COMPARISON BETWEEN THE ALGORITHMS OF SIMULATED ANNEALING, ANT COLONY AND GENETIC ALGORITHMS FOR THE RESOLUTION OF THE MULTIPLE TRAVELING SALESMAN PROBLEM

COMPARACIÓN DE LOS ALGORITMOS DE OPTIMIZACIÓN DE TEMPLE SIMULADO, COLONIA DE HORMIGAS Y ALGORITMOS GENÉTICOS PARA LA RESOLUCIÓN DEL PROBLEMA DE MÚLTIPLES AGENTES VIAJEROS

Carlos D. Quintana

Universidad de Pamplona
Semillero de Investigación en Ingeniería de Sistemas. Grupo CICOM
Ciudadela Universitaria. Pamplona, Norte de Santander, Colombia.
Tel.: 57-7-5685303, Fax: 57-7-5685303, Ext. 144
E-mail: carlos.quintana@unipamplona.edu.co

Abstract: This article presents the results of an exploration made in the field of combinatorial optimization, from different solution perspectives, tackling the multiple travelling salesman problem, based on an extended version of the classical TSP.

Keywords: mTSP, combinatorial optimization, heuristics.

Resumen: En este artículo se presentan los resultados de una exploración en el campo de la optimización combinatoria, tratando el problema de múltiples agentes viajeros, una extensión del clásico problema del agente viajero, desde múltiples perspectivas de solución.

Palabras clave: mTSP, optimización combinatoria, heurísticas.

1. INTRODUCION

El problema del Agente Viajero (Travelling Salesman Problem) ha sido uno de los problemas de optimización combinatoria que más ha dado de qué hablar en el estudio de problemas difíciles de resolver, dada su aparente simpleza y su gran rango de aplicaciones en problemas comunes, y al abordar el problema de múltiples agentes viajeros (mTSP) como una extensión de este podemos hacer uso de la extensa librería de investigación realizada sobre ésta, especialmente en las formas de resolver el problema empleando algoritmos heurísticos de optimización combinatoria. En éste documento se busca tratar con tres de los algoritmos más comunes y comparar su rendimiento al afrontar éste problema.

2. EL PROBLEMA DE LA OPTIMIZACIÓN

Al momento de solucionar un problema, podemos encontrarnos con el obstáculo de que sin importar el algoritmo utilizado, encontrar una solución óptima requiere de una cantidad significativa de recursos computacionales. La teoría de la complejidad computacional clasifica éste tipo de problemas como “inherentemente difíciles”, o de tipo NP, si encontrar ésta solución requiere de una cantidad de tiempo no polinómica.
Esto hace que ciertos problemas sean imposibles de abordar con técnicas algorítmicas tradicionales, pues la cantidad de recursos computacionales necesarios para lograrlo pueden llegar a ser inconcebibles, y es por esto que se han desarrollado técnicas alternativas aproximadas que, empleando heurísticas, permiten aproximar soluciones que resulten lo

suficientemente útiles en la práctica (Pearl, 1984).

2.1 El problema de múltiples agentes viajeros

Un problema de rutas es todo aquel problema de optimización donde se debe encontrar una ruta óptima para satisfacer las demandas de un conjunto de clientes. La relevancia que tiene el estudio de estos problemas es, en gran medida, debido a la gran utilidad que tienen en situaciones reales, principalmente en la logística y distribución de mercancías.
Existen muchos problemas de rutas en función de las restricciones adicionales que impongamos (número de comerciantes, capacidad de los vehículos...) y hay que diferenciarlos de los problemas de buscar el camino óptimo, ya que en los de rutas, establecemos un conjunto de arcos o nodos que debemos visitar, y cuando buscamos el camino óptimo, sólo tratamos de llegar a un destino desde un origen de la forma más eficiente, sin importar los nodos y arcos intermedios. (Peñalva 2015).
El problema del agente viajero simple puede definirse sobre un grafo completo $G = (V, A)$ donde *V* es un conjunto de *n* vértices, y *A* es un conjunto de arcos o aristas. A cada arco $(i, j) \in A$ se le asocia una distancia o coste c_{ij}, componiendo así una matriz *C* asociada a *A*. El problema del agente viajero consiste en determinar el circuito de menor distancia que visite cada vértice una única vez. Adicionalmente, se dice que un TSP es simétrico cuando consideramos que $c_{ij} = c_{ji} \; \forall \, i, j \in A$.
Una generalización del mismo es el mTSP, que consiste en determinar un conjunto de rutas para *m* agentes viajeros, que inicien y regresen a un mismo nodo denominado *depósito*, de manera que todos los demás nodos sean visitados una única vez por alguno de los agentes.
El mTSP puede representarse en forma de TSP si consideramos un grafo extendido de *n+m-1* nodos, donde cada uno de los agentes realiza un circuito y regresa al nodo original.
De ésta manera logramos transformar la complejidad de un mTSP a la de un TSP clásico, lo que nos permite aplicar algoritmos de optimización pensados para el segundo.
Éste problema es sumamente dinámico, pues podemos generar una gran cantidad de variantes al alterar estas simples restricciones, sea al cambiar el número de depósitos y la capacidad de albergar una cantidad de agentes en cada una, o al añadir restricciones adicionales de ventanas de tiempo o nodos mínimos y máximos a visitar por agente. Así mismo, el mTSP tiene una estrecha conexión con un problema más general de enrutamiento de vehículos (*Vehicle Routing Problem)*, donde no se tienen en cuenta las restricciones de capacidad (Bektas 2005).

2.2 Heurísticas de Optimización

Encontrar una solución exacta de un problema de enrutamiento como el TSP extendido implica evaluar todas las posibles permutaciones del orden de los nodos a visitar, lo que resulta en una complejidad de O(n!), que incluso con un número relativamente pequeño de nodos (n=10) resulta impráctico evaluar sus millones de posibilidades, mientras que, por otro lado, un algoritmo heurístico logra encontrar soluciones aproximadas a la óptima en un número de operaciones mucho más reducido, que para efectos prácticos tiene mayor valor.
Es muy extensa la lista de algoritmos desarrollados y perfeccionados para resolver el problema del agente viajero, y como propósito de éste documento se estudiarán tres de estos, considerados especialmente efectivos dentro del ámbito de las heurísticas, sin llegar a ser extremadamente complejos.

2.2.1 Algoritmo de Temple Simulado

También conocido como *Simulated Annealing* en inglés, y desarrollado por Kirkpatrick et al (1983) y Černý (1985), consiste en realizar una escalada estocástica, inspirada en el enfriamiento controlado de metales, en la cual se calienta un metal a alta temperatura y se enfría progresivamente de manera controlada, lo que se conoce con el nombre de *cristalización* o *templado* de metales. En este proceso, si el enfriamiento ha sido el adecuado, se alcanza una estructura de menor energía que la original, que toma el nombre de mínimo global.
En este método se elige un sucesor de entre todos los posibles según una distribución de probabilidad. En dicha elección se permite escoger probabilísticamente estados peores. De este modo, se dan pasos parcialmente aleatorios por el espacio de soluciones buscando la mejor solución. Otra de las características de este método es que la probabilidad de que un estado peor sea aceptado varía en función del incremento producido en la función objetivo. Esto permite al algoritmo poder salir de óptimos locales. (Rabanal 2010).
A grandes rasgos podemos definir el algoritmo de temple simulado de acuerdo al siguiente pseudo código:

```
Seleccionar solución inicial s0
Seleccionar temp. inicial t0 > 0
Repetir
  Repetir
    Seleccionar un s∈Ss0
    d ← f(s) – f(s0)
```

```
    Si (d < 0) ó (U(0,1) < exp(-d/t))
    s0 ← s
  Hasta un número de repeticiones
  t ← a(t)
Hasta condición de parada
Solución: la mejor de todas las s0
encontradas
```

Donde identificamos los elementos:

- Temperatura inicial t_0.
- Proceso de enfriamiento a(t),
- Espacio de soluciones S,
- Función de coste f(s),
- Solución inicial s_0. (Moreno et al 2007).

2.2.2 Algoritmo de Optimización por Colonia de Hormigas

La optimización por colonia de hormigas (conocida como *Ant Colony Optimization*, *ACO* en inglés), fue introducida por Marco Dorigo en los inicios de la década de los 90 como herramienta para la solución de problemas de optimización complejos (Dorigo, 1996).
Ésta es inspirada por el comportamiento real de las hormigas, pues estos insectos cuando están en búsqueda de la comida exploran inicialmente el área cercana a su nido de una forma aleatoria. Tan pronto encuentran fuentes de alimentos, evalúan su cantidad y calidad, y llevan alguna parte de esta comida para su nido. Durante el regreso al nido, las hormigas depositan una sustancia química llamada *feromona* sobre el camino, la cual servirá de guía futura para que las demás encuentren los alimentos. La cantidad de dicha sustancia depositada dependerá de la cantidad y calidad de los alimentos.
Esta característica es ampliamente utilizada para la solución de problemas de optimización que necesitan mejorar sustancialmente los tiempos de cómputo para la solución de una aplicación específica. (Algarín 2010).
Desde un punto de vista general, los algoritmos ACO se pueden aplicar a un amplio conjunto de problemas de optimización. Especialmente, este tipo de algoritmos son muy utilizados en problemas de rutas debido a su alto grado de adaptación al problema. Su empleo para la resolución se debe a que son muy complejos de resolver óptimamente mediante métodos exactos (muy costosos en cuanto a términos de tiempos de ejecución de programa).
A veces, aunque el algoritmo resulte de gran ayuda y, en muchas ocasiones, nos aporta soluciones muy próximas a la óptima, incluso la óptima, en otros casos no lo hace. Es decir, no siempre se encuentra la solución óptima empleando estas técnicas. (Revuelta 2015).

El algoritmo de Optimización por Colonia de Hormigas puede expresarse mediante el siguiente pseudocódigo:

1. Inicializar la población de hormigas.
2. Construir una ruta (Hormiga caminando a través del grafo).
3. Evaluar la ruta elaborada.
4. Actualizar la matriz de feromonas en la ruta recorrida.
5. Repetir los pasos 2-4 con cada una de las hormigas.
6. Recuperar la mejor ruta encontrada.

2.2.3 Algoritmos Genéticos

Los Algoritmos Genéticos como métodos adaptativos pueden usarse para resolver problemas de búsqueda y optimización. Están basados en el proceso genético de los organismos vivos, cuyas poblaciones evolucionan en la naturaleza de acorde con los principios de la selección natural y la supervivencia de los más fuertes, postulados por Darwin (1859).
Basados en este principio, los Algoritmos Genéticos son capaces de ir creando soluciones para problemas del mundo real. En la naturaleza los individuos de una población compiten entre sí en la búsqueda de recursos tales como comida, agua y refugio. Incluso los miembros de una misma especie compiten a menudo en la búsqueda de un compañero. Aquellos individuos que tienen más éxito en sobrevivir y en atraer compañeros tienen mayor probabilidad de generar un gran número de descendientes. Por el contrario individuos poco dotados producirán un menor número de descendientes. Esto significa que los genes de los individuos mejor adaptados se propagaran en sucesivas generaciones hacia un número de individuos creciente.
La combinación de buenas características provenientes de diferentes ancestros, puede a veces producir descendientes “superindividuos”, cuya adaptación es mucho mayor que la de cualquiera de sus ancestros.
De esta manera, las especies evolucionan logrando unas características cada vez mejor adaptadas al entorno en el que viven. Los Algoritmos Genéticos usan una analogía directa con el comportamiento natural. Trabajan con una población de individuos, cada uno de los cuales representa una solución factible a un problema dado. A cada individuo se le asigna un valor o puntuación, relacionado con la bondad de dicha solución. En la naturaleza esto equivaldría al grado de efectividad de un organismo para competir por unos determinados recursos. Cuanto mayor sea la adaptación de un individuo al problema, mayor ser ‘a la probabilidad de que el mismo sea seleccionado para reproducirse, cruzando su material genético con otro individuo seleccionado de igual forma. Este cruce

producirá nuevos individuos – descendientes de los anteriores – los cuales comparten algunas de las características de sus padres. Cuanto menor sea la adaptación de un individuo, menor ser 'a la probabilidad de que dicho individuo sea seleccionado para la reproducción, y por tanto de que su material genético se propague en sucesivas generaciones." (Inza et al. 2008).

Con los algoritmos genéticos se busca codificar las variables de decisión de un problema de búsqueda en cadenas finitas que puedan representar soluciones candidatas, las cuales la teoría trata como los *cromosomas*, cuyo alfabeto hace el papel de los *genes*, y los valores de estos a los *alelos*.

Por ejemplo, para el problema del Agente Viajero se considera un cromosoma a una ruta válida y una ciudad representa un gen.

Uno de los conceptos fundamentales de los Algoritmos Genéticos es el de la *población* de soluciones candidatas de la que éstos dependen. El tamaño de ésta población es usualmente especificado de manera manual, y lo hemos de considerar como un factor importante en el rendimiento y la escalabilidad de los algoritmos. Esto es, que poblaciones muy pequeñas pueden influenciar una convergencia prematura, de forma que el espacio de soluciones no se está explorando eficientemente. Y por otro lado, poblaciones muy grandes conllevan un desperdicio de recursos computacionales en cálculos redundantes. (Sastry et al 2005).

La estructura general de un Algoritmo Genético clásico consiste en:

1. Inicializar la población de soluciones candidatas.
2. Evaluar el rendimiento de la población.
3. Seleccionar los mejores candidatos.
4. Cruzar éstos candidatos para generar nuevos individuos con características comunes.
5. Aplicar un operador de mutación en los nuevos candidatos.
6. Reemplazar la vieja población con los individuos generados.
7. Repetir los pasos 2-6 hasta cumplir una condición terminal.
8. Recuperar la mejor ruta encontrada.

En el caso de los problemas de enrutamiento como el TSP y el mTSP, es de resaltar que los operadores de cruce y de mutación son de especial interés, pues dada la naturaleza de los problemas, éstos deben cumplir con las restricciones que presente el problema, de forma que puedan generar soluciones válidas. Para propósitos de éste experimento se empleará el operador de *Order-Based Crossover* (Sastry et al).

3. METODOLOGÍA

3.1. Extensión del TSP a mTSP

Se ha demostrado que el problema de múltiples agentes viajeros (mTSP) puede representarse en forma de un TSP, donde los recorridos de cada agente distinto se transforma en un solo gran recorrido (Junjie & Dingwei, Bellmore & Hong, Sveska & Huckfeldt).

Éste método consiste en llevar cuenta de unos nodos especiales acoplados al grafo original, los cuales representarán una vuelta al nodo origen o depósito, y marcarán el inicio del siguiente recorrido, como se ilustra en la Figura 1. Para propósitos prácticos todos estos nuevos nodos, o *nodos origen*, representan el mismo depósito o nodo inicial indistintamente, pero hacemos ésta extensión para poder aplicar los métodos heurísticos tradicionales con las mínimas modificaciones.

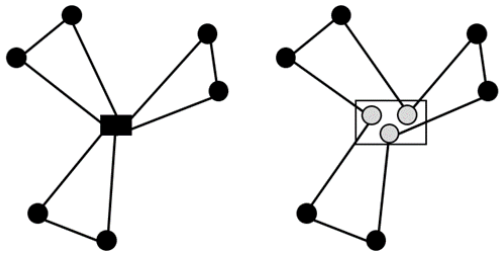

Fig. 1. Representación de un mTSP como un TSP (m=3).

Es claro que al realizar ésta extensión estamos cambiando la complejidad de las soluciones, a razón de n+m-1.

Al implementar este modelo estamos modificando una de las restricciones originales del TSP, la cual impide que de un mismo nodo surjan un número distinto a dos caminos, pues todos los *nodos origen* añadidos artificialmente representan en últimas el mismo nodo depósito. Ésta situación se aplica únicamente a éste.

Además de esto, debemos tener en cuenta que éste modelo puede presentar problemas en cuanto a la generación de recorridos inválidos, debido a disposiciones inesperadas entre los nodos.

Dentro de éstas se incluye el que un agente despache de uno de los *nodos origen*, visite exactamente un *nodo externo* e intente volver al origen a través de un *nodo origen* distinto al del que partió, efectivamente haciendo que un agente atraviese la misma arista dos veces, lo que representa una violación a las restricciones del TSP (Figura 2).

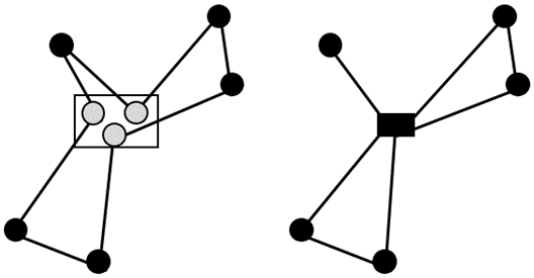

Fig. 2. Enrutamiento inválido al realizar la transformación de TSP a mTSP (m=3).

Asimismo es importante impedir que dos *nodos origen* se conecten entre sí, pues esto eliminaría uno de los agentes finales.

Estas nuevas restricciones no son triviales para los algoritmos de optimización heurísticos, pues a menos que realicemos serias modificaciones, éstos no distinguirán un recorrido inválido para éste modelo. Para un acercamiento inicial, aplicaremos los algoritmos originales y realizaremos una validación adicional después de cada operación mayor. Esto puede resultar en serias afectaciones al rendimiento de la soluciones de darse el caso que no puedan encontrarse soluciones factibles de forma consistente, pero después de pruebas experimentales encontramos una proporción de recorridos descartados lo suficientemente aceptable.

4. RESULTADOS

Para realizar las comparaciones emplearemos un conjunto de datos estándar para el estudio de éste tipo de problemas de enrutamiento, la TSPLIB. Los algoritmos han sido desarrollados en el lenguaje Java y ejecutados en una máquina Core i5 a 1.90 GHz (4 GB RAM).

Se aplicarán los algoritmos a 4 problemas distintos con un número distinto de agentes, denotando la distancia total de recorrido de la mejor solución encontrada después de 20 intentos distintos, así como la distancia promedio entre éstos.

Al momento de ser realizado este proyecto no se contaba con los resultados óptimos de cada uno de los problemas, por lo que se compararán sus rendimientos entre sí para encontrar el más efectivo.

Los resultados de estas pruebas se muestran en las Tablas 1 y 2.

Número de agentes m=2						
	SA		ACO		GA	
	mejor	avg	mejor	avg	mejor	avg
ulysses22*	7165	7296	7528	7792	11155	11504
tiempo (ms)	1234		841		861	
berlin52	8376	9064	9494	9494	21852	22517
tiempo (ms)	2125		4693		2229	
pr76	135708	145492	183944	198008	462917	465825
tiempo (ms)	3173		8803		4330	
pr152	163952	191808	274565	295206	895132	897755
tiempo (ms)	7124		36712		18138	

Tabla 1. Resultados de la ejecución de los algoritmos empleando dos agentes.
**Las distancias de ulysses22 son calculadas utilizando el método de coordenadas geográficas. Los demás escenarios emplean coordenadas euclidianas.*

Número de agentes m=5						
	SA		ACO		GA	
	mejor	avg	mejor	avg	mejor	avg
berlin52	9149	9716	11907	12055	23211	23265
tiempo (ms)	2454		4909		2619	
pr76	149324	163320	217760	223170	500091	503788
tiempo (ms)	3251		12794		4680	
pr152	174609	200955	302807	315855	906486	919067
tiempo (ms)	7284		45751		19826	

Tabla 2. Resultados de la ejecución de los algoritmos empleando cinco agentes.

5. CONCLUSIONES

Aunque el problema de extensión del TSP como mTSP no sea aplicable a muchas situaciones reales, resultó un terreno de comparación aceptable para estos algoritmos de optimización.

Experimentalmente encontramos que los mejores resultados fueron presentados por el algoritmo de Enfriamiento Simulado, seguidos del algoritmo de Optimización por Colonia de Hormigas. Sospechamos que esto se debe a la cantidad considerablemente mayor de variables que presenta éste último, y consideramos que tiene posibilidad de mejoramiento en el mismo contexto de éste experimento si se encontrasen valores más apropiados e, incluso, únicos para cada problema.

Los algoritmos genéticos son un campo sumamente amplio, y únicamente la búsqueda de un operador de cruce y mutación especializado para un problema de permutación con las restricciones propias de la extensión del TSP como mTSP, o de cualquier otra extensión del TSP como el VRP, presenta un reto digno de investigación, pues no es un tema trivial, y la falta de operadores apropiados afectó en gran

medida el rendimiento del algoritmo en esta investigación

REFERENCIAS

Abdulkarim, H. A., & Alshammari, I. F. (2015). *Comparison of Algorithms for Solving Traveling Salesman Problem*. German Academic Exchange Service International Journal of Engineering and Advanced Technology (IJEAT), 4(6), 75-79.

Algarín, C. A. R. (2010). *Optimización por colonia de hormigas: aplicaciones y tendencias*. Ingeniería solidaria, 6(10-11), 83-89.

Bektas, T. (2006). *The multiple traveling salesman problem: an overview of formulations and solution procedures*. Omega, 34(3), 209-219.

Bellmore, M., & Hong, S. (1974). Transformation of multisalesman problem to the standard traveling salesman problem. Journal of the ACM (JACM), 21(3), 500-504.

Černý, V. (1985). Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. Journal of optimization theory and applications, 45(1), 41-51.

Díaz, P. M., Fernández-Toribio, G. H., Allende, J. S., & Manso, A. G. (2007). *Metaheurísticas de optimización combinatoria: uso de simulated annealing para un problema de calendarización*. Tecnología y desarrollo, 5, 20.

Dorigo, M., & Di Caro, G. (1999, July). *Ant colony optimization: a new meta-heuristic*. In Proceedings of the 1999 congress on evolutionary computation-CEC99 (Cat. No. 99TH8406) (Vol. 2, pp. 1470-1477). IEEE.

Inza, I., Larrañaga, P., Moujahid, A. (2008). *Algoritmos Genéticos*. [Material del aula]. Métodos Matemáticos en Ciencias de la Computación, Universidad del Pais Vasco, Leoia, España.

Junjie, P., & Dingwei, W. (2006, Agosto). *An ant colony optimization algorithm for multiple travelling salesman problem*. En First International Conference on Innovative Computing, Information and Control-Volume I (ICICIC'06) (Vol. 1, pp. 210-213). IEEE.

Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). *Optimization by simulated Annealing*. science, 220(4598), 671-680.

Laporte, G. (1992). *The traveling salesman problem: An overview of exact and approximate algorithms*. European Journal of Operational Research, 59(2), 231-247.

Pearl, J. (1984). *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Addison-Wesley.

Potvin, J. Y. (1996). *Genetic algorithms for the traveling salesman problem*. Annals of Operations Research, 63(3), 337-370.

Rabanal, P., (2010). *Algoritmos Heurísticos y Aplicaciones a Métodos Formales*. (Tesis Doctoral). Universidad Complutense de Madrid, España.

Revuelta Martinez, T. (2015). *Desarrollo y aplicación del algoritmo de Optimización basado en Colonia de Hormigas (ACO) para la resolución del Problema del Viajante Asimétrico (ATSP)*. (Tesis de pregrado). Universidad de Valladolid, España.

Russell, R. A. (1977). *An effective heuristic for the m-tour traveling salesman problem with some side conditions*. Operations Research, 25(3), 517-524.

Sastry K., Goldberg D., Kendall G. (2005). *Genetic Algorithms*. En: Burke E.K., Kendall G. (eds) Search Methodologies. Springer, Boston, MA.

Schrijver, A. (2001). *On the History of Combinatorial Optimization (Till 1960)*. Handbooks in Operations Research and Management Science. 12.

Sveska JA, Huckfeldt VE, *Computational experience with an m-salesman Traveling Salesman Algorithm*. Management Science, 1973,19.790-799.

Toro Ocampo, E., Bolaños, R., & Granada Echeverri, M. (2014). *Solución del problema de múltiples agentes viajeros resuelto mediante técnicas heurísticas*. Scientia et technica, 19(2), 174-182.