Modificaciones en el algoritmo de Held & Karp

JOAQUÍN A. PACHECO
E.U.E.Empresariales de Burgos

RESUMEN

El trabajo describe un conjunto de variaciones en el algoritmo de Held & Karp para el TSP simétrico. Este es uno de los problemas más estudiado de la matemática combinatoria en las últimas décadas. La obtención de la solución óptima para este problema requiere un tiempo de computación habitualmente excesivo, sobre todo si el número de ciudades es alto. Con estas variaciones se consigue reducir enormemente el tiempo de computación en el algoritmo de Held & Karp original. De esta forma se obtiene un procedimiento que alcanza la solución exacta, para problemas de 100 nodos en ordenadores personales, en un tiempo razonable.

1.- INTRODUCCION

El problema del vendedor ambulante puede ser descrito de la siguiente forma: un individuo debe visitar n ciudades, partiendo de una ciudad inicial y regresando a ella después de haber visitado cada una de las otras n-1 ciudades una vez, de forma que la distancia total recorrida sea mínima. Este es uno de los problemas de la matemática combinatoria más estudiado y analizado en los últimos años, y está estrechamente relacionado con otros problemas de rutas, y con problemas de localización, programación y planificación, etc...

Cuadernos de Estudios Empresariales, nº 6. Servicio de Publicaciones UCM, Madrid 1996.

En la literatura existen muchos algoritmos para resolver este problema, tanto heurísticos como exactos. Entre los heurísticos destacan fundamentalmente los basados en los *intercambios k~óptimos,* concepto introducido por Lin, [11], y que ha dado lugar a procedimientos de solución para problemas simétricos, tan conocidos como el propuesto por el propio Lin en el mismo artículo, el de Lin & Kemighan, [12], o el de Or, [15]. Existen también adaptaciones de estos métodos para problemas asimétricos, como el propuesto por Webb, [24], Kanellakis y Papadimitriou, [9], etc... Un criterio de clasificación de los diversos algoritmos heurísticos, se puede encontrar en el trabajo de Golden y otros, [5]. Así mismo, adaptaciones de las técnicas mencionadas a problemas con ventanas de tiempo se pueden hallar en los trabajos de Solomon, [20] y [21], y Solomon y otros, [19].

Entre los algoritmos exactos destacan principalmente el algoritmo de Little y otros, [13], para matrices asimétricas, técnica basada en un proceso recursivo de reducción de la matriz de distancias, y que ha sido adaptado a otros problemas de rutas como el VRP, (Ver Christofides y Eilon, [3]), o el TSP con carga y descarga, (Ver Kalantari y otros, [8]); el de Eastman, [4], también para matrices asimétricas; el de Held & Karp, [6] y [7], basado en los *1-árboles de expansión,* al igual que el propuesto por Volgenant y Jonker, [23], ambos para matrices simétricas. Por su adaptabilidad a problemas de rutas con ventanas de tiempo, no se puede dejar de mencionar las técnicas basadas en la *relajación del espacio de estados* cuando se plantea el problema en términos de programación dinámica, idea desarrollada por Christofides, Mingozzi y Thot, [2].

En cualquier caso, en la literatura no dejan de surgir nuevos procedimientos de solución o adaptaciones de las técnicas ya existentes. Recientemente Nurmi, [14], ha propuesto una serie de modificaciones en algunos de los algoritmos antes mencionados mejorando su eficacia en muchos casos.

En este trabajo se describen variaciones en el algoritmo original de Held & Karp, concretamente en el proceso de acotación que, de igual forma que el algoritmo original, está basado en los 1-árboles de expansión, sin embargo en este caso se cambia la forma en que varían los vectores de penalización que se introducen. De esta forma se consiguen cotas inferiores en cada una de las ramas del árbol de búsqueda muy ajustadas al valor del óptimo en dichas ramas, eliminándose exploraciones *innecesarias* y reduciendo enormemente el tiempo de computación. Este trabajo supone la continuación de una línea de trabajo iniciada por el autor meses atrás, y reflejada en Pacheco, [16] y [17].

El trabajo viene organizado de la siguiente forma: en el segundo apartado se describe brevemente el algoritmo de Held & Karp, (idea básica y proceso general), en el tercer apartado se explica y se describe en seudocódigo las modificaciones propuestas; finalmente en el cuarto apartado se muestran los resultados obtenidos al resolver un conjunto de problemas simulados con 100 nodos, tanto por el algoritmo original como por las variantes propuestas.

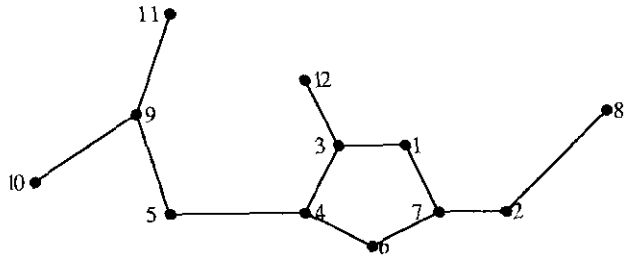
El software utilizado para ejecutar dichos problemas con estos algoritmos, esta formado por los compiladores TURBO PASCAL 6.0 y BORLAND PASCAL (TURBO PASCAL 7.0 v.Profesional) de BORLAND. El equipo utilizado tanto para la programación como para la ejecución es un ordenador personal de tipo PC-AT, i.486.

2. - ALGORITMO DE HELD & KARP

Held & Karp, [6] y [7], proponen una estrategia de solución Branch & Bound para hallar una solución exacta al TSP simétrico, en la que se van a utilizar los mínimos árboles de expansión tanto en el proceso de ramificación como el de acotamiento.

2. 1. - ASPECTOS TEORICOS DEL ALGORITMO

Para explicar este proceso considérese K_n un grafo completo no dirigido, con el conjunto de nodos {1,2,...,n}; sea (i,j) el arco incidente en los nodos i y j con coste o *peso* c_{ij}. Un 1-árbol consiste en un árbol definido sobre los nodos {2,3,...,n}, junto con dos arcos incidentes en el nodo 1. Por tanto, un l-árbol de mínimo peso se puede encontrar a partir de un mínimo árbol de expansión en los vértices {2,3,...,n} añadiendo los arcos incidentes en el nodo 1 de menor coste. (Ver gráfico).



Un 1-Arbol (Gráfico 1)

Por tanto si un 1-árbol de mínimo peso o coste es un tour, entonces es la solución al TSP definido en este grafo.

Lema.-

Sea $\pi = (\pi_1, \pi_2, ..., \pi_n) \in R^n$ si C*es un tour de mínimo coste para la matriz de costes (c_{ij}), entonces lo es también para la matriz de pesos $(c_{ij} + \pi_i + \pi_j)$.

Demostración.- sea C un tour, el coste con respecto a (c_{ij}), es (c_{ij}) es $\sum_{(i,j)\in C} c_{ij}$, y con respecto a $\left(c_{ij} + \pi_i + \pi_j\right)$ es $\sum_{(i,j)\in C} \left(c_{ij} + \pi_i + \pi_j\right) = \sum_{(i,j)\in C} c_{ij} + 2\sum_{i=1}^{n} \pi_i$; ya que cada nodo tiene exactamente dos arcos incidentes.

Sin embargo la transformación de (c_{ij}) a $(c_{ij} + \pi_i + \pi_j)$, en general, sí que puede afectar la búsqueda del 1-árbol de mínima expansión o coste. Por tanto *una posible estrategia para el TSP, es buscar, si es posible, un vector* π *de tal* forma que un 1-árbol de mínima expansión con respecto a $c_{ij} + \pi_i + \pi_j$ sea un tour.

Se define $f(\pi)$, función *hueco*, como la cantidad en la que el coste del tour óptimo supera al coste del 1-árbol de mínima expansión, ambos con respecto a $(cij + \pi i + \pi j)$. Se tiene que, $\forall\ \pi$, $f(\pi) \geq 0$, lo que no se tiene garantizado es que $\min f(\pi) = 0$, sin embargo puede suponer una buena aproximación a la solución del TSP determinar donde se halla este mínimo.

Sea:

W el coste del tour óptimo con respecto a (c_{ij});
C_k el peso del k-ésimo 1-árbol con respecto a (c_{ij});
d_{ik} el número de arcos incidentes en el nodo i en el k-esimo 1-árbol; y
K el número de 1-árboles;

se tiene que

$$f(\pi) = W + 2\Sigma_{i=1}^{n} \pi_i - \min_{k=1..K}\left[c_k + \Sigma_{i=1}^{n} \pi_i d_{ik}\right],$$

Luego

$$f(\pi) = W - \min_k\left[c_k + \Sigma_{i=1}^{n} \pi_i\left(d_{ik} - 2\right)\right];$$

por tanto minimizar $\mathbf{f}(\pi)$ equivale a maximizar $w(\pi) = \min_{k=1..K}\left[c_k + \pi^T \cdot \mathbf{v}_k\right]$, , donde se ha hecho $v_{ik} = d_{ik}$,-2;

en forma vectorial

$$w(\pi) = \min_{k=1..K} \left[c_k + \pi^T \cdot \mathbf{v}_k \right],$$

donde $w(\pi) \leq W$, ya que $f(\pi) \leq 0$.

Luego $w(\pi)$ puede servir de cota inferior del coste del tour óptimo, y mejor aún $\max_{\pi} w(\pi)$. En los trabajos mencionados se explican diversos métodos para aproximarse a $\max_{\pi} w(\pi)$, concretamente se propone el siguiente proceso iterativo:

$$\pi^{m+1} = \pi^m + t_m v_{k(\pi^m)};$$

siendo $k(\pi)$ el índice del 1-árbol de menor peso en π, es decir, respecto $(c_{ij} + \pi_i + \pi_j)$; y t_m sucesión de escalares, suelen tomarse t_m=t constantes.

2.2.- DESCRIPCION DEL PROCESO DE ACOTACION Y RAMIFICACION

Este proceso se incorpora al siguiente algoritmo Branch & Bound: En cada rama del árbol de búsqueda se consideran los siguientes parámetros de estado, $(X, Y, \pi, W_{X,Y}(\pi))$, donde

π es el vector de penalizaciones en ese momento;
X e Y son subconjuntos de arcos disjuntos del grafo inicial K_n;
$W_{X,Y}(\pi) = \min_{k \in \Gamma(X,Y)} (c_k + \pi \mathbf{v}_k)$.

siendo $\Gamma(X,Y)$ el conjunto de 1-árboles que incluyen los arcos de X, y excluyen los de Y.

El estado del vértice inicial es de la forma $(\emptyset, \emptyset, 0, W(0)$ posteriormente se elige en cada paso la rama de conjuntos de estado, (X, Y, p, $W_{XY}(\pi)$) de menor cota, y se calcula la iteración $\pi^{m+1} = \pi^m + t\mathbf{v}_{k(\pi^m)}$ antes mencionada.

Se pueden dar únicamente uno de los dos resultados siguientes:

• Para algún m, $W_{X,Y}(\pi^m) \geq \bar{C}$, donde $\bar{C}$ es una cota superior conocida del TSP. Esta rama debe ser rechazada.

• Para algún entero h, $\max_{0\le j\le ph+1} W_{X,Y}(\pi^j) = \max_{0\le j\le p[h+1]+1} W_{X,Y}(\pi^j)$; luego no hay *mejora* en un bloque de p iteraciones, donde p es un parámetro previamente establecido.

En este segundo caso se escoge un $\pi'_{X,Y}$ tal que $W_{X,Y}(\pi') = \max_{0\le j\le ph+1} W_{X,Y}(\pi^j)$ y se realiza la ramificación: se crean nuevas entradas, $\left[X_i, Y_i, \pi', W_{X_i,Y_i}(\pi')\right]$, donde $X \subseteq X_i, Y \subseteq Y_i$, y cada uno de los tours de $\Gamma(X,Y)$ está en alguno de los conjuntos $\Gamma(X_i, Y_i)$.

Más concretamente, en la ramificación se forman los conjuntos X_i e Y_i de la siguiente forma: los arcos que no están presentes en los conjuntos X e Y se ordenan de la forma e_1 , e_2, ..., e_r, donde:

$$W_{X,Y\cup\{e_1\}}(\pi') \ge W_{X,Y\cup\{e_2\}}(\pi') \ge .... \ge W_{X,Y\cup\{e_r\}}(\pi'),$$

es decir se ordenan según el tamaño de la cota a que den lugar si son excluidos. Los conjuntos X_i e Y_i que se consideran son los siguientes:

$X_1 = X$, $Y_1 = Y \cup \{e_1\}$,
$X_2 = X \cup \{e_1\}$, $Y_2 = Y \cup \{e_2\}$,
$X_3 = X \cup \{e_1, e_2\}$, $Y_3 = Y \cup \{e_3\}$,
.....
.....
$X_q = X \cup \{e_1, e_2, ..., e_{q-1}\}$, $Y_q = Y \cup R_i$,

siendo q el menor entero para el cual hay un nodo *i* tal que en X no hay dos arcos coincidentes en él pero en X_q sí; y R_i, el conjunto de arcos coincidentes en este vértice i que no están en Xq (no debe de haber más de dos arcos coincidentes en cada vértice). Si hubiera dos o más vértices i,j de esta forma, entonces . $Y_q = Y \cup R_i \cup R_j$.

De esta forma se elige la rama con menor cota inferior, y se continúa con la exploración hasta obtener sólo una ruta de los conjuntos $\Gamma(x,y)$ de 1-árboles. A continuación se exploran las ramas que no han sido analizadas siempre que la cota inferior correspondiente no supere el valor de la mejor solución obtenida hasta ese momento.

3. - DESCRIPCION DE MODIFICACIONES

3.1. - IDEA BASICA

Las modificaciones propuestas afectan al proceso de cálculo de la cota inferior en cada vértice, es decir, del cálculo de $\max_\pi w(\pi)$, mediante el proceso iteractivo $\pi^{m+1} = \pi^m + t_m V_{k(\pi^m)}$ descrito en el apartado anterior.

Frente al criterio de los autores, que proponen mantener t_m constante ($t_m = t$), en este trabajo se propone variar este parámetro, según los valores que va tomando $w(\pi^m)$, de la siguiente forma:

- Comenzar con un valor inicial para t_1;
- Cada vez que $w(\pi^m)$, mejore el valor del $\max_\pi w(\pi)$, obtenido hasta ese momento aumentar el valor de t_m;
- Si en un determinado número de iteraciones no hay mejora en el $\max_\pi w(\pi)$ reducir el valor de t_m.

El objeto de este proceso es evitar caer en máximos locales, de los que muchas veces no se puede salir si se mantiene t constante, a la vez que se impide que el proceso emplee un tiempo excesivo de cálculo.

El tiempo empleado en este proceso modificado es, en cualquier caso, mayor que el empleado por el procedimiento original, sin embargo se obtienen cotas inferiores realmente muy ajustadas al óptimo en cada vértice, con lo que el número de exploraciones se reduce enormemente al igual que el tiempo total de computación del algoritmo, como se verá mas adelante.

La segunda variación que se propone sigue la misma idea que la anterior, sólo que en este caso el valor inicial de t_1 no es constante sino que viene establecido como un parámetro de entrada para cada vértice, junto con X, Y, π, y $W_{X,Y}(\pi)$. El valor t' que van a *heredar* las siguientes ramas va a ser $t' = t_k$, siendo $k = \min \{j / w(\pi^j) = \max_\pi w(\pi)\}$; o en otras palabras, el valor de t_m en el momento que se alcanza dicho máximo.

El objeto de esta variante es *dar continuidad* a todo el proceso de acotación en las diferentes ramas de la misma forma que se hace con el vector de penalizaciones π. Los resultados obtenidos por estas dos variantes son, en conjunto, parecidos y siempre mucho mejores que con el algoritmo original.

3.2. - DESCRIPCION DE LOS NUEVOS PROCEDIMIENTOS

A continuación se describe las variantes señaladas, haciendo hincapié fundamentalmente en los pasos correspondientes a las modificaciones introducidas: Para ello se definen los siguientes variables:

- COSTEMINIMO: Coste de la mejor solución obtenida hasta ese momento;

- RUTA: Conjunto de arcos que forman la mejor solución obtenida hasta el momento;
- CONTADOR: Número de iteraciones sin mejora en la cota inferior desde la última modificación del parámetro t;
- CONTADORMAX: Número de iteraciones sin mejora en la cota inferior desde la última mejora;
- NUMARCOS(i) : Número de arcos incidentes en cada nodo i en cada iteración.

y los siguientes parámetros:

- MAXITERTOT: Máximo número de iteraciones sin mejora en la cota inferior desde la última mejora;
- MAXITER: Máximo número de iteraciones sin mejora en la cota inferior desde la última modificación;
- *alfa* y *beta:* Coeficientes por el que se multiplica respectivamente el parámetro t para aumentar y disminuir su valor.
- t1: valor inicial del parámetro t en cada exploración.
- tmin: valor mínimo del parámetro t por debajo del cual se interrumpe el cálculo de la cota inferior.

A continuación se describe en primer lugar el procedimiento recursivo de la exploración en cada rama, y a posteriormente como queda insertado dicho procedimiento en el procedimiento principal:

PROCEDIMIENTO EXPLORACION (X,Y, π):

Paso 0.: Si los arcos de X forman un tour:
Definir *costetot* como la suma de los costes de los arcos de X;
Si *costetot* < COSTEMINIMO, entonces:
hacer COSTEMINIMO = *costetot*,
hacer RUTA = X;
Parar.
en caso contrario ir al paso siguiente.

Paso 1.: Inicializar variables:
hacer t2 = t1, π2 = π, w1= -∞, CONTADOR = 0, CONTADORMAX=0.

Paso 2.: Hacer: CONTADOR = CONTADOR+1,
CONTADORMAX = CONTADORMAX+1,
Determinar w = W(π2) (según se definió en el apartado anterior).

Paso 3.: Si w > wl entonces hacer:
CONTADOR = 0, CONTADORMAX = 0, wl = w, $\pi 1 = \pi 2$, t2 *alfa*·t2;
en caso contrario hacer:
CONTADOR = CONTADOR+1, CONTADORMAX = CONTADORMAX+1.

Paso 4.: Si CONTADOR = MAXITER entonces:
hacer t2 = *beta*·t2, CONTADOR = 0.

Paso 5.: Definir NUMARCOS(i) como el número de arcos incidentes en el nodo i correspondientes al cálculo de W($\pi 2$);
Hacer v(i) = NUMARCOS(i) - 2, i = 1, 2, ..., n;
Hacer $\pi 2 = \pi 2 + t2 \cdot \mathbf{v}$.

Paso 6.: Si ((CONTADORMAX = MAXITERTOT) o (t2 < tmin) o (wl ≥ COSTEMINIMO)
o (v(i) = 0, i = 1, 2, ..., n;)) entonces ir al paso 7;
en caso contrario ir al paso 2.

Paso 7.: Si wl < COSTEMINIMO entonces ir al paso siguiente;
en caso contrario parar.

Paso 8.: Si v(i) = 0, i = 1, 2,..., n:
Hacer RUTA como el conjunto de arcos que forman el 1-árbol correspondiente al cálculo de wl,
Definir COSTEMINIMO la suma de los costes de dicho conjunto.
Parar;
en caso contrario ir al paso siguiente.

Paso 9.: Seleccionar los q arcos correspondientes a $\pi 1$ según se explicó en el apartado anterior;
Para i = 1,...,q:
Determinar los conjuntos X_i e Y_i
Ejecutar EXPLORACION(X_i,Y_i, πl).

Paso 10.: Parar.

PROCEDIMIENTO PRINCIPAL

Paso 0.: Leer datos iniciales: *n* número de nodos y $\mathbf{c} = (c_{ij})$ matriz de distancias.

Paso 1.: Determinar RUTA y COSTEMINIMO como las solución obtenida por la ejecución de un heurístico para el TSP simétrico.

Paso 2.: Hacer X, Y = $\emptyset$, π = (0),
Ejecutar EXPLORACION(X,Y,π)

Paso 3.: Finalizar.

La segunda variante opera igual que la anterior salvo unas pequenas modificaciones que se indican a continuación:

PROCEDIMIENTO EXPLORACION1(X, Y, π, t);

Paso 1.: Inicializar variables:
hacer t2 = t, (el resto del paso igual que en EXPLORACION1(X, Y, π)).

Paso 3.: Si w > wl entonces hacer:
t3 = t2, (el resto del paso igual que en EXPLORACION1(X, Y, π));

Paso 9.: Seleccionar los q arcos correspondientes a π1 según se explicó en el apartado anterior;
Para i = 1, ..., q:
Determinar los conjuntos X_i e Y_i,
Ejecutar EXPLORACION (X_i,Y_i, π1, t3).

siendo t3 la variable auxiliar donde se guarda el valor de t correspondiente al $\max_{\pi} w(\pi)$.

PROCEDIMIENTO PRINCIPAL1

Paso 0.: Leer datos iniciales: n y **c.** leer valor de t0.

Paso 2.: Hacer X, Y = $\emptyset$, π = (0), t = t0,
Ejecutar EXPLORACION(X,Y, π, t).

4.- RESULTADOS COMPUTACIONALES

A continuación se muestran los resultados conseguidos por estas variantes para la resolución de una serie de problemas correspondientes a la simulación de 20 matrices de distancias simétricas de dimensión 100, comparándose los resultados de estas variantes con los obtenidos por el algoritmo original de Held & Karp.

Los detalles de esta simulación son los siguientes:

- las matrices de distancia son obtenidas mediante generación de numeros siguiendo una variable aleatoria que toma valores enteros entre 0 y 100 con la misma probabilidad;
- en el caso de la primera variante el valor inicial de t1 es 1.0 en el vértice inicial, y 0.5 en los demás; en la segunda variante el valor inicial de t0 es 1; de igual forma en la ejecución del algoritmo de Held & Karp original el valor de t es 1.0 en el vértice original y 0.5 en los demás;
- el valor de *alfa,* para ambas variantes, es 2.0 y el de *beta* es 0.5;
- el valor de MAXITER es 10 y MAXITERTOT es 40 en todas las exploraciones, para ambas variantes; para el algoritmo de Held & Karp se ha dado a MAXITERTOT el valor de 40 en el vértice inicial y 20 en los demás;
- el valor de tmin para ambas variantes es 0.000 1;
- para la resolución del problema de determinar el *1-árbol* de menor coste se ha utilizado el algoritmo de Prim, [18], que según experiencias desarrolladas por Syslo y otros, [22], utiliza menos tiempo de computación que otros, como el de Kruskal, [11], si se trata con redes con muchos arcos *(redes densas);*
- el heurístico utilizado para obtener la solución inicial, en la ejecución de los tres algoritmos, es uno de los propuestos por Aragón y Pacheco, [1], (concretamente el conocido como RECHAZO HEURISTICO) combinado con el 3-óptimo de Lin;
- en ningún caso se generan cotas superiores en la exploración de cada vértice, sencillamente se utiliza el valor de la solución obtenida hasta ese momento.

A continuación se muestran los resultados obtenidos. Junto con el tiempo máximo de computación también se muestra el valor de la cota inferior obtenida en el vértice inicial de los algoritmos examinados y el valor del óptimo, (lógicamente el valor de la cota inferior en el vértice inicial para ambas variantes es el mismo ya que parten del mismo valor inicial de t).

Problemas	Tiempos de Computación (seg.)			Cota Inferior Inicial		Valor Optimo
	Variante 1	*Variante 2*	*H. & Karp*	*Variantes*	*H. & Karp*	
n. 1	179.93	171.48	17403.18	174.00	153.00	175
n. 2	360.31	238.49	15921.12	164.00	143.00	164
n. 3	193.72	226.84	23486.20	167.00	139.00	167
n. 4	352.79	261.44	9941.31	150.24	132.00	151
n. 5	404.91	336.86	13940.43	162.00	139.00	162
n. 6	98.75	106.17	11787.98	139.91	120.00	140
n. 7	320.44	334.88	15017.05	162.00	137.00	162
n. 8	120.39	128.09	9699.08	166.98	148.00	168
n. 9	515.92	533.54	17921.08	181.00	158.00	181
n. 10	459.84	408.75	19324.84	161.98	136.00	162
n. 11	215.69	179.44	14061.21	200.11	181.00	201
n. 12	138.35	165.27	22977.46	164.96	148.00	165
n. 13	802.79	241.56	10542.13	162.00	143.00	163
n. 14	264.80	280.23	24110.12	148.00	129.00	149
n. 15	112.98	112.65	16981.18	173.00	150.00	173
n. 16	204.93	248.81	22986.33	135.00	113.00	135
n. 17	369.93	308.02	17098.55	162.97	143.00	163
n. 18	587.37	621.92	23224.45	168.00	142.00	169
n. 19	351.53	312.47	26248.71	167.44	141.00	168
n. 20	298.96	791.97	18611.19	158.00	133.00	158

Estos resultados se resumen en el siguiente cuadro

		Variante 1	*Variante 2*	*Held & Karp*
Tiempo de Computación en segundos	**Media**	317.717	300.444	17564.18
	Mínimo	98.75	106.17	9699.08
	Máximo	802.79	791.97	26248.71
Cota Inferior Inicial		163.43		141.4
Desviación del Optimo de la cota inferior (%)	**Media**	0.22		13.67
	Mínimo	0		9.95
	Máximo	0.67		16.76

En estos resultados hay que hacer una observación: la cota inferior que se muestra en el cuadro de resultados es el valor de wl obtenido en el vértice inicial por los procesos iterativos descritos en cada caso; ahora bien, al ser el valor de la solución óptima un valor entero, se puede tomar como cota inferior el menor entero mayor o igual que wl. Esta nueva cota inferior alcanza el valor óptimo en 15 de los 20 casos, y se desvía de este una media del 0'15%.

5. - CONCLUSIONES

En vista de los resultados obtenidos las conclusiones son evidentes:

- la obtención de una cota inferior ajustada al óptimo en cada vértice, sobre todo el inicial, es fundamental para la reducción del tiempo de computación, especialmente cuando el número de nodos es grande (la obtención de la cota inferior utiliza un tiempo polinomial, mientras que el número de ramas crece exponencialmente);
- las variantes propuestas aportan un método intuitivamente sencillo pero eficaz para la obtención de una buena cota inferior ajustada al óptimo ya que, al variar el parámetro t permite *salir* de máximos locales en los que se puede caer en el proceso iterativo;
- las dos variantes emplean unos tiempos de computación muy parecidos en todos los casos, aunque la segunda variante parece ser ligeramente mejor.

Aunque los resultados son en conjunto satisfactorios, se pueden introducir algunas mejoras. Una de ellas es cambiar o combinar el heurístico utilizado para obtener la solución inicial. El heurístico propuesto por Aragón y Pacheco combinado con el método 3-óptimo de Lin da buenas soluciones, pero en pocas ocasiones (para un número grande de nodos) da la solución óptima. El algoritmo de Lin & Kerninghan o el método de Or son dos heurísticos de reconocida eficacia y que en muchas casos consiguen el óptimo, (en este sentido son interesantes las mejoras introducidas por Nurmi, [14]). De esta forma, en estos casos, en el proceso iterativo para alcanzar la cota inferior inicial se puede *identificar* este óptimo, finalizando la ejecución del algoritmo en la primera rama.

Para un número de nodos superior, otra posible mejora sería generar una cota superior en cada rama. En cualquier caso, en este trabajo, se muestra un posible camino para obtener soluciones óptimas para los problemas de rutas en ordenadores personales, para problemas de gran tamaño.

6.- REFERENCIAS Y BIBLIOGRAFIA

1.- ARAGON,A. y PACHECO,J.
"Desarrollo de Algoritmos Heurísticos Mejorados para el TSP".
VI Congreso de ASEPELT-ESPAÑA, GRANADA 1992.

2.- CHRISTOFIDES,N., MINGOZZI,A. and TOHT,P.
"State-Space Relaxation Procedures for the Computation of Bounds to Routing Problems".
Networks, vol. 11, nº 2, (1.981), 145-164.

3.- CHRISTOFIDES,N. and EILON,S.
"An Algorithm for the Vehicle-Dispatching Problem".
Operational Research Quaterly, vol. 20, nº 3, (1.969), 309-318.

4.- EATSMAN,W.L.
"Linear programming with Pattern Constraints".
PH.D.Thesis. Harvard University. Cambridge, MA, (1.958).

5.- GOLDENB., BODIN,L. DOYLE,T. and STEWART,W.Jr.
"Approximate Traveling Salesman Algorithms".
Operations Research, vol.28, nº 3, parte II, (1.980), 694-711.

6.- HELD,M., and KARP,R.M.
"The Traveling Salesman Problem and Minimum Spanning Trees".
Ops.Res. 18,(1.970), 1.138-1.162.

7.- HELD,M., and KARP,R.M.
"The Traveling Salesman Problem and Minimum Spanning Trees: Part II".
Mathematical Programing I, (1.971), 6-25.

8.- KALANTARI,B., HIIL,A.V. and ARORA, S.R.
"An Algorithm for the Traveling Salesman Problem with Pickup and Delevery Customers".
European Journal of Operational Research, 22 (1.985), 377-386.

9.- KANELLAKIS,P.C., and PAPADIMITRIU,C.H.
"Local Search for the Asymetric Traveling Salesman Problem".
Ops.Res. 28, (1.980), 1.086-1.099.

10.- KRUSKAL, J.B.
"On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem".
Proc.Amer.Math.Soc., 7 (1.956), 48-50.

11. - LIN, S,
"Computer Solutions to the Traveling Salesman Problem".
Bell Syst.Tech.Jou., vol 44, (1.965), 2245-2269.

12. - LIN S. and KERNIGHAN,B.W.
"An Effective Heuristic Algorithm for the Traveling Salesman Problem".
Operations Research, vol.20, (1.973), 498-516.

13.- LITTLE,J., MURTY,K, SWEENEY,D. and KAREL,C.
"An Algorithm for the Traveling Salesman Problem".
Operations Research 11 (5), (1.963), 972-989.

14.- NURMI,K.
"Traveling Salesman Problem Tools for Microcomputers".
Computers & Ops.Res. Vol. 18, nº 8, (1.991), 741-749.

15.- OR,I.
"Travelin,g Salesman Type Combinatorial Problems and their Relations to the Logistics of Blood Banking".
Ph.D. Thesis, Dpt. of Industrial Enginnering and Management Sciences, Northwestern Univ. (1.976).

16.- PACHECO,J.
"Problemas de Rutas con Ventanas de Tiempo".
Tesis Doctoral leída en el Dpto de Estadística e I. Optva de la Facultad de Matemáticas de la U. Complutense de Madrid. Mayo 1.994.

17.- PACHECO,J.
"Modificaciones en el algoritmo de Held & Karp".
XXI Congreso de Estadística e Y. Optva celebrado en Cálella en Abril de 1. 994.

18. - PRIM, R.C.
"Shortest Connection Networks and Some Generalizations".
Bell System Tech.J, 36 (1.957), 1.389-1.401.

19.- SOLOMON,M., BAKER, E. and SCHAFFER,J.
"Vehicle Routing and Scheduling Problems with Time Window Constraints".
In Vehicle Routing.- Methods and Studies, (Studies in Management Sciences and Systems, vol. 16), eds: GOLDEN,B.L. and ASSAD,A.A., (1.988), Nort-Holland, 85-106.

20,- SOLOMON,M.M.
"On the Worst-Case Perfomance of Some Heuristics for the Vehicle Routing and Scheduling Problem with Time Windows Constraints".
Networks, 16, (1.986), 161-174.

21.- SOLOMON,M.M.
"Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints". *Operations Research,* vol.35, nº 2, (1.987), 254-265.

22.- SYSLO,M.M., DEO,N. and KOWALIK, J.S.
Discrete Optimization Algorithms.
Prentice-Hall-Inc, (1.983), Englewood Cliffs.

23.- VOLGENANT,T. and JONKER, R.
"A Branch and Bound Algorithm for the Symmetric Traveling Salesman Problem based on the 1-tree Relaxation".
Eur.J Ops.Res. 9, (1.982), 83-89.

24.- VIEBB,M.H.J.
"Some Methods of Producing Aproximate Solutions to Traveling Salesman Problem with Hundreds or Thousands of Cities".
Ops.Res. Q., 22, (1.971), 49-66.