ABORDAJE DEL PROBLEMA TSP UTILIZANDO EL ALGORITMO ÁVIDO CODICIOSO

[1]Crysthel Aparicio, QA Analyst en ICONIC,[2]Richardson Cárcamo, estudiante de Ing. En Sistemas Computacionales
[3]César Figueroa, Jefe de Proyecto en Agile Solution
[1]*Universidad Tecnológica Centroamericana (UNITEC), Honduras,*
crysthel.aparicio@unitec.edu, alejandro.carcamo@unitec.edu
[2]*Universidad Tecnológica Centroamericana (UNITEC), Honduras, cesar.figueroa@unitec.edu*

Abstract–. Se aborda el problema del agente viajero o TSP (por sus siglas en inglés "Travelling Salesperson Problem") que busca en su versión de optimazación dar como resultado la ruta más óptima en costo basándose en el peso de las aristas o caminos, visitando todas las ciudades destinadas una tan sola vez. Se abordó con la solución con el heurístico del algoritmo codicioso, el cual basa en la regla del vecino más cercano, se posiciona sobre una ciudad, luego ve cuál de las ciudades adyacentes están más cercanas o tienen menos costos, y la añade a la trayectoria si esta no ha sido visitada; funciona de esa manera hasta regresar a la ciudad de origen.

Keywords—TSP, NP-Completo, Heuristico, Complejidad, Algoritmo Codicioso

I. INTRODUCCIÓN

En el mundo de las ciencias de la computación surgió una duda común: ¿Qué hace a algunos problemas computacionalmente más complejos y a otros más sencillos? En resultado de una búsqueda exhaustiva de tratar de responder a dicha interrogante nace la *Teoría de la Complejidad* Ref. [1] que tiene como finalidad la creación de múltiples herramientas y mecanismos capaces de modelar y analizar tanto la complejidad intrínseca a un problema como la complejidad del algoritmo que resuelve el mismo

La complejidad algorítmica que se aborda en este documento es del problema del agente viajero o TSP (por sus siglas en inglés "Travelling Salesperson Problem"), que modela un viaje de negocios, que busca recorrer todas las ciudades pasando una sola vez por cada una, al menor costo posible y poder regresar a la ciudad de origen del viaje.

O más formalmente hablando, el problema TSP Ref. [2], se define como el problema del agente viajero, dado un entero n > 0 y las distancias entre cada par de las n ciudades, estás distancias se da por medio de la matriz (d_{ij}) de adyacencia de dimensión n x n, donde d_{ij} es un entero mayor o igual a cero. Un recorrida, tour o ruta, es una trayectoria que visita todas las ciudades exactamente una vez. El problema consiste en encontrar un recorrido con longitud total mínima.

Una permutación π cíclica representa un recorrido, si se interpreta π(j) como la ciudad después de la ciudad j, j = 1, 2, …n. Así el costo del recorrido es:

$$\sum_{j=1}^{n} d_{j\pi(j)}.$$

Aunque parece un asunto relativamente sencillo, el problema tiene una complejidad alta. En TSP se presentan N ciudades, por las cuales existen N! rutas posibles, por lo cual, sabemos que TSP es un problema combinatorio (factorial) y por lo cual es NP-Duro. Además, TSP ref. [4] posee una versión de Decisión que no busca la solución más óptima, sino que, dado el costo y un número x, decide si existe o no una ruta de viaje con menor o igual costo que x; la solución a la versión de Decisión del TSP es comprobable en tiempo polinómico, ya que solo debemos validar la existencia y costo de las aristas resultantes; entonces, podemos concluir que TSP en su versión de decisión en un problema NP-Completo.

Versiones del problema del TSP

N	TSP	
	Problema	Clasificación
	TSP de decisión: Dado un Grafo G con pesos y un valor K verificar si existe una trayectoria T' dentro del grafo G tal que el peso total de T´ sea igual o menor a K.	NP-Complete.
	TSP de Optimización: Dado un Grafo G, encontrar la trayectoria hamiltoniana más óptima que exista en G.	NP-Duro.

Dada la categoría de la complejidad del TSP y ya que este problema es común, se han desarrollado múltiples heurísticos y/o aproximaciones, las cuales; aunque aparentemente o probablemente den buenas soluciones, no darán con el óptimo, pero, su complejidad es menor y, por lo tanto, el tiempo de ejecución es considerablemente mucho más corto que el tiempo que toma encontrar el óptimo.

Ahora, surge la interrogante, ref. [5] de si en verdad vale la pena utilizar estos heurísticos o aproximaciones, si no dan el óptimo. Como se mencionó, el TSP independientemente si es versión de Optimización o Decisión, es NP-Duro, una complejidad algorítmica O(n!). Cuando hablamos de viajes relativamente cortos como la visita a 7 ciudades (7! - 1 =

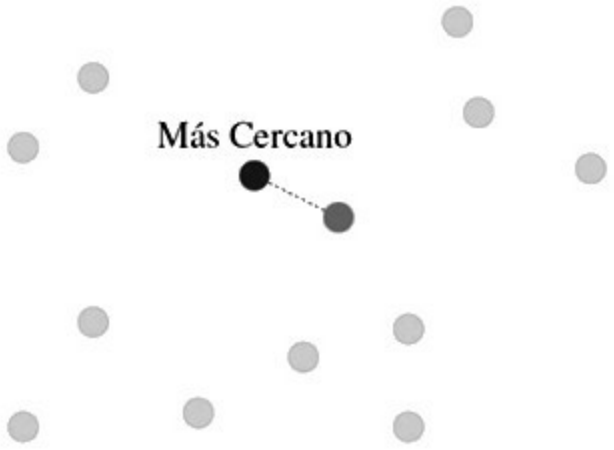
5,039), una cantidad de rutas que la computadora puede evaluar sin problemas; pero, qué pasa si le añadimos una tan sola ciudad más: 8 ciudades (8! - 1 = 40,319), se puede observar que al aumentar en una unidad las ciudades a visitar, las rutas posibles pasaron de 5,039 a 40,319, añadir una ciudad incluye una cantidad abismal (exponencial) de rutas posibles más ¿Qué pasa si se habla de giras mundiales, como las recurrentes por los equipos de fútbol o bandas musicales? Ejecutar la versión de la optimización pura, tardará una cantidad exageradamente desmesurada de tiempo en encontrar la ruta óptima definitiva. Aunque, al ejecutar una aproximación o heurístico, no tenemos el óptimo, la mayoría de las ocasiones sus trayectorias resultantes son aceptables.

II. METODOLOGÍA

Como se mencionó, la alternativa que se implementó fue el heurístico del TSP llamado algoritmo codicioso ("Greedy Algorith", su nombre en inglés) o del vecino más cercano ("Nearest Neighbor Rule", NNR), la manera en que funciona para calcular un camino que visite todas las ciudades a lo sumo una vez y vuelva a la ciudad de origen (la cual llamaremos v1) es la siguiente:

1- Empieza por la ciudad v1 buscando la ciudad cercana (vn) a la misma, la encuentra y la añade a la trayectoria t.
2- Luego, se posiciona cada vez en la última ciudad añadida a la trayectoria y busca su respectiva ciudad más cercana (vn), luego valida que esta no se encuentre ya en la trayectoria y la añade, si no es así, busca de nuevo hasta encontrar la más próxima que no se encuentre añadida a la trayectoria ya y la añade. Repite el proceso hasta que haya visitado todas las ciudades respectivas.
3- Una vez en la última ciudad, vuelve a vn desde la respectiva ciudad.

Básicamente el algoritmo codicioso construye un ciclo Halmitoniano de bajo costo basándose en un vértice cercano a uno dado.



Se escribió un programa en el lenguaje de programación JAVA de Oracle, con la versión 8 actualización 111, con una interfaz de usuario relativamente sencilla para el usuario generada en su mayoría con el IDE Netbeans 8.2. Se utilizaron grafos implementados con una matriz de adyacencia que se genera a partir de los datos ingresados por el usuario en una mini interfaz gráfica de coordenadas en X y Y, que simulan un mapa con las locaciones de las ciudades a visitar, de tamaño 10 x 10. Cabe mencionar que, por razones de recursos (ya que la versión de optimación esta implementada de manera recursiva), se limitó el programa a un máximo de 9 ciudades (9! – 1 = 362,879), ya que, con cantidades mayores se tratan con millones de rutas disponibles (10! = 3,628,800; 11! = 39,916,800); además, 9 ciudades representan un largo viaje de negocios en la realidad, así que tanto objetivamente como subjetivamente es una cantidad considerable de rutas para el objetivo del proyecto.

El programa cuenta con tres clases (más un main.java, que su única labor es ejecutar las clases y cargar un pequeño menú con GUI) las cuales son:
1- TSP_AdjacencyMatrixBoard: clase encargada de ejecutar la simulación en la mini interfaz gráfica de cómo se calculó la trayectoria resultante.
2- TSP_GraphAgent: clase encargada de calcular la trayectoria resultante, tanto en versión de optimización como el algoritmo heurístico codicioso.
3- TSP_GraphMap: clase que genera mini interfaz gráfica donde el usuario ingresa la ubicación de las ciudades.

Se utiliza un hilo para ejecutar la simulación de evaluación de rutas del TSP de optimación y el algoritmo heurístico codicioso.

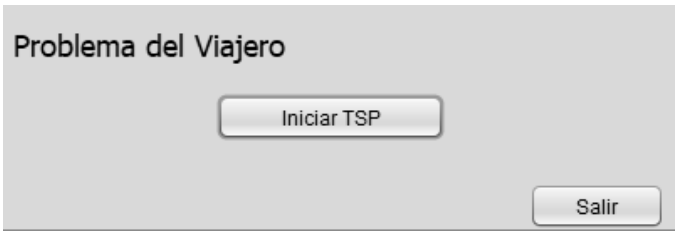


Ilustración 1 Menú Principal

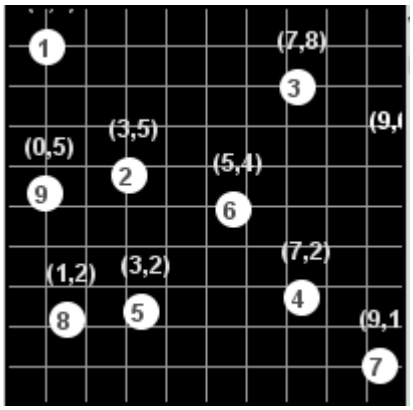


Ilustración 2 Mini interfaz gráfica de coordenadas

Ilustración 3 Matriz de adyacencia generada a partir de la mini interfaz gráfica

Respecto a las pruebas, con ambos algoritmos, se realizaron con rutas desde 3 hasta 9 ciudades, en todas las simulaciones se varió la ciudad de origen de 1 – N. Solo con el algoritmo heurístico codicioso de probó con cantidades de ciudades mayores a 20. Cabe mencionar que, dichas pruebas se intentaron realizar en el ambiente más favorable posible en el momento para medir los tiempos: únicamente el proceso del programa estaba activo. Se utilizó una computadora portátil Acer Aspire V5 con procesador Core i3-2375M 1.50 GHz, 4 GB RAM, un Windows 10 Enterprise 64 bits, recién reiniciada.

III. Resultados

A continuación, se presentarán los resultados obtenidos de las pruebas realizadas tanto en el TSP optimizado como en el algoritmo heurístico codicioso, con la cantidad de ciudades de: 5, 7 y 9, en el orden respectivo. Por cada uno se presentará: la tabla de adyacencia utilizada (y generada aleatoriamente en la simulación de un territorio cuadrado de 10 x 10) que mostrará las distancias de las aristas entre cada ciudad; y la tabla de resultados que contiene el vértice de inicio de la trayectoria, el tiempo de ejecución del algoritmo respectivo, la distancia total de la trayectoria y la trayectoria misma; por último, se hará una comparación de ambas tablas de resultados. Para simplicidad de las salidas de los resultados, se escogieron números sucesivos del 1 – 9 para representar las ciudades.

Es importante mencionar que, en las tablas de resultado del TSP Óptimo: los tiempos de ejecución van disminuyendo considerablemente, eso es debido a que la versión del algoritmo esta implementada de manera recursiva, y como las pruebas se hicieron de manera secuencial con las mismas tablas de adyacencia, queda un cierto historial sobre las trayectorias, así que ya en las pruebas posteriores ignora esas rutas y ahorra una cantidad considerable de tiempo. Así que, el tiempo más significativo a considerar es el de la primera trayectoria calculada; ya que, es la cual ejecuta todo el proceso.

La complejidad del algoritmo ávido codicioso es $O(n^2)$, ya que se recorre la matriz de adyacencia por completo, para verificar el vecino más cercano de cada ciudad.

A. Cinco Ciudades

Tabla 1 **Matriz de adyacencia 5 ciudades**

Matriz de Adyacencia					
Ciudad	1	2	3	4	5
1	0.00	7.28	7.00	9.22	4.00
2	7.28	0.00	8.60	4.00	3.61
3	7.00	8.60	0.00	7.07	8.06
4	9.22	4.00	7.07	0.00	6.71
5	4.00	3.61	8.06	6.71	0.00

Tabla de Resultados TSP Óptimo			
Vértice Inicio	Ruta	Distancia	Tiempo de Ejecución (ms)
1	152431	25.68	8
2	251342	25.68	1
3	342513	25.68	0
4	431524	25.68	0
5	524315	25.68	0

Tabla 2 Resultados TPS Óptimo 5 ciudades

Tabla de Resultados Algoritmo Codicioso			
Vértice Inicio	Ruta	Distancia	Tiempo de Ejecución (ms)
1	152431	25.68	0
2	251342	25.68	0
3	315243	25.68	1
4	425134	25.68	0
5	524315	25.68	0

Tabla 3 Resultados Algoritmo Codicioso 5 ciudades

Según logramos ver en los datos resultantes de las pruebas, con cinco ciudades el algoritmo heurístico codicioso concuerda con cada ruta óptima del TSP versión de optimización; pero sus tiempos de ejecución son menores, llegando a apenas un milisegundo por cálculo de trayectoria; a diferencia de 8 ms que tarda el TPS óptimo.

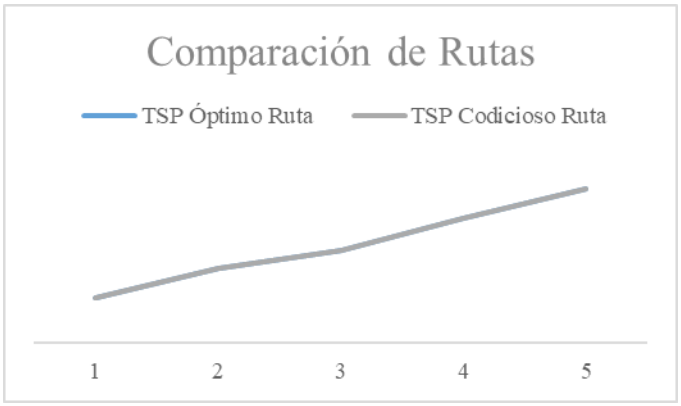


Figura 1 Gráfico comparativo de diferencia de rutas 5 ciudades

B. Cinco Ciudades

Matriz de Adyacencia							
Ciudad	1	2	3	4	5	6	7
1	0.00	5.00	7.21	5.39	4.47	8.60	7.00
2	5.00	0.00	4.12	5.83	2.24	7.00	8.60
3	7.21	4.12	0.00	4.12	2.83	3.16	6.71
4	5.39	5.83	4.12	0.00	3.61	3.61	2.83
5	4.47	2.24	2.83	3.61	0.00	5.10	6.40
6	8.60	7.00	3.16	3.61	5.10	0.00	5.00
7	7.00	8.60	6.71	2.83	6.40	5.00	0.00

Tabla 4 Matriz de adyacencia 7 ciudades

Tabla de Resultados TSP Óptimo			
Vértice Inicio	Ruta	Distancia	Tiempo de Ejecución (ms)
1	12536741	26.44	159
2	25367412	26.44	102
3	35214763	26.44	22
4	41253674	26.44	37
5	53674125	26.44	13
6	63521476	26.44	8
7	76352147	26.44	11

Tabla 5 Resultados TPS Óptimo 7 ciudades

Tabla de Resultados Algoritmo Codicioso			
Vértice Inicio	Ruta	Distancia	Tiempo de Ejecución (ms)
1	15236471	27.43	0
2	25364712	26.66	0
3	35214763	26.44	0
4	47635214	26.44	0
5	52364715	27.43	0
6	63521476	26.44	1
7	74523617	31.56	0

Tabla 6 Resultados Algoritmo Codicioso 7 ciudades

Podemos observar (vea Tabla 5 y Tabla 6) que añadiendo dos ciudades más al problema, los resultados entre el algoritmo óptimo y el heurístico empiezan variar, aunque, la diferencia es a lo mucho 1 (a diferencia de la última ruta que difirió con el óptimo en alrededor de 5), los resultados del heurístico siguen siguen siendo buenos, ya están bajo un margen de error aceptables.

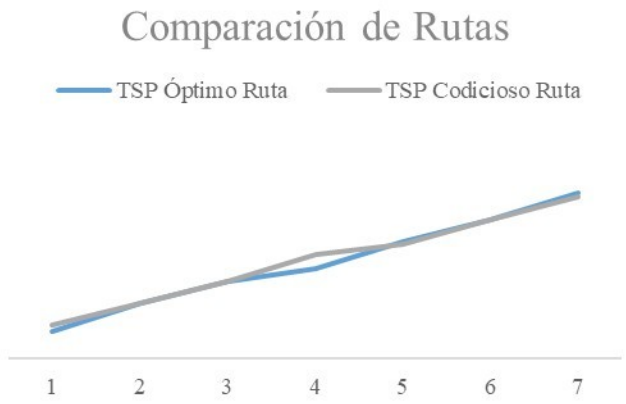


Figura 2 Gráfico comparativo de diferencia de rutas 7 ciudades

C. Cinco Ciudades

Matriz de Adyacencia									
Ciudad	1	2	3	4	5	6	7	8	9
1	0.00	7.28	8.54	4.12	7.81	7.00	4.12	9.43	5.66
2	7.28	0.00	7.21	6.32	3.16	8.60	3.16	6.32	3.61
3	8.54	7.21	0.00	4.47	4.24	3.16	7.07	2.00	4.12
4	4.12	6.32	4.47	0.00	5.10	3.16	4.24	5.66	3.00
5	7.81	3.16	4.24	5.10	0.00	6.32	4.47	3.16	2.24
6	7.00	8.60	3.16	3.16	6.32	0.00	7.21	5.10	5.00
7	4.12	3.16	7.07	4.24	4.47	7.21	0.00	7.07	3.00
8	9.43	6.32	2.00	5.66	3.16	5.10	7.07	0.00	4.12
9	5.66	3.61	4.12	3.00	2.24	5.00	3.00	4.12	0.00

Tabla 7 Matriz de adyacencia 9 ciudades

Tabla de Resultados TSP Óptimo			
Vértice Inicio	Ruta	Distancia	Tiempo de Ejecución (ms)
1	1729583641	28.74	4295
2	2958364172	28.74	3025
3	3859271463	28.74	2703
4	4172958364	28.74	2276
5	5927146385	28.74	1938
6	6417295836	28.74	1541
7	7295836417	28.74	1242
8	8592714638	28.74	869
9	9583641729	28.74	396

Tabla 8 Resultados TPS Óptimo 9 ciudades

Tabla de Resultados Algoritmo Codicioso			
Vértice Inicio	Ruta	Distancia	Tiempo de Ejecución (ms)
1	1495273861	36.85	0
2	2594638712	35.20	1
3	3859461723	35.06	1
4	4952716384	33.50	0
5	5946382715	34.98	0
6	6385941726	33.57	0
7	7952463817	36.60	0
8	8364952718	33.44	0
9	9527146389	29.25	1

Tabla 9 Resultados Algoritmo Codicioso 9 ciudades

Seguimos y añadimos otras dos ciudades más (ver Tabla 7). Ya a este punto empezamos a ver una diferencia notable en la variación de las rutas resultantes del algoritmo heurístico codicioso (ver Tabla 9) con las rutas óptima del TSP Óptimo (ver Tabla 8). En este punto, el error crece, calculando el promedio tenemos que:

((36.85 - 28.74) + (35.20 - 28.74) + (35.06 - 28.74) + (33.50 - 28.74) + (34.98 - 28.74) + (33.57 - 28.74) + (36.60 - 28.74) + (33.44 - 28.74) + (29.25 - 28.74)) / 9

El resultado a lo anterior nos da un de error promedio de 6.05. Dependiendo del margen de error aceptable, pero vemos que entre más ciudades agregamos.

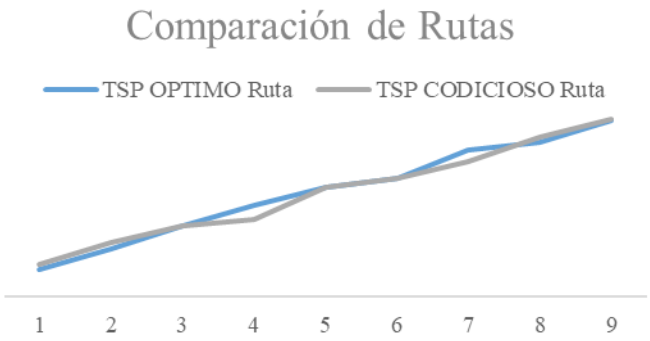


Figura 3 Gráfico comparativo de diferencia de rutas 9 ciudades

IV. CONCLUSIONES

El problema TSP realmente es muy cotidiano alrededor de todo el mundo, en diferentes áreas. Su aplicabilidad va desde el enunciado del mismo TSP hasta tour de equipos de fútbol, o desde una simple salida vacacional en familia hasta giras mundiales de bandas o cálculo de planes de tour para las agencias de viaje locales o internacionales; entre otras varias aplicaciones tanto como aplicar el TSP adaptado o en conjunto con otro problema para modelar una situación. Pero, debido a que encontrar su solución óptima es NP-Dura y la verificación de la misma no polinomial, se vuelve ref. [6] un problema intratable con un número considerable de nodos (solo con 11 nodos tenemos un total de 39,916,799 rutas distintas) si no es abordado con heurísticos o aproximaciones. En el caso del algoritmo codicioso, es simple de entender y de implementar, su debilidad según lo observado es que, aunque para trayectorias cortas de TSP se pueden obtener buenas soluciones, en situaciones que involucraban más de 7 ciudades se obtuvieron trayectorias muy diferentes a la óptima; esto debido a que, como su visión es miope (a corto alcance) busca la mejor solución al alcance, decisión que lo puede llevar a obligarlo a escoger entre solo rutas caras. En promedio el algoritmo heurístico codicioso ref. [7] retorna una trayectoria de un 25% más largo/caro que el óptimo, aunque, se menciona en algunos casos especiales retorna no solo no el óptimo, si no, el peor de los casos.

Por tales razones, podemos concluir que para cálculos de trayectorias no tan grandes (no mayor a 20 ciudades) y que se cuente con un margen de error de al menos del 25%, se puede tomar en cuenta el algoritmo heurístico codicioso para resolver el problema TSP.

Además, cabe mencionar que hay otros heurísticos o aproximaciones que son mejores para resolver el problema del TSP, pero la dificultad de implementación es mayor que del heurístico tratado.

REFERENCIAS BIBLIOGRAFICAS

[1] Bellman, R. (1960), «Combinatorial Processes and Dynamic Programming», Bellman, R., Hall, M., Jr. (eds.), ed., Combinatorial Analysis, Proceedings of Symposia in Applied Mathematics 10, American Mathematical Society, pp. 217-249..

[2] Applegate, D. L.; Bixby, R. M.; Chvátal, V.; Cook, W. J. (2006), The Traveling Salesman Problem, ISBN 0-691-12993-2.M. King, B. Zhu, and S. Tang, "Optimal path planning," *Mobile Robots*, vol. 8, no. 2, pp. 520-531, March 2001.

[3] Christofides, N. (1976), Worst-case analysis of a new heuristic for the travelling salesman problem, Technical Report 388, Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburgh..

[4] Papadimitriou, Christos H. (1977), «The Euclidean traveling salesman problem is NP-complete», Theoretical Computer Science 4 (3): 237-244, MR 0455550, doi:10.1016/0304-3975(77)90012-3.J.-G. Lu, "Title of paper with only the first word capitalized," *J. Name Stand. Abbrev.,* in press.

[5] Bellman, R. (1962), «Dynamic Programming Treatment of the Travelling Salesman Problem», J. Assoc. Comput. Mach. 9: 61-63, doi:10.1145/321105.321111.

[6] De los Cobos, S.; Goddard, J.; Gutiérrez, M.; Martínez, A. (2010) Búsqueda y Exploración Estocástica. Universidad Autónoma Metropolitana, México D.F.

[7] Gutin, G.; Yeo, A.; Zverovich, A. (2002), «Traveling salesman should not be greedy: domination analysis of greedy-type heuristics for the TSP», Discrete Applied Mathematics 117 (1–3): 81-86, doi:10.1016/S0166-218X(01)00195-0