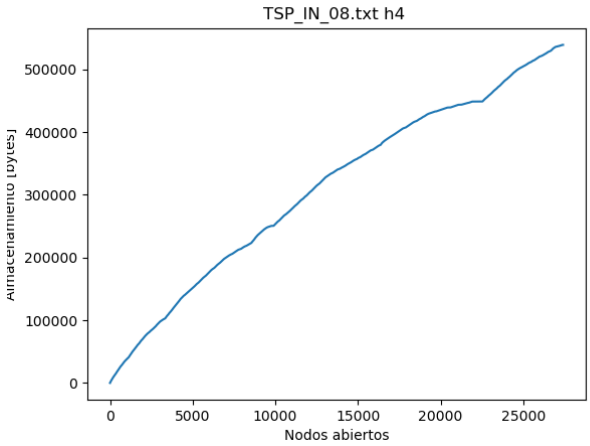

Fig. 4 Almacenamiento utilizado a lo largo del mapa 8, con heurística tipo 4 y sin usar umbral, el pico de almacenamiento ronda los 488 kB y es monótonamente creciente.

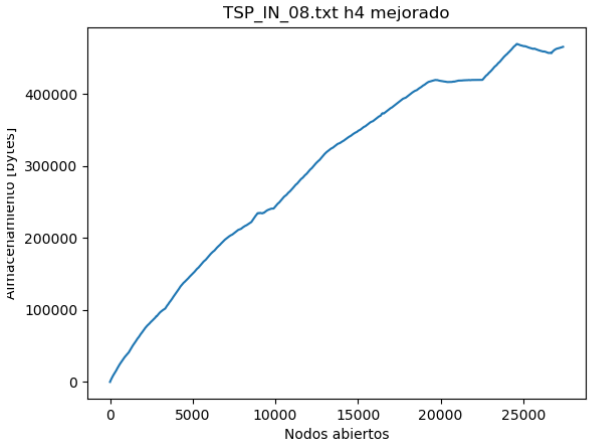

Fig. 5 Almacenamiento utilizado a lo largo del mapa 8, con heurística tipo 4 y usando umbral e_0, el pico de almacenamiento ronda los 390 kB y la curva presenta algunos asentamientos (misma velocidad de extracción e inserción de nodos a lista abierta).

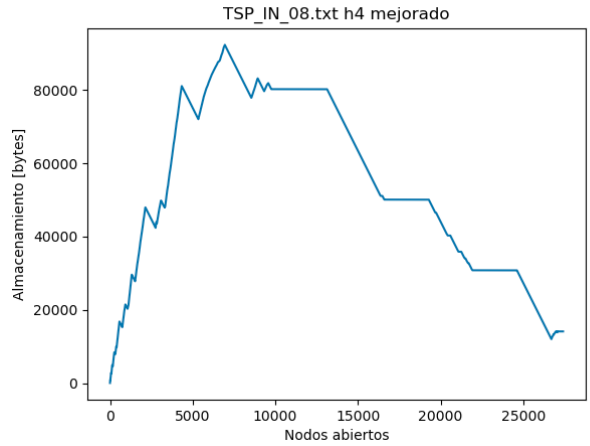

Fig. 6 Almacenamiento utilizado a lo largo del mapa 8, con heurística tipo 4 y usando umbral e_1, el pico de almacenamiento ronda los 78 kB y además la curva presenta grandes periodos de decrecimiento (extracción de

nodos a mayor velocidad que la inserción).

B. Resultados de Recocido Simulado

En la Tabla 4 se representa el costo total obtenido y la diferencia porcentual con el camino óptimo para cada uno de los mapas con la función de perturbación "Permutación" e "Inversión" .

TABLA IV
COSTO FINAL DE LAS DIFERENTES FUNCIONES DE PERTURBACIÓN

	Costo total final		**Costo final porcentual**		
Mapa	Permutación	Inversión	Permutación	Inversión	**Costo Óptimo**
berlin52.tsp	7988	7544	5,92 %	0,0265 %	7542
kroB100.tsp	25692	23045	16,04 %	3,22 %	22141
kroB150.tsp	33050	27015	26,48 %	3,68 %	26130
kroB200.tsp	43862	31545	57,96 %	5,11 %	29437

Se observa que al incrementar el número de ciudades aumenta la diferencia con el camino óptimo y que la inversión logra mejores resultados.

En la Fig.7 se muestra la evolución del costo en función de las iteraciones y en la Fig.8 y Fig.9 la representación gráfica de las ciudades. Estos resultados se obtuvieron utilizando la función de inversión.

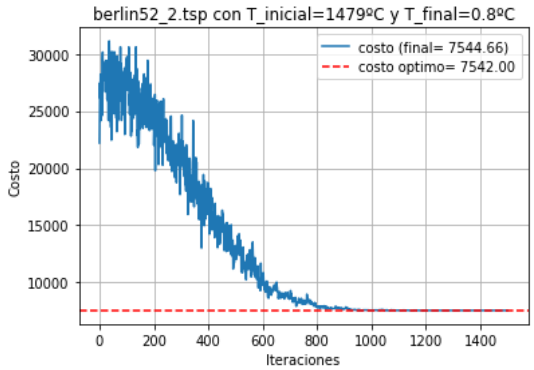

Fig. 7 Costo en función de las iteraciones en azul y camino óptimo del mapa berlin52.tsp.

Como se advierte en la imagen, al principio de la simulación la temperatura es mayor, por lo tanto la probabilidad de aceptación de un camino con costo superior aumenta. Al incrementar el número de iteraciones, la temperatura disminuye y consigo dicha probabilidad, logrando menores variaciones en el costo del camino y tendiendo hacia un mínimo local.

En la Fig.8 se muestra el camino inicial del mapa berlin52.tsp representado con la línea azul y cada una de las ciudades con puntos rojos, mientras que en la Fig.9 presenta el camino final resuelto con la función inversión. A simple vista se observa que en la última figura el camino no contiene cruces ni sub-bucles cerrados que aumentan el costo del camino total.

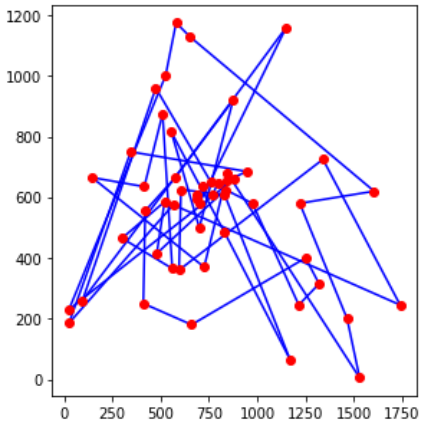

Fig. 8 En los ejes se encuentran las coordenadas, los puntos rojos representan las ciudades y el camino inicial se encuentra en azul.

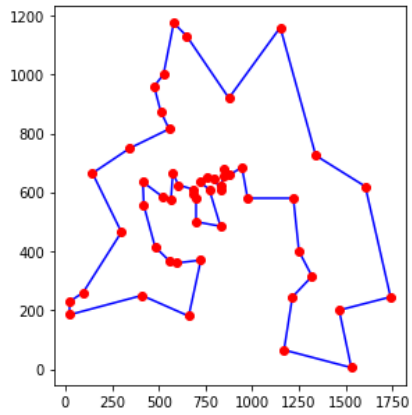

Fig. 9 En los ejes se encuentran las coordenadas, los puntos rojos representan las ciudades y el camino final resuelto con la función inversión se encuentra en azul.

C. Resultados con Condición Mayorante

Los resultados obtenidos con el SA (e_1) para los mapas previamente resueltos con A Estrella se pueden observar en la Tabla V, estos valores son el promedio de 10 corridas y posteriormente son utilizados como condición mayorante para acelerar el algoritmo.

TABLA V
RESULTADOS OBTENIDOS CON EL SA

	Resultado obtenido		
Mapa	e_0	e_1	Costo óptimo
1	30	30	30
2	73	35	35
3	32	31	31
4	30	30	30
5	251	83	83
6	170	76	76
7	10	10	10
8	183	143	143
9	10	10	10
10	53	17	17
11	55	15	15
12	30	15	15

IV. CONCLUSIONES

Para el caso del SA, la función de perturbación cumple un rol central en el algoritmo, obteniéndose resultados claramente superiores con el uso de la función inversión. Esto se debe a que dicha función realiza potencialmente menor cantidad de cambios en el camino que la función propuesta restante y por lo tanto, menor aleatoriedad en las perturbaciones.

Los resultados del SA son probabilísticos y difieren para un mismo problema con iguales parámetros de entrada, por lo tanto es necesario realizar numerosas simulaciones para evaluar la eficiencia del algoritmo.

El SA no asegura encontrar el camino óptimo pero se lo utiliza en problemas donde la elevada cantidad de ciudades no permite la implementación de otros métodos.

Los resultados obtenidos fueron similares al utilizar el parámetro IIT o una mayor temperatura inicial e iteraciones totales, por lo tanto este parámetro no se considera relevante.

El algoritmo A estrella encuentra el resultado óptimo a costa de mucho procesamiento, por lo tanto queda limitado a problemas simples. Además requiere un gran conocimiento del problema para poder proponer una buena heurística.

RECONOCIMIENTOS

Agradecimientos a Juan Balbi y Federico Bua por colaborar en el inicio del trabajo.

REFERENCIAS

[1] Duman, E., & Or, I. (2004). Precedence constrained TSP arising in printed circuit board assembly. International Journal of Production Research, 42(1), 67–78.

[2] Punnen A.P. (2007) The Traveling Salesman Problem: Applications, Formulations and Variations. In: Gutin G., Punnen A.P. (eds) The Traveling Salesman Problem and Its Variations. Combinatorial Optimization, vol 12. Springer, Boston, MA.

[3] Matai, R., Singh, S. P., & Mittal, M. L. (2010). Traveling salesman problem: an overview of applications, formulations, and solution approaches. Traveling salesman problem, theory and applications, 1.

[4] Inteligencia Artificial un Enfoque Moderno S. J. Russell y P. Norvig, , 2nd ed., Pearson Educación, Madrid, España (2004).

[5] Principles of Artificial Intelligence N. J. Nilsson, Morgan Kaufmann Publishers, Palo Alto, California, U.S. (1980).

[6] Gerhard Reinelt. Universität Heidelberg. Institut für Angewandte Mathematik.
http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsplib.html