TÉCNICAS HEURÍSTICAS APLICADAS AL PROBLEMA DEL CARTERO VIAJANTE (TSP)

RESUMEN
El problema del cartero viajante (Traveling Salesman Problem – TSP) es un problema típico de optimización. En este documento se presentan algunas técnicas heurísticas de optimización (Algoritmos Genéticos, Simulated Annealing, Colonia de Hormigas, Búsqueda Tabú y Grasp) aplicadas a la solución de este problema. La solución al problema consiste en encontrar la ruta óptima para recorrer n ciudades sin repetirlas finalizando en la ciudad de origen.

PALABRAS CLAVES: Optimización, cartero viajante, algoritmos genéticos, simulated annealing, colonia de hormigas, tabu search, grasp, metaheurísticas.

ABSTRACT
The Traveling Salesman Problem is a typical problem of optimization. In this document are presented some optimization heuristic techniques (Genetic Algorithms, Simulated Annealing, Ant Colony Optimization, Tabu Search and Grasp) used in the solution of this problem. To solve this problem is necessary to find the optimal way to travel on n cities without repeat them and ending at the original city.

KEYWORDS: *Optimization, traveling salesman problem, genetic algorithms, simulated annealing, ant colony optimization, tabu search, grasp, metaheuristics.*

RICARDO ALBERTO HINCAPIÉ
Ingeniero Electricista
Estudiante Maestría Ingeniería Eléctrica
ricardohincapie@utp.edu.co

CARLOS ALBERTO RÍOS PORRAS
Ingeniero Electricista
Estudiante Maestría Ingeniería Eléctrica
alpor@utp.edu.co

RAMÓN ALFONSO GALLEGO
Profesor Facultad de Ingeniería Eléctrica
Universidad Tecnológica de Pereira
ralfonso@utp.edu.co

Grupo de Investigación en Planeamiento de Sistemas Eléctricos. Universidad Tecnológica de Pereira.

1. INTRODUCCION

Una gran cantidad de problemas importantes de optimización no pueden ser resueltos usando métodos exactos, es decir, no es posible encontrar su solución óptima con esfuerzos computacionales aceptables aunque se pueda contar con computadores de alta velocidad operando en paralelo. Un gran problema de la optimización es el fenómeno llamado explosión combinatorial, que significa, que cuando crece el número de variables de decisión del problema, el número de decisiones factibles y el esfuerzo computacional crecen en forma exponencial. Sin embargo, no todos los problemas combinatoriales son tan complejos de resolver; existen algunos para los cuales hay algoritmos que resuelven esos problemas con un esfuerzo computacional que crece de manera polinomial con el tamaño del problema, [4, 6].

Enmarcado dentro de estos problemas combinatoriales se encuentra el problema del cartero viajante (TSP). Este problema consiste en encontrar el camino más corto (ruta) para visitar n ciudades bajo la condición de visitar cada ciudad una sola vez, regresando a la ciudad de partida.

Desde la década de los 50`s muchos algoritmos han sido desarrollados para encontrar la solución a este problema encontrando buenas soluciones pero no necesariamente las soluciones óptimas. En los 80`s las técnicas de solución se centraron en la aplicación de metaheurísticas de propósito general incluyendo entre estas el simulated annealing, algoritmos genéticos, colonia de hormigas y búsqueda tabú entre otros.

En este trabajo se aplican algunas técnicas heurísticas como Simulated Annealing, Algoritmos Genéticos, Búsqueda Tabú, Grasp y Colonia de Hormigas para encontrar la solución óptima al problema del cartero viajante usando como sistema de prueba un conjunto de ocho ciudades.

2. PLANTEAMIENTO MATEMÁTICO DEL PROBLEMA

En el problema del cartero viajante – Traveling Salesman Problem (TSP) existe un conjunto de n ciudades (nodos), V = {1, 2, 3,…, n}, y un conjunto de caminos (arcos) uniendo cada una de las ciudades, así el camino (i,j)ЄA, c_{ij} es la "distancia" (función objetivo) para ir de la ciudad i a la ciudad j (c_{ij} no necesariamente es igual a c_{ji}). Un cartero debe realizar un tour (recorrido) comenzando en una cierta ciudad de origen y luego visitar todas las otras ciudades una única vez y retornar a la ciudad de origen. El problema consiste en hallar el tour (recorrido) de distancia mínima evitando subtours.
El problema del TSP tiene el siguiente modelo matemático, [4]:

Fecha de recepción: 29 Marzo de 2004
Fecha de aceptación: 16 Abril de 2004

$$\min\ z(x) = \sum_{(i,j)\in A} c_{ij}x_{ij}$$

s.a :

$$\sum_{\{i:\,(i,j)\in A\}} x_{ij} = 1 \quad \forall\ j \in V \qquad (1)$$

$$\sum_{\{j:\,(i,j)\in A\}} x_{ij} = 1 \quad \forall\ i \in V$$

$$\sum_{\{(i,j)\in A:\, i\in U,\, j\in (V-U)\}} x_{ij} \geq 1 \quad 2 \leq |U| \leq |V| - 2$$

3. TÉCNICAS DE SOLUCIÓN

Los problemas combinatoriales pueden ser divididos en dos grandes grupos considerando la existencia de algoritmos polinomiales para resolver cada tipo de problema. El primero es el problema tipo P (polinomial) para el cual existen algoritmos con esfuerzos computacionales de tipo polinomial para encontrar la solución óptima; y el segundo es el problema tipo NP (no polinomial) para el cual no se conocen algoritmos con esfuerzos computacionales de tipo polinomial para encontrar la solución óptima.

Las técnicas heurísticas son algoritmos que encuentran soluciones de buena calidad para problemas combinatoriales complejos; o sea, para problemas tipo NP. Los algoritmos heurísticos son más fáciles de implementar y encuentran buenas soluciones con esfuerzos computacionales relativamente pequeños; sin embargo, renuncian (desde el punto de vista teórico) a encontrar la solución óptima global de un problema. En problemas de gran tamaño rara vez un algoritmo heurístico encuentra la solución óptima global.

Los algoritmos heurísticos se clasifican en algoritmos constructivos (golosos), algoritmos de descomposición y división, algoritmos de reducción, algoritmos de manipulación del modelo y algoritmos de búsqueda usando vecindad. En esta última categoría pueden ser agrupados los Algoritmos Genéticos (AG), Simulated Annealing (SA), Búsqueda Tabú (TS), Colonia de Hormigas (ACO) y GRASP, [4, 5].

3.1. Algoritmos genéticos (ag)

Son herramientas matemáticas que imitan a la naturaleza e intentan resolver problemas complejos empleando el concepto de la evolución. El algoritmo ejecuta una búsqueda simultánea en diferentes regiones del espacio factible, realiza una intensificación sobre algunas de ellas y explora otros subespacios a través de un intercambio de información entre configuraciones. Emplea tres mecanismos básicos que son: La selección, el crossover y la mutación, [4, 5]:

3.1.1. Selección: Es el operador genético que permite elegir las configuraciones de la población actual que deben participar de la generación de las configuraciones de la nueva población (nueva generación). Este operador termina después de decidir el número de descendientes que debe tener cada configuración de la población actual, [4].

3.1.2. Crossover o "Recombinación": Es el mecanismo que permite pasar información genética de un par de cromosomas originales a sus descendientes, o sea, saltar de un espacio de búsqueda a otro, generando de esta forma diversidad genética.

3.1.3. Mutación: Permite realizar la intensificación en un espacio en particular caminando a través de vecinos. Significa intercambiar el valor de un gene de un cromosoma en una población. En forma aleatoria, se elige un cromosoma como candidato, se genera un número aleatorio y si es menor que la tasa de mutación ($\rho<\rho_m$), entonces se realiza la mutación. La tasa de mutación se elige del rango [0.001, 0.05].

3.2. Simulated annealing (sa)

3.2.1. Algoritmo de Metrópolis: La metodología de SA fue definida al inicio de los años 80, como una nueva herramienta para ser empleada en la solución de problemas combinatoriales de gran complejidad. Surgió del campo de la termodinámica como consecuencia de la comparación de los problemas formulados en este campo con los del campo de la investigación de operaciones, es una metodología simple y de gran potencialidad para ser aplicada a una gran variedad de problemas.

La idea original que da lugar a esta metaheurística es el *"algoritmo de Metrópolis"*, el cual a su vez está basado en el *"método de Monte-Carlo"*, con el cual se estudian las propiedades de equilibrio en el análisis del comportamiento microscópico de los cuerpos, [6].

El algoritmo de Metrópolis genera una secuencia de estados de un sólido, o sea, dado un sólido en un estado i con energía Ei, se genera el estado siguiente j mediante la aplicación de un mecanismo que transforma al estado siguiente a través de un pequeño disturbio. La energía del próximo estado es Ej; si la diferencia de energía (Ej - Ei) es menor o igual a cero, el estado j es aceptado. Si ocurre lo contrario, el estado j se acepta con cierta probabilidad de acuerdo a la ecuación 2, [6].

$$e^{\left\{\frac{E_i - E_j}{T * k_b}\right\}} \qquad (2)$$

donde T es la temperatura y k_b es una constante física conocida como constante de Boltzmann. La regla de aceptación descrita anteriormente es llamada *Criterio de Metrópolis* y el algoritmo como *Algoritmo de Metrópolis.*

Si la disminución de la temperatura es hecha de manera paulatina, el sólido podrá alcanzar un estado de equilibro en cada nivel de temperatura. En el algoritmo de Metrópolis esto se consigue después de generar un gran número de transiciones en un nivel dado de temperatura.

3.2.2. Algoritmo de Simulated Annealing: Este algoritmo aplica una acción combinada del mecanismo de generación de alternativas y del criterio de aceptación. T_k denota el valor del parámetro de control (temperatura) y N_k el número de alternativas generadas en la k-ésima iteración del algoritmo.

Inicialmente cuando T es grande, se aceptan grandes deterioros de la función objetivo; cuando T decrece, solamente se aceptan pequeños deterioros y finalmente cuando T tiende a cero, ningún deterioro es aceptado. Esta característica hace que el algoritmo SA sea diferente en relación con los algoritmos de óptimos locales. A partir del estado i con costo f(i) se genera el estado j con costo f(j). El criterio de aceptación determina si este nuevo estado es aceptado; para esto se calcula la siguiente probabilidad, [6]:

$$P_T\left\{\text{acepta}_j\right\} = \begin{cases} 1 & \text{si } \leftrightarrow f(j) \le f(i) \\ e\left\{\dfrac{f(i)-f(j)}{T}\right\} & \text{si } \leftrightarrow f(j) > f(i) \end{cases} \quad (3)$$

La estrategia seguida en SA es partir de una temperatura alta, permitiendo aceptar soluciones de pobre calidad; posteriormente, se disminuye la temperatura y a la vez la posibilidad de aceptar las soluciones peores. Se empieza con una temperatura inicial T_0 y una velocidad de enfriamiento; la temperatura T_{k+1} se calcula a partir de T_k y la velocidad de enfriamiento después de haber hecho $N(T_k)$ iteraciones en la temperatura T_k.

3.2. Colonia de hormigas (aco)

La optimización por ACO, es otra forma de imitar a la naturaleza en la forma que resuelve sus problemas, de la misma manera que lo intentan los métodos como los AG y el SA entre otros.

En este método las actividades de búsqueda son distribuidas entre *"hormigas"*, esto es, agentes con capacidades simples, que son similares al comportamiento de las hormigas verdaderas.

Uno de los problemas estudiados por los entomólogos es el de entender cómo unos insectos casi ciegos como las hormigas pueden establecer las rutas más cortas entre el nido y una fuente de comida y viceversa. Se encontró que el medio usado para intercambiar información entre los agentes sobre las rutas, consiste en rastros de *feromonas*. Una hormiga que se desplaza de un punto A a un punto B deja un rastro de feromonas (en distintas cantidades) en el suelo, marcando así el camino seguido. Mientras que una hormiga solitaria vaga en forma aleatoria, una hormiga que encuentre un rastro puede decidir, con alta probabilidad, el seguirlo, aumentando el nivel de feromonas con la suya y volviéndola más fuerte. El comportamiento colectivo que se produce es una especie de comportamiento autocatalítico, donde mientras más hormigas sigan el rastro, más atractivo se vuelve el camino para ser seguido, [1, 2, 3, 7, 8, 9, 10].

En el problema del cartero viajante, una hormiga artificial es un agente que se mueve de una ciudad a otra, el agente escoge la ciudad a la cual se va a desplazar de acuerdo a una función probabilística que depende de la cantidad de feromonas dejada en ese trayecto y de una función heurística, la cual se define como una función de la distancia. Inicialmente, *m* hormigas artificiales son localizadas en ciudades aleatorias, en cada unidad de tiempo ellas se mueven de una ciudad a otra actualizando el rastro de feromonas en los caminos usados, esto se considera como *actualización local de rastro*. Cuando todas las hormigas han completado su tour, la hormiga que realizó el camino más corto, realiza una nueva actualización de feromonas, pero solo en los caminos que usó, esto es considerado como *actualización global de rastro*, esta actualización depende del inverso de la distancia recorrida por la hormiga.

Una hormiga *k* en la ciudad *r*, escoge irse para la ciudad *s*, entre las ciudades que no ha visitado de acuerdo con la siguiente fórmula probabilística, [3]:

$$s = \begin{cases} \max\limits_{u \in posibles} \left\{ \left[\tau(r,u)\right]^{\alpha} * \left[\eta(r,u)\right]^{\beta} \right\} & \text{si } q \le q_0 \\ S & \text{de lo contrario} \end{cases} \quad (4)$$

$\tau(r,u)$ es la cantidad de rastro de feromona en el segmento de viaje (r,u), $\eta(r,u)$ es una función heurística que es escogida para este caso como el inverso de la distancia entre *r* y *u*, . y . son parámetros que pesan la importancia de el rastro de feromona y de la función heurística, *q* es un valor escogido en forma aleatoria entre 0 y 1. *S* es escogida mediante una formula probabilística que favorece las ciudades cuya distancia sea más corta y tengan un alto nivel de rastro de feromona, ecuación (5):

$$p_k(r,s) = \begin{cases} \dfrac{\left[\tau(r,s)\right]^{\alpha} * \left[\eta(r,s)\right]^{\beta}}{\sum\limits_{u \in permitidos} \left[\tau(r,s)\right]^{\alpha} * \left[\eta(r,s)\right]^{\beta}} & \text{si } s \in permitidos \\ 0 & \text{de lo contrario} \end{cases} \quad (5)$$

$p_k(r,s)$ es la probabilidad de que la hormiga *k* se mueva de la ciudad *r* a la ciudad *s*.

El rastro de feromona se actualiza tanto localmente como globalmente. La actualización global se hace para recompensar los caminos tomados por el mejor tour. Una

vez que las hormigas artificiales han terminado sus tours, la mejor hormiga deposita una cantidad de feromona en los caminos visitados, la cantidad de feromona depositada es inversamente proporcional a la longitud del tour, mientras más corto el tour mayor la cantidad de feromona depositada, [3]:

$$\tau(r,s)_{t+1} = \rho\tau(r,s)_t + \lambda\frac{1}{dis(r,s)} \qquad (6)$$

. es un factor entre 0 y 1 que representa la tasa de evaporación de la feromona para evitar que la feromona se acumule infinitamente y . es otro parámetro.

La actualización global es similar a un proceso de aprendizaje reforzado, donde las mejores soluciones tienen prioridad.

La actualización local pretende evitar que un camino muy fuerte sea escogido por todas las hormigas, cada vez que una hormiga escoge un camino la cantidad de feromona en el camino es actualizada con la ecuación (7), [3]:

$$\tau(r,s)_{t+1} = \rho\tau(r,s)_t + \lambda\tau_0 \qquad (7)$$

τ_0 es un parámetro.

3.3. Búsqueda tabú (ts)

Es un procedimiento metaheurístico utilizado para manejar un algoritmo heurístico de búsqueda local y así evitar que el proceso se detenga en un óptimo local. Por lo tanto, TS realiza una exploración a través del espacio de configuraciones delimitando adecuadamente los óptimos locales.

Para evitar que el proceso regrese a los óptimos locales y entre en un ciclo repetitivo, la búsqueda tabú clasifica los movimientos más recientes como "movimientos tabú"; estos prohíben que una configuración sea visitada nuevamente.

Este método tiene dos tipos de memoria: *Memorias de corto plazo y largo plazo*. La de corto plazo contiene los eventos ocurridos recientemente y en la de largo plazo, se almacenan los datos de frecuencia de determinados eventos. La información de la memoria de largo plazo es fundamental para definir las estrategias de diversificación, las cuales permiten explorar otras regiones no visitadas anteriormente, [6].

Cuando un movimiento ha sido clasificado como tabú y después de ser analizado produce una función objetivo mejor que un valor de referencia escogido (que puede ser una incumbente u otra buena solución previamente encontrada), entonces se aplica la denominada *regla de aspiración*, esta consiste en cancelar la prohibición y aceptar el movimiento, [6].

3.4. Grasp

Es una evolución de los algoritmos heurísticos constructivos, especialmente de aquellos que usan indicadores de sensibilidad. Con estos indicadores se calcula la variación de la función objetivo con respecto a las variables de interés del problema de optimización, y se usan para identificar los atributos atractivos de tal problema. Emplea una propuesta intermedia entre Simulated Annealing y Búsqueda Tabú para realizar la fase de exploración y ha mostrado ser eficiente en la solución de problemas complejos de optimización, [6, 11].

Para un problema genérico, el algoritmo GRASP tiene los siguientes pasos:

1. Implementar una fase de pre-procesamiento.
2. Realizar la fase de búsqueda constructiva.
3. Realizar la fase exploratoria y actualizar la mejor solución encontrada si se supera la incumbente.
4. Si el criterio de parada no se satisface volver al paso 2. Sino, finalizar el proceso. La respuesta del algoritmo es la mejor solución almacenada.

Con el pre-procesamiento se trata de identificar los atributos más interesantes del problema con los que se realiza el proceso de búsqueda. Esto permite disminuir el espacio de soluciones que se quiere explorar.

La fase de búsqueda constructiva consiste en encontrar una solución de calidad para el problema con base en un algoritmo heurístico constructivo donde se escoge, en cada paso, un elemento de una lista de tamaño *k* denominada *RCL,* donde se clasifican las variables más atractivas y se obtiene una incumbente de buena calidad del problema.

La fase constructiva del algoritmo presenta los siguientes pasos:

1. Escoger una solución inicial que puede ser vacía, es decir, sin adicionar variables la cual se transforma en la solución actual del problema.
2. Para la solución actual del problema, elaborar una lista que clasifica las mejores *k* variables que identifican el indicador de sensibilidad.
3. Escoger en forma aleatoria o probabilística una de las variables de la lista y actualizar la solución con la adición o sustracción de la variable escogida.
4. Si la solución actual es factible o se satisface el criterio de parada, se finaliza la fase constructiva.

La fase exploratoria procura encontrar una solución óptima local en la vecindad de la solución de la fase constructiva. Esta fase es prácticamente equivalente a un

proceso de intensificación en el algoritmo de búsqueda tabú. El proceso consiste en definir una vecindad de la solución de la fase constructiva y encontrar una solución factible de mejor calidad en esa vecindad. Siempre que se encuentra una solución factible de mejor calidad, la búsqueda local se debe reiniciar encontrando una nueva vecindad de la nueva solución. Este proceso, en general, precisa de un alto esfuerzo computacional dependiendo del tipo de problema, ya que analizar un vecino requiere de la solución de un problema de programación lineal PL, [6, 11].

4. SISTEMA DE PRUEBA Y RESULTADOS

El problema del TSP se aplica a 8 ciudades y las distancias se encuentran en la tabla 1.

	1	2	3	4	5	6	7	8
1	0	60	270	40	350	550	550	300
2	60	0	200	550	320	100	550	550
3	270	200	0	300	300	550	550	450
4	40	550	300	0	350	100	550	270
5	350	320	300	350	0	200	350	500
6	550	100	550	100	200	0	120	280
7	550	550	550	550	350	120	0	300
8	300	550	450	270	500	280	300	0

Tabla 1. Distancia entre ciudades [Km]

4.1. Algoritmos Genéticos

En forma aleatoria se conformó una población inicial, en la que cada cromosoma representa una posible ruta a seguir y cada gen indica el número de la ciudad.

Los datos utilizados por el algoritmo son:

Población inicial: 14 cromosomas (aleatoria)
Número de genes por cromosoma: 8
Proceso de selección: Esquema de la ruleta
Tipo de crossover: Punto simple
Tasa de crossover: $\rho_c = 0.9$
Tasa de mutación: $\rho_m = 0.02$
Máximo número de generaciones: 400
Criterio de parada: 20 generaciones consecutivas sin que mejore la función objetivo o máximo número de generaciones.

Los resultados encontrados por el algoritmo de AG son:

Ruta óptima: *1 4 8 7 6 5 3 2*
Función objetivo: 1490
Total de generaciones: 125

4.2. Simulated Annealing

El tamaño de N_0 (Cadena Inicial de Markov) es estimado en función del número de variables del sistema, donde k generalmente asume valores enteros. Entre más grande el sistema a resolver mayor debe ser el k, por lo tanto $N_0 = k*n = 1*8 = 8$. Para este ejemplo se asumió un valor de k = 1 y el valor de n correspondiente al problema es igual a 8 (número de ciudades).

Para calcular el valor de T_0 se inició la cadena con $N_0 = 8$ (8 iteraciones) a partir de la ruta inicial aleatoria Xi = [1 2 3 4 5 6 7 8] donde F(Xi) = 1830. Las funciones objetivo de las transiciones de una ruta a otra (para las 8 transiciones) se ilustran a continuación:

1830 - 1830 - 1810 - 2720 - 3550 - 4320 - 3410 - 2580 - 3490

De la lista anterior se puede observar que el número de funciones objetivo que empeoraron con respecto al valor anterior fueron 4 y las que mejoraron o quedaron igual fueron 4. Por lo tanto, $m_1 = 4$ y $m_2 = 4$. Adicionalmente se asumió un valor para la tasa de aceptación x = 0.85. El valor de T_0 se halla según [6], $T_0 = 2397{,}1$. Como $m_1 + m_2 = N_0 = 8$ entonces se termina el algoritmo con el último valor hallado de T_0.

Con la temperatura T_0 se repite el procedimiento 8 veces (Tamaño de la cadena de Markov). Después de terminado el ciclo se actualizan los valores de N_0 y T_0. Se debe garantizar que la temperatura y la cadena de Markov tengan una relación inversa, por lo tanto cuando la temperatura se este enfriando (disminuyendo) la cadena de Markov (número de iteraciones) debe aumentar. Los nuevos valores están dados por $Nk+1 = \rho*Nk$, donde ρ generalmente es mayor que 1. Asumiendo un valor de 1.2 se tiene que $Nk+1 = 1.2 * 8 = 9.6 \cong 10$. Para hallar la tasa de enfriamiento de T se empleó una relación lineal donde en cada iteración la temperatura disminuye un 10%, por lo tanto $Tk+1 = 0.9*Tk = 2157{,}39$. Con estos nuevos valores de Nk y Tk se inicia otra iteración para el algoritmo. El criterio de parada empleado fue de 1000 iteraciones totales de Nk.

Parámetros y respuestas del algoritmo Simulated Annealing:

Ruta óptima: 1 4 8 7 6 5 3 2
Función Objetivo: 1490
Número de niveles de temperatura y cadena: 14
Valor inicial – final de temperatura: [2397.1 - 607.71]
Rata de variación de temperatura: $\rho_T = 0.9$
Rata de variación de cadena: $\rho_N = 1.2$
Valor inicial - final de cadena: [10 - 122]
Número de propuestas: 625

4.3. Colonia de hormigas

Los parámetros utilizados fueron: α=1, β=5, ρ=0.5, λ=0.5. Máximo número de iteraciones: 20.
Respuestas halladas por el algoritmo de ACO:

Ruta óptima:	*1 4 8 7 6 5 3 2*
Función objetivo:	*1490*
Número de ciclos:	*10*

4.4. Búsqueda Tabú

Configuración inicial:	Aleatoria
Máximo número de prohibiciones (Número tabú estático):	4
Máximo número de vecinos analizados:	3
Máximo número de iteraciones en proceso de intensificación:	20

Respuesta encontrada:

Ruta óptima:	*1 4 8 7 6 5 3 2*
Función objetivo:	*1490*
Número de iteraciones:	*14*

4.5. Grasp

Se implementaron la fase constructiva y búsqueda local. La fase constructiva se resuelve a través de una metaheurística para encontrar una solución de buena calidad. La fase constructiva se realiza por medio de un algoritmo de Simulated Annealing, [6, 11].

Máximo número de iteraciones:	50
Indicador de sensibilidad:	0.8
Temperatura inicial:	1000
Tasa de aceptación:	0.85
Criterio de parada:	1000 iteraciones totales de Nk.

Respuesta proporcionada por el algoritmo GRASP:

Ruta óptima:	*1 4 8 7 6 5 3 2*
Función Objetivo:	*1490*

5. CONCLUSIONES

A pesar de que el sistema de prueba es pequeño, se podrán establecer algunas conclusiones respecto a los métodos de optimización utilizados en su solución: Todos los métodos alcanzan la solución óptima, sin embargo, estos presentan inconvenientes si se implementan en su forma natural, por lo que tienen que ser usados operadores especializados, como en el caso de los AG, que requieren operadores especializados tanto para el crossover como para la mutación con el fin de evitar la generación de rutas infactibles, así todos los métodos usados requieren que el movimiento que se lleve a cabo no produzca infactibilidades, caso contrario no será posible encontrar la solución.

Se encontró que los algoritmos que ofrecen mayores ventajas son Colonia de Hormigas, Simulated Annealing y Búsqueda Tabú debido a que son más fáciles de implementar y convergen con el menor número de iteraciones. En los casos que involucran muchas ciudades, los métodos que más facilidades presentan son el Simulated Annealing, Colonia de Hormigas y el Búsqueda Tabú.

6. AGRADECIMIENTOS

Los Autores expresan su agradecimiento a la Universidad Tecnológica de Pereira, Colombia, por su apoyo al grupo de Planeamiento de Sistemas Eléctricos.

7. BIBLIOGRAFÍA

[1] Dorigo, M.: "Optimization, learning and natural algorithm", Ph.D Thesis, Dip. Elettronica e Informazione, Politecnico di Milano, Italy, 1992.
[2] Dorigo, M., Maniezzo, V. And Colorni, A.: "Ant System: Optimization by a Colony of Cooperating Agents", IEEE trans on Systems, Man and Cybernetics – Part B, Vol 26 No 1, pp29-41, Feb 1996.
[3] Dorigo, M., Gambardella, L.: "Ant Colonies for the travleing salesman problem", Biosystems 1997.
[4] Gallego R., Ramón; Romero L., Rubén y Escobar Z., Antonio. "Algoritmos Genéticos", texto guía en Maestría en Ingeniería Eléctrica U.T.P.
[5] Gallego R., Ramón y Escobar Z. Antonio: "Statical Planning of Colombia`s Transmission Systems Using Genetics Algorithm", 16th Internacional Conference on CAD/CAM Robotic & Factories of the Future, Trinidad y Tobago, Junio 2000.
[6] Gallego R., Ramón y Romero L., Rubén. "Optimización Combinatorial", texto guía en Maestría en Ingeniería Eléctrica U.T.P.
[7] Merkle, D., Middendorf, M. and Schmeck, H.: "Ant Colony Optimization for Resource – Constrained Project Scheduling", IEEE Trans on Evolutionary Computing, Vol 6 No 4, pp333-346, Aug 2002.
[8] Parpinnelli, R., Lopes, H. and Freitas, A.: "Data Mining with an Ant Colony Optimization Algorithm", IEEE Trans on Evolutionary Computing, Vol 6 No 4, pp321-332, Aug 2002.
[9] Solnon, C.: "Ants can solve Constrint Satisfaction Problems", IEEE Trans on Evolutionary Computing, Vol 6 No 4, pp347-357, Aug 2002.
[10] Stützle, T.; Dorigo, M.: "A short convergence proof for a Class of Ant Colony Optimization Algorithm," IEEE Trans on Evolutionary Computing, Vol 6 No 4, pp 358-365, Aug 2002.
[11] Glover, F. and Kochenberger G. A. Handbook of Mehaheuristics, Kluver Academic Publishers, Norwell, M. A., 2003