Rev. Acad. Canar. Cienc., XVIII (Núms. 1-2), 61-70 (2006) (publicado en agosto de 2007)

ANÁLISIS COMPARATIVO DE NUEVOS ALGORITMOS TIPO SIMPLEX PARA PROBLEMAS DE CAMINOS MÍNIMOS

A. Sedeño Noda & C. González Martín
Departamento de Estadística, Investigación Operativa y Computación
Universidad de La Laguna
38271 - La Laguna, Tenerife (España)

Resumen

En este trabajo se realiza un análisis computacional comparativo de algoritmos tipo simplex para resolver problemas de caminos mínimos. Los métodos elegidos tienen la misma complejidad computacional (la mejor cota fuertemente polinomial para el caso general) y han sido aplicados a un mismo conjunto de problemas.

Abstract

In this paper a comparative computational analysis of shortest path algorithms is made. The chosen methods have the same computational complexity (the best polynomial strongly bound for the general case) and they have been applied to the same set of problems.

Palabras Clave: Problemas de caminos mínimos, algoritmos tipo simplex, análisis computacional.

1. Introducción

Entre los problemas estudiados en Programación Combinatoria (y, por extensión, en Investigación Operativa), los de caminos mínimos ocupan un lugar preponderante por su importancia práctica y por la amplia gama de procedimientos que se han propuesto para resolverlos (ver, por ejemplo, Ahuja et al. [1]). Una versión sencilla de este problema plantea la determinación de caminos de longitud mínima entre un determinado vértice (*fuente*) y el resto de los $n-1$ vértices de una red con m aristas (caso no dirigido) o m arcos (caso dirigido).

Entre los algoritmos propuestos para resolverlo en la literatura especializada, se puede hacer una clasificación distinguiendo entre métodos de etiquetas (label methods) y métodos tipo simplex (shortest path simplex methods). En el primer grupo se asocian etiquetas provisionales a los vértices que son cotas superiores de la distancia del correspondiente camino mínimo desde el vértice fuente. En este grupo se pueden distinguir entre algoritmos de fijación de etiquetas (label-setting), en los que se convierte en permanente la etiqueta de uno de los vértices en cada iteración, y los métodos de corrección de etiquetas (label-correcting) en los que las etiquetas usadas son provisionales hasta finalizar.

El representante genuino de los métodos de fijación de etiquetas es el algoritmo de Dijkstra [7]. Este método, con complejidad $O(n^2)$, sólo resuelve problemas de caminos mínimos para redes con longitudes no negativas sobre los arcos.

Entre los algoritmos de corrección de etiquetas sobresale el método de Bellman-Ford-Moore ([3], [8] y [13]) el cual resuelve problemas de caminos mínimos sobre redes con longitudes negativas (detectando, en su caso, circuitos con longitud negativa). Su complejidad es de $O(nm)$ (la mejor cota fuertemente polinomial para el caso general).

La formulación de los problemas de caminos mínimos como problemas de Programación Lineal ha motivado la aparición de distintos procedimientos tipo simplex (ver, por ejemplo, Dantzig [6], Minty [12], Cunningham [5], Orlin [14], Akgül [2], Goldfarb, Hao y Kai [9, 10],...). Hay que destacar que Goldfarb y Jin [11] introducen el primer algoritmo tipo simplex que se ejecuta en un tiempo de $O(nm)$.

En esta línea de trabajo, Sedeño-Noda y González- Martín ([15] y [16]) han propuesto dos nuevos métodos que utilizan, respectivamente, una nueva regla de pivotaje basada en el concepto de *vértice pseudo permanentemente etiquetado* y una nueva regla de pivotaje múltiple. Ambos procedimientos tienen también una complejidad de $O(nm)$.

En este trabajo haremos un estudio comparativo que involucra a los métodos de Goldfarb y Jin y de Sedeño-Noda y González Martín. Después de esta introducción, haremos, en el apartado 2, la formalización del problema e introduciremos los conceptos y las propiedades básicas necesarias para plantear un algoritmo tipo simplex genérico. En los siguientes apartados introduciremos de forma resumida los métodos mencionados y, por último expondremos el correspondiente estudio computacional comparativo.

2. Elementos básicos, formulación y algoritmo genérico

Sea $R=(V,A)$ una red dirigida donde $V=\{1,...,n\}$ es el conjunto de vértices y A es el conjunto de m arcos. Para cada $(i,j)\in A$, sea $c_{ij}\in\mathbb{R}$ su longitud. Un *camino dirigido* (a partir de ahora, *camino*) desde un vértice i a un vértice j permite ir desde el primer vértice al segundo recorriendo una serie de vértices (que no se repiten) y un conjunto de arcos. La longitud de un camino es la suma de las longitudes de sus arcos.

En la versión que estudiamos, distinguimos un vértice s (*vértice o nodo fuente*) en la red. El problema de caminos mínimos se plantea entonces como el de encontrar caminos de longitud mínima entre s y cualquier otro vértice $i\in V\setminus\{s\}$ o, en su caso, detectar un circuito (un camino que empieza y acaba en un mismo vértice) de longitud negativa.

Denotamos por $\Gamma_i^-=\{j\in V\mid (j,i)\in A\}$ y $\Gamma_i^+=\{j\in V\mid (i,j)\in A\}$, $\forall i\in V$, respectivamente, los conjuntos de vértices predecesores y sucesores de un vértice dado.

Si x_{ij} es el flujo asociado con cada arco (i,j), $b_s = 1-n$ y $b_i = 1$, $\forall i \in V \setminus \{s\}$ son las disponibilidades en los distintos vértices, el problema descrito previamente admite la siguiente formulación:

$$\begin{aligned} \min \quad & c(x) = \sum_{(i,j)\in A} c_{ij} x_{ij} \\ \text{s. a:} \quad & \sum_{j\in\Gamma_i^-} x_{ji} - \sum_{j\in\Gamma_i^+} x_{ij} = b_i, \forall i \in V \quad (SPP) \\ & x_{ij} \geq 0, \ \forall (i,j) \in A \end{aligned}$$

Este problema es un caso particular del problema de flujo de costo mínimo y, por tanto, su resolución se puede afrontar con el *método simplex para redes*. En este caso las soluciones básicas son árboles generadores no degenerados $T \subseteq A$ de R (es decir, $x_{ij} > 0$, $\forall (i,j) \in T$) y, por tanto, el método simplex para redes no ejecuta pivotajes degenerados. Además, cada árbol generador factible contiene caminos dirigidos (únicos) desde s (considerado el *vértice raíz*) a cualquier otro vértice (*directed out-spanning trees*). En esta clase de árboles cualquier vértice $i \in V \setminus \{s\}$ tiene un único predecesor, $(pred_i)$, y, por ello, el grado de entrada de cualquier vértice distinto de la raíz es igual a uno. Denotamos por $B_i(T)$ al conjunto de vértices pertenecientes al único camino sobre T desde s hasta i, $\forall i \in V$.

Dado un árbol básico T, se le puede asociar una solución dual $\pi_i(T)$, $\forall i \in V$, haciendo $\pi_s(T) = 0$ y forzando la verificación de la ecuación $c_{ij} - \pi_i(T) + \pi_j(T) = 0 \ \ \forall (i,j) \in T$. Los valores $\pi_i(T)$ son denominados *potenciales* de los vértices y cada uno de ellos es el valor negativo de la distancia sobre T desde s al correspondiente vértice. Por tanto $d_i(T) = -\pi_i(T)$ será la distancia del vértice i, $\forall i \in V$. Por otro lado, definimos los *costos relativos* como $\overline{c}_{ij}(T) = c_{ij} + d_i(T) - d_j(T)$, $\forall (i,j) \in A$.

Es conocido (ver, por ejemplo, Ahuja et al. [1] y Goldfarb et al. [9]) que cualquier solución factible para el problema de caminos mínimos es un punto extremo del poliedro definido por las restricciones de (SPP) y que cada punto extremo de dicho poliedro tiene asociado un único árbol generador de los descritos previamente (*directed out-spanning tree*). Sólo a este tipo de árboles nos referiremos en el resto del trabajo.

Notaremos por $C(T) = \sum_{(i,j)\in T} c_{ij} x_{ij}$ al valor de la función objetivo correspondiente a un árbol T. Además, $\forall i \in V$, $D_i(T)$ contendrá los vértices que están en los diferentes caminos

dirigidos desde i hacia otros vértices de V sobre T. Por tanto, $i \in D_i(T)$ y, como consecuencia, $|D_i(T)| \geq 1$, $\forall i \in V$.

Propiedad 1

i) $\forall (i,j) \in T$, $x_{ij} = |D_j(T)|$ ya que, necesariamente $\forall j \in V$, $x_{pred_j\, j} = \sum_{\{k/(j,k)\in T\}} x_{jk} + 1 = |D_j(T)|$

ii) Si $T \cup \{(i,j)\}$ tiene un circuito, entonces $\overline{c}_{ij}(T) = c_{ij} + d_i(T) - d_j(T)$ es la longitud de dicho circuito.

Como consecuencia de esto $C(T) = \sum_{(i,j)\in T} c_{ij} |D_j(T)|$.

Al aplicar el Método Simplex para Redes, a partir de un árbol básico T se selecciona un arco no básico (i,j) con $\overline{c}_{ij}(T) < 0$. Pueden suceder dos casos:

i) Si $i \in D_j(T)$, entonces existe un circuito de longitud negativa.

ii) En otro caso, (i,j) se convierte en básico y el arco $(pred_j, j)$ deja de ser básico. Para el nuevo árbol, las etiquetas distancia se actualizan sólo cambiando $d_k(T)$ por $d_k(T) + \overline{c}_{ij}(T)$, $\forall k \in D_j(T)$.

Un esquema del citado algoritmo es el siguiente:

Algoritmo Simplex Genérico (**SG**);
Sea T un árbol básico inicial;
Calcular las etiquetas distancias $d_i(T)$, $\forall i \in V$;
Mientras exista un arco $(i,j) \in A \setminus T$ con $\overline{c}_{ij}(T) < 0$, **Hacer**
Selecciona un arco entrante (i, j) con $\overline{c}_{ij}(T) < 0$;
Si $(i \in D_j(T))$, **Entonces Parar**, $T \cup \{(i,j)\}$ contiene un circuito de longitud negativa
En Otro Caso
$T = T \cup \{(i,j)\} \setminus \{(pred_j, j)\}$;
Para todo $k \in D_j(T)$, **Hacer** $d_k(T) = d_k(T) + \overline{c}_{ij}$;

Este método concluye con un árbol óptimo T^* o detectando un circuito de longitud negativa. La aplicación eficiente requiere ejecutar adecuadamente las etapas de actualización

de etiquetas y la detección de circuitos negativos. Precisamente, la fijación de determinadas estrategias o procedimientos para llevar a cabo las operaciones correspondientes, conduce a distintos algoritmos con diferentes órdenes de complejidad computacional.

Nota. Suponemos que en las redes que manejamos existen caminos dirigidos desde s a cualquier otro vértice $i \in V \setminus \{s\}$. Si esto no es cierto, añadimos, cuando es necesario, arcos artificiales (s,i) de longitud infinita. Si estos arcos aparecen en la solución óptima, el problema es no factible.

La definición y las propiedades siguientes aparecen en Goldfarb et al. [9].

Propiedad 2. *En una secuencia de pivotajes con los árboles básicos* $T_0, T_1, ..., T_r, T^*$, *se verifica que* $d_i(T_l) \geq d_i(T_{l+1})$, $\forall i \in V$, $\forall l \in \{0,1,...,r\}$.

Definition 1. Dado un árbol básico T, un vértice $i \in V$ está *permanentemente etiquetado* en T si $d_i(T) = d_i(T^*)$.

Propiedad 3. *Si i está permanentemente etiquetado en T pero j no y, además, $(i,j) \in T^*$, entonces*:

(1) $\overline{c}_{ij}(T) < 0$

(2) *Si G no tiene circuitos negativos, entonces* $\overline{c}_{ij}(T) \leq \overline{c}_{kj}(T)$, $\forall k \in \Gamma_j^-$.

Propiedad 4. *Si j está permanentemente etiquetado en T, entonces* $\overline{c}_{kj}(T) \geq 0$, $\forall k \in \Gamma_j^-$.

La inversa de la propiedad 3 no es cierta. Esto motiva la siguiente definición que aparece en Sedeño-Noda y González-Martín [16]:

Definición 2. *Un vértice $j \in V$ está pseudo permanentemente etiquetado en T si, y sólo si,* $\overline{c}_{ki}(T) \geq 0$, $\forall k \in \Gamma_i^-$, $\forall i \in B_j(T)$.

A partir de la definición 2 es evidente que un vértice $j \in V$ está pseudo permanentemente etiquetado si, y sólo si, todos los vértices del camino sobre T desde s a j están pseudo permanentemente etiquetados.

3. Método Simplex con pivotaje múltiple

Este método, propuesto por Sedeño-Noda y González-Martín [15], ejecuta O(n) pivotajes múltiples con una complejidad total de O(nm).

Es conocido que dos árboles T y T' son adyacentes cuando difieren en un arco, es decir, cuando tienen $n-2$ arcos en común. Por tanto T' puede obtenerse a partir de T realizando una operación de pivotaje sobre el arco $(i,j) \in T' \setminus T$.

Cuando T y T' difieren en los $p < n$ arcos: $(i_1, j_1), \ldots, (i_p, j_p)$, se verifica la propiedad siguiente (Sedeño-Noda y González Martín [15])

Proposición . *Si T y T' difieren en los arcos $(i_1, j_1), \ldots, (i_p, j_p)$, entonces:*

(1) $(i_1, j_1), \ldots, (i_p, j_p)$ *no contienen circuitos*

(2) $j_u \neq j_v$ *se verifica,* $\forall u, v \in \{1, \ldots, p\}$ *con* $u \neq v$

(3) *Estos arcos definen la secuencia más corta de pivotajes necesarios para obtener T' a partir de T, siendo irrelevante el orden en que estos pivotajes son ejecutados.*

Esta proposición indica que para obtener un árbol óptimo T^* desde un árbol T, son necesarios un número $p < n$ de pivotajes sobre los arcos $(i_1, j_1), \ldots, (i_p, j_p)$, verificando las condiciones expuestas. Evidentemente, esta consideración abre la puerta a la ejecución de un pivotaje múltiple.

Sin embargo, ¿cuáles son los arcos que se deben seleccionar? La respuesta tiene la misma dificultad que la resolución del problema en estudio. No obstante, la proposición anterior permite identificar conjuntos de arcos que mejoran la función objetivo del problema cuando participan en un pivotaje múltiple. Al respecto, en [15] se hace un estudio pormenorizado de distintas opciones de que conducen al siguiente procedimiento de selección de arcos:

Procedimiento Identificando_Arcos_Entrantes (i, C);
    $Visitado = Visitado + 1$;
    $Mejor_i = Null$;
    $Min_i = C$;
    Para todo $j \in \Gamma_i^-$ **Hacer**
        Si $\overline{c}_{ji}(T) < Min_i$ **Entonces**
            $Min_i = \overline{c}_{ji}(T)$;
            $Mejor_i = j$
    Para todo $j \in T_i^+$ **Hacer**
        Identificando_Arcos_Entrantes (j, Min_i)

Este procedimiento recursivo genera un vector de punteros, *Mejor,* que permite identificar los arcos $(Mejor_i, i)$ para ser convertidos en básicos aplicando la regla:

Si $\overline{c}_{i_k j_k}(T) < \min_{j_{k'} \in B_{j_k}(T)} \left\{0, \overline{c}_{i_{k'} j_{k'}}(T)\right\}$ *entonces el arco* (i_k, j_k) *es seleccionado para un pivotaje múltiple*

El siguiente algoritmo hace uso efectivo del procedimiento anterior:

Algoritmo Simplex con Pivoteo Múltiple (**SPM**);
Sea T un árbol básico inicial;
Calcular las etiquetas distancias $d_i(T)$ para todo $i \in V$;
$Optimo$ = **Falso**; $k = 2$;
Mientras ($k < n$) **y no** ($Optimo$), **Hacer**
$Visitado = 0$; Identificando_Arcos_Entrantes(s, 0);
Si ($Visitado < n$), **Entonces Parar:** G contiene un circuito negativo;
En Otro Caso
$Pivoteos = 0$;
Para todo $i \in V$ **Hacer**
Si $Mejor_i <> Nulo$, **entonces**
$T = T + \{(Mejor_i, i)\} - \{pred_i, i\}$; $Pivoteos = Pivoteos + 1$;
Si ($i == s$) **o** ($Pivoteos > n - k$), **Entonces Parar:** G contiene un circuito negativo;
Si ($Pivoteos == 0$), **Entonces** $Optimo$ = **Verdadero**;
Calcular las etiquetas distancias $d_i(T)$ para todo $i \in V$;
$k = k + 1$;

El algoritmo anterior implementa un regla de selección de arcos que es una modificación de la regla de pivoteo *inward most negative* (IMN) (entrante más negativo) analizada por Cunningham [5] y Goldfarb et al [9]. Además, en Goldfarb y Jin [11], también se introduce una versión de la regla de pivoteo IMN donde el orden en el que se examinan los nodos es el denominado preorden inverso. Si en el anterior algoritmo incluimos adecuadamente esta regla obtendremos el método de Goldfarb y Jin [11] denominado como **G&J**.

4. Método Simplex con una nueva regla de pivotaje

Para un árbol T, sea $L(T)$ el conjunto de nodos pseudo permanentemente etiquetados. Se tiene, entonces, una partición el conjunto de vértices en los subconjuntos $L(T)$ y $\overline{L}(T)$.

$\forall i \in \overline{L}(T)$, sea $L_i(T) = \left\{ j \in \Gamma_i^- \cap L(T) \right\}$ y $(j^*, i) = \arg\min \left\{ \overline{c}_{ji}(T) \mid j \in L_i(T) \right\}$. Notamos por $H(T) = \left\{ i \in \overline{L}(T) \mid \overline{c}_{j^* i}(T) < 0 \right\}$.

Se demuestra en [16] (lemas 1 y 2) que si la red no contiene circuitos negativos y se ejecuta un pivotaje simplex sobre un arco (j^*, i) con $i \in H(T)$, entonces al menos un vértice de $H(T)$ se convierte en pseudo permanentemente etiquetado y al menos un vértice de $H(T)$ se convierte en permanentemente etiquetado. Además (ver lema 3 de [16]), los conjuntos $L(T)$ y $H(T)$ permiten detectar la presencia de circuitos negativos.

El uso efectivo de los conceptos y resultados anteriores motivan el siguiente algoritmo:

Algoritmo Simplex Eficiente (**SE**)

Sea $T := T_0$ el árbol básico inicial; $k = 2$;
Calcula las etiquetas distancias $d_i(T)$ para todo $i \in V - \{s\}$ con $d_s(T) := 0$;
Sea $L(T)$ el conjunto de nodos pseudo permanentemente etiquetados;
$\overline{L}(T) := V \setminus L(T)$;
Mientras $(k \leq |L(T)| < n)$, **Hacer**
Para todo $i \in \overline{L}(T)$, **Hacer** $(j^*, i) = \arg \left\{ \overline{c}_{j^* i}(T) = \min \left\{ \overline{c}_{ji}(T) \mid j \in L_i \right\} \right\}$;
Sea $H(T) = \left\{ i \in \overline{L}(T) \mid \overline{c}_{j^* i}(T) < 0 \right\}$;
Si $(|H(T)| = \varnothing)$, **Entonces Parar:** G contiene un circuito negativo;
En Otro Caso
Para todo $i \in H(T)$, **Hacer**
$T := T + \left\{ (j^*, i) \right\} - \left\{ pred_i, i \right\}$;
Calcula las etiquetas distancias $d_i(T)$ para todo $i \in V - \{s\}$ con $d_s(T) := 0$;
Sea $L(T)$ el conjunto de nodos pseudo permanentemente etiquetados;
$\overline{L}(T) := V \setminus L(T)$;
$k := k + 1$;
Si $(|L(T)| < k)$, **Entonces Parar:** G contiene un circuito negativo;
En Otro Caso T es un árbol óptimo;

Nota: Se demuestra en Goldfarb y Jin [11] y en Sedeño Noda y González-Martín [15] y [16] que los algoritmos G&J, SPM y SE, respectivamente, se ejecutan en un tiempo $O(nm)$.

5. Análisis computacional comparativo

En lo que respecta al experimento computacional que hemos realizado, hemos de indicar que los algoritmos SPM, SE and G&J han sido escritos en C++ y ejecutados sobre Red Hat Linux en un ordenador con procesador ADM Athlon a 1,41 GHz. Los problemas tests se

obtuvieron mediante el generador SPRAND desarrollado por Cherkassky et al. [4] utilizando los siguientes parámetros de las redes: el número de nodos $n \in \{1000, 2000, ..., 10000\}$, el número de arcos $m \in \{5n, 50n, 500n\}$ y la longitud de los arcos uniformemente distribuida en el intervalo $[0, l]$ con $l \in \{10, 100, 100000, 100000000\}$. Para cada combinación de los parámetros de redes anteriores (120), se generaron diez réplicas usando las siguientes semillas: 12345678, 36581249, 23456183, 46545174, 35826749, 43657679, 378484689, 23434767, 56563897 y 78656756. En total se han estudiado 1200 problemas.

En la Tabla 1 se muestran el número de nodos (n), el tiempo de CPU en segundos y el número de pivoteos realizados por los algoritmos, promediados en el número de réplicas y en las longitudes máximas de los arcos y agrupados según el cociente m/n.

	$5 \le m/n \le 25$						$35 \le m/n \le 50$						$150 \le m/n \le 450$					
	SE		SPM		G&J		SE		SPM		G&J		SE		SPM		G&J	
n	CPU	Pivoteos	CPU	Pivoteos	CPU	Pivoteos	CPU	Pivoteos	CPU	Pivoteos	CPU	Pivoteos	CPU	Pivoteos	CPU	Pivoteos	CPU	Pivoteos
1000	0,03	2591,58	0,03	2246,63	0,03	2413,83	0,06	3219,14	0,05	2726,72	0,05	2320,25	0,64	3074,73	0,53	2535,01	0,43	2078,01
2000	0,08	5394,67	0,05	4614,98	0,08	5193,89	0,20	6750,07	0,15	5591,08	0,16	5047,11	1,55	6791,46	1,23	5522,21	1,02	4474,83
3000	0,15	8260,46	0,10	7079,88	0,19	8212,82	0,40	10384,18	0,29	8524,59	0,32	7969,28	2,46	10666,71	1,98	8659,54	1,67	6909,38
4000	0,25	11118,78	0,16	9492,07	0,32	11249,54	0,64	13952,73	0,44	11383,30	0,52	10693,18	3,46	14725,61	2,78	11955,3	2,37	9532,97
5000	0,39	14222,53	0,23	12117,16	0,51	14472,47	0,92	17973,00	0,62	14569,96	0,77	13879,33	4,54	18871,86	3,63	15337,8	3,09	12142,16
6000	0,54	17269,78	0,32	14668,02	0,74	17680,98	1,22	21715,18	0,82	17616,69	1,05	16768,76	5,72	23475,50	4,57	18936,9	3,89	15228,04
7000	0,73	20777,70	0,41	17516,77	1,02	21841,42	1,60	26156,62	1,03	20939,50	1,42	20822,49	6,88	27654,41	5,43	22318,8	4,69	17788,16
8000	0,93	24031,59	0,50	20158,30	1,30	24499,58	1,97	30831,70	1,24	24531,09	1,79	24209,95	8,18	32425,71	6,43	26111,4	5,63	20820,83
9000	1,11	26399,20	0,60	22295,40	1,59	27199,32	2,33	33994,95	1,44	27250,47	2,10	26535,39	9,29	36327,57	7,32	29363,5	6,41	23362,79
10000	1,38	30230,11	0,71	25256,61	2,02	31886,49	2,79	39003,28	1,66	30973,48	2,59	31267,38	10,65	40954,42	8,32	33042,5	7,48	26085,39

Los criterios de clasificación utilizados son el tiempo de CPU y el número de pivoteos. El sentido decreciente de ambos implica mayor eficiencia.

Si entendemos que a medida que crece el cociente m/n las redes tienen una mayor densidad, podemos clasificar los problemas del estudio en poco densos (primera columna), medianamente densos (segunda columna) y muy densos (tercera columna). Un simple estudio descriptivo revela que para redes densas y medianamente densas el tiempo de CPU utilizado por SPM es menor que el utilizado por SE y por G&J. Por tanto, para estos casos SPM es el algoritmo más eficiente. Este comportamiento de SPM se reproduce también para el número de pivoteos sobre problemas poco densos. Además, para redes poco densas, el método SE mejora a G&J, respecto al tiempo de CPU y el número de pivoteos, cuando crece el número de vértices. También se observa que, para redes muy densas, el orden decreciente de eficiencia de los algoritmos, en relación con los dos criterios, es: G&J, SPM y SE.

Agradecimientos: Este trabajo está parcialmente financiado por los proyectos MTM2006-10170 (Ministerio de Educación y Ciencia con la colaboración de los Fondos Europeos para el Desarrollo Regional) y PI042004/078 (Consejería de Educación del Gobierno de Canarias).

6. Referencias

[1] Ahuja, R., Magnanti, T. y Orlin, J. B. (1993). *Network Flows*. Prentice-Hall, Inc.

[2] Akgül, M., "Shortest Paths and the Simplex Method", Tecnichal Report, Department of Computer Sciences and Operations Research Program, North Carolina State University, Raleigh, N.C. (1986).

[3] Bellman, R., "On a Route Problem", *Quart. Of Appl. Math.* 16 (1958), 87-90.

[4] Cherkassky, B. V., Goldberg, A. V. y Radzik, T., "Shortest paths algorithms: Theory and experimental evaluation", *Mathematical Programming* **73** (1996), 129-174.

[5] Cunningham, W. H., "Theoretical Properties of the Network Simplex Method", *Mathematical Programming* 4 (1979), 196-208.

[6] Dantzig, G. B., "Discrete-variable extremum principles", *Operations Research* 5 (1957), 266-277.

[7] Dijkstra E. W. "A note on two problems in connection with graphs". *Numer. Math.* **1** (1959), 269-271.

[8] Ford, L. R. (1956). *Network Flow Theory*. The Rand Corporation Report P-923, Santa Monica, Calif.

[9] Goldfarb D., Hao, J. y Kai, S.R., "Efficient Shortest Path Simplex Algorithms", *Operations Research* 38 (4) (1990), 624-628.

[10] Goldfarb D., Hao, J. y Kai, S.R., "Anti-stalling Pivot Rules for the Network Simplex Algorithm", *Networks* 20 (1990), 79-91.

[11] Goldfarb D. y Jin, Z., "An O(*nm*)-Time Network Simplex Algorithm for the Shortest Path Problem", *Operations Research* 47 (3) (1999), 445-448.

[12] Minty, G. J., "A Variant on the Shortest Route Problem", *Operations Research* 6 (1958), 882.

[13] Moore, Z. F., "The Shortest Path Through a Maze", *In Proceedings of the International Symposium on Theory of Switching,* Part II (1957), 285-292.

[14] Orlin, J. B. "On the Simplex Algorithm for Networks and Generalized Networks", *Mathematical Programming Study* 25 (1985), 166-178.

[15] Sedeño-Noda A., C. González-Martín. "Shortest path simplex algorithm with a multiple pivot rule". Technical Report Nº 2., DEIOC, Universidad de La Laguna. (2006). Enviado a *EJOR*.

[16] Sedeño-Noda A., C. González-Martín. "A new efficient shortest path simplex algorithm". Technical Report Nº 3. DEIOC, Universidad de La Laguna. (2006). Enviado a *Comp. Optim. and Appl*.