QÜESTIIÓ, vol. 24, 1, p. 55-82, 2000

DISEÑO DE ALGORITMOS PARA EL PROBLEMA DEL TRANSPORTE ESCOLAR. APLICACIÓN EN LA PROVINCIA DE BURGOS

J.A. PACHECO
A. ARAGÓN
C. DELGADO
Universidad de Burgos*

La problemática del transporte escolar es en Burgos especialmente significativa al ser una provincia extensa con muchos núcleos de población muy dispersos y poco poblados. En este trabajo se describen las aportaciones realizadas por los autores para dar solución a dicho problema, a través de técnicas que den soluciones lo más racionales posibles. En este sentido, hay que indicar que el término de racionalidad no sólo hace referencia a la minimización del coste total del transporte, sino también al cuidado de determinados aspectos como el tiempo que permanecen los alumnos en el vehículo, la elección de carreteras cómodas, etc., en especial a partir de determinados acontecimientos recientes. Asimismo, se muestran los resultados obtenidos con los datos del actual curso.

Algorithm design for school transportation. Application to Burgos province

Palabras clave: Transporte escolar, VRP, VRPTW, búsqueda tabú

Clasificación AMS (MSC 2000): 90B20

* Departamento de Economía Aplicada. Universidad de Burgos. Parralillos, s/n. 09001 Burgos.

– Recibido en junio de 1999.
– Aceptado en enero de 2000.

1. INTRODUCCIÓN

Considérese el problema del transporte de un conjunto de alumnos que han de ser recogidos en una serie de localizaciones distribuidas geográficamente y llevados a un centro de enseñanza. Se denota por 1 el punto correspondiente al centro de enseñanza, y por $2,\ldots,n$ los puntos correspondientes a las localizaciones donde se recogen los alumnos. Sea $q(i)$ el número de alumnos que se recogen en cada punto $i, i = 2,\ldots,n$. Para cumplir estos requerimientos la legislación autoriza diferentes tipos de vehículos; cada tipo de vehículo con un número de plazas diferente. Para cada alumno el tiempo que trascurre desde que sube al autobús y entra en clase no debe sobrepasar de un determinado tiempo máximo, que denotamos por *tmax*, y la ruta debe finalizar antes del inicio de las clases en el instante *tinicio*. Las distancias d_{ij} y tiempos t_{ij} entre cada par de puntos $i, j \in \{1,2,\ldots,n\}$ son conocidas. El número de tipos de vehículos autorizados se denota por *ntipos* y las capacidades por *capactipo(i)*, $i = 1,\ldots,$*ntipos*.

Se ha de diseñar un conjunto de rutas de coste mínimo, verificando que se respeten los horarios y tiempos de conducción, y que el número de alumnos transportados en cada ruta sea inferior a la capacidad del vehículo asignado a dicha ruta.

Además, se va a imponer la restricción de que los alumnos de una determinada localización sean transportados por un solo vehículo. (En el caso en que en una localización el número de alumnos de ésta superen la máxima capacidad de todos los tipos de vehículo, se trataría dicha localización como dos diferentes: una con un número de alumnos igual a dicha capacidad, y otra con el resto. De todas formas, este caso no se ha dado en los datos reales analizados.)

El coste de transporte de cada ruta viene dado básicamente por el número de kilómetros recorridos, aunque también interviene el número de alumnos transportados y el número de paradas. En concreto, recientemente (13 de diciembre de 1997), el Ministerio de Educación y Cultura, a través de la Secretaría General de Educación y Formación Profesional, sugirió una fórmula que podría servir de referencia para el cálculo de las cantidades necesarias para las contrataciones de las rutas del transporte escolar. Estas cantidades (en pesetas) vendrían descompuestas en los 3 apartados antes mencionados (kilómetros, alumnos, paradas) de la siguiente forma:

– Coste por kilómetros $= \begin{cases} Pr * k & \text{si } k \leqslant 35 \\ Pr * 35 + (k-35) * (1'33) * Pr & \text{si } k > 35 \end{cases}$

donde Pr es el precio de referencia por Km. que oscila entre 125 y 163 (en función de la calidad del vehículo), y k el nº de Kms. La cantidad por este concepto no podrá exceder de 14.250.

– Coste por alumnos $= \begin{cases} 100 * (M-33) & \text{si } M > 33 \\ 0 & \text{si } M \leqslant 33 \end{cases}$

donde M es el nº de alumnos.

– Coste por paradas $= 75 * (L1 - 6)$, si $L1 > 6$
0 si $L1 \leqslant 6$

siendo $L1 = \text{mín}\{L, M/3\}$ y L el número de paradas.

Sin embargo, en otros casos los responsables en cada provincia han de negociar las cantidades por diferentes sistemas de tarifas. Es decir, se paga por kilómetro recorrido, y dicho precio por kilómetro depende del número de alumnos transportados en la ruta. En cualquier caso, la fórmula anteriormente diseñada supone una buena aproximación y es la que utilizaremos en este trabajo.

Finalmente, en cuanto al coste se ha de mencionar que la distancia recorrida comienza a contar desde el primer punto de recogida, y no desde el punto en el que sale el autobús[1], y en cualquier caso no hay un coste fijo en cada ruta por el tipo de vehículo utilizado. Esto es importante en el diseño de las estrategias de solución, puesto que de esta forma no va a ser necesario minimizar el número de vehículos a utilizar, es más, se ha observado que, en muchos casos, aumentando el número de rutas se puede llegar a soluciones menos costosas.

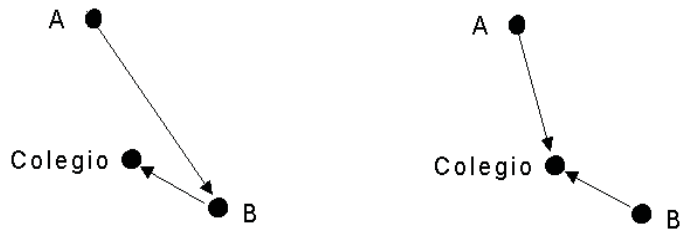


Figura 1. Al no haber coste fijo por vehículo, y al comenzar a contar desde el primer punto de recogida (origen y destino no coinciden), es fácil encontrar soluciones que con más rutas son más baratas que otras con menos. (La solución con las rutas A-Colegio y B-Colegio, recorre menos distancias que la solución con la ruta A - B - Colegio.)

Así considerado, este modelo es un caso particular del conocido *Problema de Rutas de Vehículos* o VRP (Vehicle Routing Problem) o, para ser más precisos, es un *Problema de Rutas de Vehículos con Restricciones de carga y tiempo* (ver Laporte y otros (1984) y (1985)).

Existen muchos algoritmos de solución para el VRP (y/o de variantes, principalmente del VRPTW) en la literatura. Se pueden encontrar recopilaciones de los principales en

[1] Obviamente a partir de ahora $d_{1i} = t_{1i} = 0$, para $i = 2, \ldots, n$.

trabajos como los de Bodin y Golden (1981), Desrochers y otros (1988), Haouri y otros (1990), Laporte (1992) y Laporte y Osman, (1995). En los últimos años han tomado importancia el desarrollo de algoritmos basados en procesos denominados Metaheurísticos como *Algoritmos Genéticos, Temple Simulado, Búsqueda Tabú, GRASP, Búsqueda Local Guiada (GLS), Colonias de Hormigas, etc.*, especialmente a partir de los trabajos de Gendreau y otros (1991), (1994) en versión posterior, y de Osman (1993); y más recientemente en los de Potvin y otros (1993) y (1994), Thangiah y otros (1993) y (1994), Campos y Mota (1995), Kantoravdis (1995), Rochat (1995), Kilby y otros (1997), Backer y otros (1997), Bullnheimer y otros (1997), o Rego (1998).

Al ser éste un problema NP-Hard (complejidad no Polinomial, ver Lenstra et al. (1981)), los algoritmos exactos (es decir, aquellos que garantizan la solución óptima) requieren un tiempo de computación que crece de forma exponencial con el número de elementos que intervienen en el problema. Por tanto, el uso de estos algoritmos puede requerir un tiempo de computación excesivo, incluso en problemas no muy grandes (menos de 100 puntos) cuando, como es el caso, se resuelven en ordenadores personales.

Por tanto, se va a optar por el diseño de una técnica heurística, es decir, que no garantiza la obtención del óptimo, pero sí una buena solución en un tiempo de computación más razonable. Esta estrategia está basada en dos partes: la primera consiste básicamente en una serie de procesos de Búsqueda Local, que dan soluciones rápidamente; la segunda (opcional según el tiempo de cálculo disponible), está basada en un proceso de Búsqueda Tabú que lleva incorporado un novedoso método de Intensificación. Por tanto, el algoritmo propuesto es una técnica compuesta por varias partes o subalgoritmos, basados en su mayor parte en movimientos vecinales.

En las dos siguientes secciones se describen las diferentes definiciones de vecindarios y un método para calcularlos. En la cuarta sección se describe la estructura del algoritmo general. En la quinta se describe un algoritmo basado en un proceso de Búsqueda Tabú que complementa el algoritmo anterior y que puede mejorar en muchas ocasiones la solución obtenida. Finalmente, en la sexta se describen los resultados obtenidos por los diferentes algoritmos y subalgoritmos, así como los tiempos de computación usados para una serie de experiencias en problemas reales con datos obtenidos en la provincia de Burgos.

A partir de ahora se denotará por S el conjunto de soluciones factibles del problema, y f la función de costes a minimizar definida en S.

2. CONSTRUCCIÓN DE SOLUCIONES VECINAS

Se van a considerar dos tipos de soluciones vecinas, una basada en la idea propuesta por Or (1976), para el Problema del Viajante o TSP, y otra más reciente que extiende el usado en los trabajos de Gendreau et al (1991), Osman (1993), Campos y Mota

(1995), etc., que se explicará en la siguiente sección. (En los trabajos de Taillard y otros (1995) y (1997), se definen interesantes estructuras vecinales parecidas aunque algo más complejas.)

2.1. Vecindario tipo Or

En este subapartado se van a definir como soluciones vecinas las obtenidas por el método de intercambio propuesto por Or (1976) que sean factibles. El método de intercambio Or es una variante de los conocidos intercambios r-óptimos desarrollados por Lin, (1965) y Lin & Kernighan (1973) para el TSP simétrico. Como se verá mas adelante, el método de Or se puede utilizar en problemas asimétricos. La eficacia de este método para el TSP ha sido contrastada en trabajos como el de Nurmi (1991).

Obsérvese que una solución puede expresarse de forma sencilla, como una única secuencia de puntos; por ejemplo, la solución formada por las dos rutas siguientes:

$$\textit{ruta 1:}\ 1-3-5-1; \quad \text{y} \quad \textit{ruta 2:}\ 1-4-2-1$$

puede expresarse como:

$$1-3-5-1-4-2-1$$

los '1' representan la vuelta de un vehículo al origen y la salida del siguiente (obviamente el último y el primer elemento siempre serán '1'). De esta forma, como se ilustra a continuación, se puede aplicar los Or-intercambios para obtener soluciones vecinas de la actual, considerando a ésta como una única secuencia.

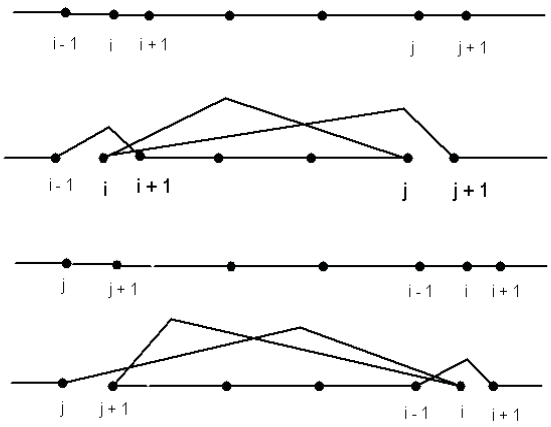


Figura 2. Posible recolocación del elemento *i* hacia adelante y hacia atrás entre j y $j+1$.

Or propone restringir la búsqueda de intercambios a los *3-intercambios* en los que cadenas[2] de uno, dos o tres puntos consecutivos son recolocadas entre otras dos. Nótese que con estos intercambios no se cambia el sentido de los diferentes tramos.

En nuestro caso seguiremos la misma idea, pero sólo consideraremos recolocaciones hacia adelante (se ha observado que considerando también las recolocaciones hacia atrás apenas hay diferencias en los resultados finales y el tiempo de computación es el doble); además, se debe chequear la factibilidad de cada posible recolocación respecto a las restricciones del problema. A continuación se ilustra la recolocación de una cadena de *k* elementos comenzando en *i* entre j y $j+1$.

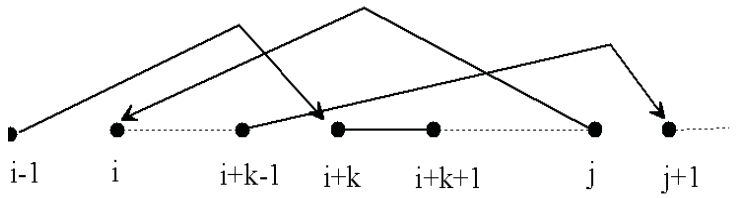


Figura 3. Recolocación de una cadena de *k* elementos.

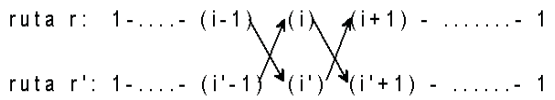
Para cada $s \in S$ se va a denotar por $N_1^k(s)$ al conjunto de soluciones factibles obtenidas por recolocaciones hacia adelante de cadenas de a lo sumo *k* elementos en *s*. Se denota por $N_1^\infty(s)$ el conjunto de soluciones factibles obtenidas por todas las recolocaciones.

2.2. Vecindarios tipo Gendreau-Clarke

El segundo tipo de vecindarios no considera las soluciones como una única secuencia de puntos, sino que sólo considera 3 tipos de intercambios entre 2 rutas diferentes:

– Tipo I: Intercambio del elemento *i* de la ruta *r* y con el elemento i' de la ruta r':

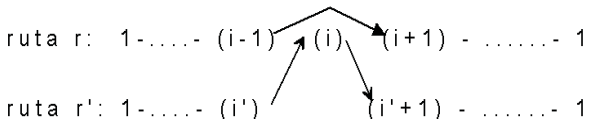
- Eliminación de los arcos $(i-1,i),(i,i+1),(i'-1,i'),(i',i'+1)$
- Incorporación de los arcos $(i-1,i'),(i',i+1),(i'-1,i),(i,i'+1)$



[2]En este trabajo se denomina cadena a toda secuencia de puntos consecutivos en la solución actual.

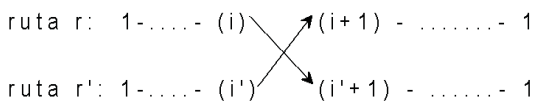
– Tipo II: Inserción del elemento i de la ruta r entre los elementos i' e $i'+1$ de la ruta r':

  - Eliminación de los arcos $(i-1,i),(i,i+1),(i',i'+1)$
  - Incorporación de los arcos $(i',i),(i,i'+1),(i-1,i+1)$



– Tipo III: Cruce de las rutas r y r' por los elementos i e i' según la figura:

  - Eliminación de los arcos $(i,i+1),(i'i'+1)$
  - Incorporación de los arcos $(i',i+1),(i,i'+1)$.



Los dos primeros tipos aparecen en el trabajo de Gendreau et al. (1991) y otros mencionados anteriormente; el tercero, sin embargo, no aparece en dichos trabajos: se basa en algunas de las ideas propuestas por Clarke and Wright (1964).

Obsérvese que el tipo I es en realidad un 4-intercambio, el tipo II un 3-intercambio y el tipo III un 2-intercambio. Para cada $s \in S$ se denotará a los conjuntos de soluciones factibles generadas por cada uno de estos 3 tipos de intercambio como $N_2^1(s), N_2^2(s)$ y $N_2^3(s)$ respectivamente; asimismo, se denota por $N_2(s)$ la unión de estos 3 conjuntos.

Una vez descritos los tipos de vecindarios que se van a utilizar, se deben hacer las siguientes puntualizaciones:

– En los subapartados 2.1. y 2.2. anteriores se han considerado rutas como ciclos que comienzan y acaban en 1, pero en la descripción del problema las rutas son caminos abiertos que comienzan en alguna población donde se recogen a los primeros niños y finaliza en el colegio 1. Sin embargo, como se indica en la descripción del problema, se va a considerar o redefinir $d_{1i} = t_{1i} = 0$, para $i = 2,\ldots,n$; lo que nos permite considerar a las rutas como ciclos (aunque en realidad sean caminos abiertos) y esto, al menos para los autores, resulta más cómodo para la programación de los algoritmos.

– A las rutas que componen una solución s se añade una ruta ficticia vacía $(1-1)$. El objeto de esto es permitir obtener soluciones vecinas con más rutas reales (no vacías) añadiendo en la ruta ficticia elementos de las otras rutas. Recuérdese, como se dijo en la introducción, que a veces se consiguen mejores soluciones con más rutas.

El chequeo de la factibilidad y la valoración de cada intercambio (tanto de tipo Or como de tipo I, II o III) puede suponer un tiempo de computación excesivo. En los trabajos de Pacheco y Delgado (1996) y (1997) se propone el uso de variables globales, con lo que el número de operaciones para chequear y evaluar cada intercambio es constante, es decir, independiente del tamaño del problema. En la siguiente sección se intenta resumir las ideas básicas de estos trabajos.

3. FACTIBILIDAD Y VALORACIÓN DE CADA INTERCAMBIO

El chequeo de la factibilidad y la valoración de cada intercambio (tanto de tipo Or como de tipo I, II o III) puede suponer un tiempo de computación excesivo. Valorar un cambio en el tipo de problema que se está estudiando, no sólo consiste en calcular la diferencia entre los costes de los arcos que se añaden y los que se quitan: hay que determinar también y tener en cuenta los nuevos tipos de vehículos requeridos. En los trabajos de Pacheco y Delgado (1996) y (1997) se propone el uso de variables globales, con las que el número de operaciones para chequear y evaluar cada intercambio es constante, es decir, independiente del tamaño del problema.

3.1. Valoración y factibilidad respecto a la carga

Sea una ruta de *nr* puntos $r(1)-r(2)-\cdots-r(nr-1)-r(nr)$, (obviamente $r(nr)=r(1)=1$). Para calcular los espacios requeridos tras cada intercambio se define:

– *esp_ocup(s)* como la carga en el vehículo después de visitar el punto $r(s)$, para $s=2,\ldots,nr$;

asimismo sea:

– *cadena* = $r(p)-r(p+1)-\cdots-r(q)$, cualquier cadena de la solución actual,

se define

– *maximo_esp_ocup(p,q)* como la máxima carga en el vehículo durante la visita a los puntos de cadena;

obviamente

– *maximo_esp_ocup(p,q)* = $\max_{s=p,\ldots,q}$ *esp_ocup(s)* y
– *maximo_esp_ocup(p,q)* = $\min\{$*maximo_esp_ocup*$(p,q-1)$, *esp_ocup(q)*$\}$, con
– *maximo_esp_ocup(p,p)* = *esp_ocup(p)* como valor inicial.

4.2. Factibilidad respecto a las ventanas de tiempo

Obsérvese que los valores *tmax* y *tinicio*, definidos en la introducción, implícitamente definen a su vez unos intervalos de visita en cada punto $i, [e,l]$, donde *e = tinicio-tmax*, y *l = tinicio*, para $i = 1,\ldots,nr$.

Para todo $s = 2,\ldots,nr$ se van a definir las siguientes variables, inspiradas en el trabajo de Savelsberg (1985):

- $A(s)$: Tiempo de llegada del vehículo a $r(s)$,
- $D(s)$: Tiempo de salida de $r(s)$; se tiene que $D(s) = \text{máx}\{A(s), e\}$;
- $M(s)$: Margen de tiempo en la llegada a $r(s)$, es decir, $M(s) = l - A(s)$;
- $E(s)$: Tiempo de espera del vehículo en $r(s)$, es decir, $E(s) = \text{máx}\{e - A(s), 0\}$.

A continuación se muestra un gráfico con un ejemplo que ilustra estas definiciones.

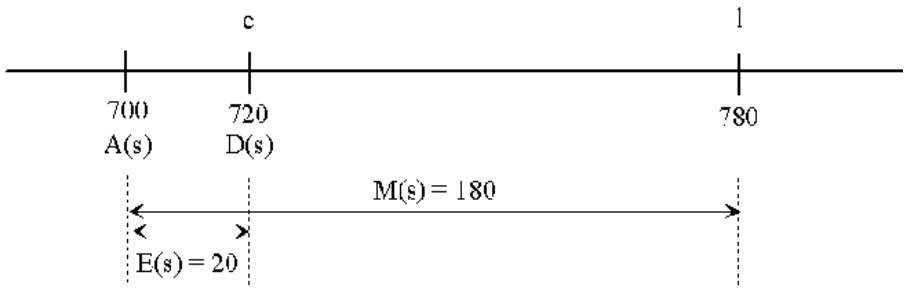


Figura 6. Ejemplo de la llegada del vehículo a un punto $r(s)$, con er(s)=720 y lr(s)=780

Si la llegada $A(s)$ se produce en 700, entonces el tiempo de salida es $D(s)$ = 720, el tiempo de espera $E(s)$, es 20, y el margen, $M(s)$, es 80.

Proposición 1. *Para cualquier $p = 1,\ldots,nr$, considérese un retardo en la llegada a $r(p)$ de x unidades de tiempo; sea A^* y D^* los nuevos tiempos de llegada y salida que este incremento produce en los puntos siguientes de la solución, se tiene que:*

$$A^*(p+h) = A(p+h) + \text{máx}\left\{x - \sum_{s=p,\ldots,p+h-1} E(s), 0\right\}, \quad \text{para } h = 0,\ldots,nr-p.$$

Demostración. Por inducción en los valores de h:

- para $h = 0$ es trivial: $A^*(p) = A(p) + x$.
- sea cierto para $h-1$, entonces,

$$A^*(p+h-1) = A(p+h-1) + \text{máx}\left\{x - \sum_{s=p,\ldots,p+h-2} E(s), 0\right\}.$$

■

Observando la figura 6 anterior y teniendo presente las relaciones entre A, D y E, es fácil de comprobar que un retardo en la llegada a un punto, repercute en el tiempo de salida en dicho punto, si este retardo es mayor o igual que el tiempo de espera; en este caso el aumento en el tiempo de salida será igual a la diferencia entre el retardo y el tiempo de espera; por tanto:

$$\begin{aligned} D^*(p+h-1) &= D(p+h-1) + \text{máx}\left\{\text{máx}\left\{x - \sum_{s=p,\ldots,p+h-2} E(s), 0\right\} - E(p+h-1), 0\right\} = \\ &= D(p+h-1) + \text{máx}\left\{x - \sum_{s=p,\ldots,p+h-1} E(s), 0\right\} \end{aligned}$$

luego:

$$\begin{aligned} A^*(p+h) &= D^*(p+h-1) + tr(p+h-1)r(p+h) = \\ &= A(p+h) + \text{máx}\left\{x - \sum_{s=p,\ldots,p+h-1} E(s), 0\right\}. \end{aligned}$$

Corolario 1. Sea $H(p,q)$, siendo $p = 1,\ldots,nr$, y $q = p,\ldots,nr$, el máximo retardo en la llegada a $r(p)$, es decir, incremento en $A(p)$ que no produce violación de las ventanas de tiempo en $r(q)$ (se mantiene $A^*(q) \leqslant 1$); se tiene que:

$$H(p,q) = \sum_{s=p,\ldots,q-1} E(s) + M(q)$$

Sea:

$$cadena = r(p) - r(p+1) - \cdots - r(q),$$

se define

- *maxretraso(p,q)* = Máximo retardo en la llegada a $r(p)$, que no produce violaciones de las ventanas de tiempo en los puntos de cadena;
- *tespera(p,q)* = Tiempo total de espera acumulado en cadena;

obviamente

$$\begin{aligned} \textit{tespera(p,q)} &= \sum_{s=p,\ldots,p+h-1} E(s) \quad \text{y} \\ \textit{tespera(p,q)} &= \textit{tespera}(p,q-1)+E(q); \end{aligned}$$

por comodidad si $q < p$ (si cadena no contiene elementos), se define *tespera(p,q) = 0.*

Proposición 2. *Se tiene que los valores de maxretraso se pueden obtener de la forma siguiente:*

$$\textit{maxretraso(p,q)} = \min_{s=p,\ldots,q} H(p,s) = \min_{s=p,\ldots,q} \{M(s)+\textit{tespera}(p,s-1)\}$$

Demostración. Obvio a partir de Proposición 1 y Corolario 1 y la definición de maxretraso.

■

Un método recursivo para obtener el valor de maxretraso viene dado por la siguiente fórmula:

$$\textit{maxretraso(p,q)} = \min\{\textit{maxretraso}(p,q-1), M(q)+\textit{tespera}(p,q-1)\}$$

con

$$\textit{maxretraso}(p,p) = M(p) \quad \text{como valor inicial.}$$

Asimismo se define:

– *maxadelanto(p,q)* = Máximo adelanto en la llegada a $r(p)$, es decir disminución de $A(p)$, que no produce tiempo de espera en dicha cadena (es decir, se mantiene $A(s) \geqslant e$ para $s = p,\ldots,q$); obviamente si en algún punto de cadena ya existe espera maxadelanto(cadena) es 0.

Proposición 3. *Los valores de maxadelanto pueden obtenerse de la siguiente forma:*

$$\textit{maxadelanto(p,q)} = \min_{s=p\ldots q} \{\max[0, A(s)-e]\}$$

A partir de aquí se puede obtener un método recursivo para su cálculo:

$$\textit{maxadelanto(p,q)} = \min\{\textit{maxadelanto}(p,q-1), \max[0, A(q)-e]\}$$

con

$$\textit{maxadelanto(p,p)} = \text{máx}[0, A(p) - e] \quad \text{como valor inicial.}$$

Estas variables son importantes a la hora de determinar cómo afecta la alteración en el tiempo de llegada al primer punto de una cadena, al tiempo de salida del último punto de esa cadena y a la posible violación o no de las ventanas de tiempo en los puntos de la cadena. Concretamente, sea para una determinada cadena $r(p) - r(p+1) - \cdots - r(q)$, adelanto_inicial la cantidad de tiempo en que se adelanta la llegada a $r(p)$, y adelanto_final la cantidad de tiempo en que se adelanta la salida de $r(q)$, se tiene que:

$$\textit{adelanto\_final} = \text{mín}\{\textit{adelanto\_inicial}, \textit{maxadelanto}\}$$

análogamente, sean *retraso_inicial* y *retraso_final* respectivamente los atrasos en los tiempos de llegada a $r(p)$ y salida de $r(q)$, se tiene que:

$$\textit{retraso\_final} = \text{máx}\left\{ \textit{retraso\_inicial} - \sum_{s=p,\ldots,q} E(s), 0 \right\}.$$

Las variables globales definidas en este apartado pueden ser calculadas con orden $\theta(n^2)$ de operaciones, facilitando posteriormente el chequeo de la factibilidad y valoración de los intercambios.

4. DISEÑO DEL ALGORITMO INICIAL

Como se ha mencionado en la introducción, el algoritmo propuesto se divide en varios subalgoritmos que, en la mayoría de los casos, se basan en movimientos vecinales. Sea $s^* \in S$ la solución actual en cada momento, y $k \in IN$ un número entero prefijado, inicialmente se van a definir los siguientes procedimientos:

▶ Procedimiento Búsqueda_Local_Or(k, sf)

Repetir

Determinar $s' \in N_1^k(sf)$ *verificando* $f(s') = \text{mín}\{f(s)/s \in N_1^k(sf)\}$

Si $f(s') < f(sf)$ *entonces hacer* $sf = s'$

hasta que $f(s) \geqslant f(sf), \forall s \in N_1^k(sf)$.

▶ Procedimiento Búsqueda_Local_Ge(sf)

Repetir

Determinar $s' \in N_2(sf)$ *verificando* $f(s') = \text{mín}\{f(s)/s \in N_2(sf)\}$

Ejecutar Búsqueda_Local_Or$(3, r')$ y Búsqueda_Local_Or$(3, r'')$ *donde* r' *y* r'' *son las dos rutas de sf modificadas para dar lugar a* s'

Si $f(s') < f(sf)$ *hacer* $sf = s'$

hasta que $f(s) \geqslant f(sf), \forall s \in N_2(sf)$.

Estos procedimientos de Búsqueda Local son análogos, aunque obsérvese como en el segundo caso, cuando se realiza un intercambio (tipo Gendreau-Clarke) a continuación se ejecuta el primero para mejorar cada una de las dos rutas implicadas en este intercambio.

Una vez descritos estos procedimientos el Algoritmo Inicial queda de la siguiente forma:

▶ Procedimiento Algoritmo_Inicial(Output $s^* \in S$)

Paso 1: Obtención de una Solución Inicial s^* por un método constructivo

Paso 2: Ejecutar Búsqueda_Local_Ge(s^*)

Paso 3: A cada ruta de s^* aplicar Búsqueda_Local_Or con $k = \infty$

Paso 4: Ejecutar Búsqueda_Local_Or(∞, s^*)

Para la obtención de una solución inicial se usa una adaptación del algoritmo de Fisher & Jaikumar (1981) para este modelo. Aunque en este caso también se obtienen soluciones aceptables con una adaptación para este modelo del algoritmo de *inserción más cercana*. (Una colección exhaustiva de estos métodos constructivos para el TSP se puede encontrar en el trabajo de Golden y otros, (1980).)

Tras el paso 2, cada una de las rutas de la solución s^* obtenida es óptimo local con respecto a N_1^3, es decir, no son mejorables con recolocaciones de tipo Or de cadenas de hasta 3 elementos. Se ejecuta el paso 3 por si pueden ser mejoradas con recolocaciones de cadenas de mayor tamaño (obsérvese que en este paso sólo se consideran recolocaciones de elementos dentro de una misma ruta).

Finalmente, en el paso 4 se vuelve a ejecutar el mismo procedimiento *Búsqueda_Local_Or* a la solución obtenida, pero considerada como una sola cadena, con el objeto de analizar las posibles recolocaciones que afecten a rutas diferentes.

5. MEJORA CON UN PROCEDIMIENTO DE BÚSQUEDA TABÚ

El algoritmo descrito en el apartado anterior da resultados satisfactorios mejorando sensiblemente las soluciones usadas actualmente en los Centros Escolares analizados, con un tiempo de cálculo razonable (ver apartado siguiente).

Sin embargo, las soluciones obtenidas aún podrían ser ligeramente mejoradas, en algunos casos, añadiendo un paso posterior en el que se aplique alguna técnica Metaheurísti-

ca desarrollada en los últimos años. Más concretamente, se propone un procedimiento basado en un proceso de búsqueda tabú que se describe a continuación.

La *búsqueda tabú* es un procedimiento o estrategia dado a conocer en los trabajos de Glover (1989) y (1990), y que está teniendo grandes éxitos y mucha aceptación en los últimos años. Según su creador, es un procedimiento que «*explora el espacio de soluciones más allá del óptimo local*» (Glover y Laguna (1993)). Se permiten cambios *hacia arriba* o que empeoran la solución, una vez que se llega a un óptimo local. Simultáneamente, los últimos movimientos se califican como *tabúes* durante las siguientes iteraciones para evitar que se vuelva a soluciones anteriores y el algoritmo cicle. El termino tabú hace referencia a «*un tipo de inhibición a algo debido a connotaciones culturales o históricas y que puede ser superada en determinadas condiciones...*». (Glover (1996)). Recientes y amplios tutoriales sobre búsqueda tabú, que incluyen todo tipo de aplicaciones, pueden encontrarse en Glover y Laguna (1997) y (1999).

5.1. El algoritmo básico

El algoritmo que se propone básicamente actúa de la forma siguiente:

► Procedimiento Búsqueda_Tabú_Básico

Leer como solución inicial s^* (obtenida por el Algoritmo Inicial descrito en la sección 4)

Hacer $s_0 = s^*; T = \emptyset, niter = 0, kiter = 0$

Repetir

niter:=niter+1;

Seleccionar $s \in N_2(s_0)/s \notin T$ *o s verifica criterio de 'aspiración' con* $f(s)$ *mínimo*

Hacer $s_0 = s$

Ejecutar Búsqueda_Local_Or$(3, r')$ *y* Búsqueda_Local_Or$(3, r')$ *donde* r' *y* r'' *son las dos rutas de* s_0 *modificadas*

Si $f(s_0) < f(s^*)$ *entonces: hacer* $s^* = s_0$ *y kiter = niter*

Actualizar T

hasta $niter - kiter \geqslant maxiter$

Se denota por s^* a la solución óptima en cada momento. T es el conjunto de movimientos tabúes y se obtiene determinando qué conjunto de soluciones tienen ciertos *atributos tabús activos*. Por tanto, se han de definir estos atributos tabús y durante cuántas iteraciones van a permanecer activos (y, por tanto, las soluciones que les contienen).

El objeto de aplicar el *criterio de aspiración* es determinar en qué condiciones un movimiento tabú puede ser admisible. Habitualmente se considera que una solución *s* cumple el *criterio de aspiración* si $f(s) < f(s^*)$. Este movimiento facilita una nueva dirección de búsqueda y garantiza que no se produzcan ciclos.

Por otra parte, cualquier movimiento vecinal de este tipo supone la incorporación de un conjunto de arcos y la eliminación de otros (según se ilustró en el apartado 3). Para que en las iteraciones siguientes no se vuelva a la solución anterior, se van a impedir los movimientos que supongan la incorporación de algunos de estos arcos en la solución actual. En otras palabras, los *atributos tabús* van a ser los arcos que componen cada solución, y un movimiento es tabú si supone la incorporación de algún arco eliminado en iteraciones recientes (*atributo tabú activo*).

Para identificar qué atributos tabús (arcos) van a estar activos se define arco_tabú una matriz $n \times n$ de la siguiente forma:

$$arco\_tabú(r,l) = \text{nº de la última iteración en la que fue eliminado el arco}(r,l).$$

Un determinado arco (r,l) será un atributo tabú activo si:

$$niter - arco\_tabú(r,s) < maxiter\_tabú$$

siendo *maxiter_tabú* el número de iteraciones que permanece activo como atributo tabú desde que es eliminado de la ruta actual.

Inicialmente se define *arco_tabú(r,l) = 0*, para cada arco (r,l) en la solución inicial, *arco_tabú(r,l)* = $-$*maxiter_tabú* (o un valor más negativo) para el resto. De esta forma se impide que en las primeras iteraciones sean declarados tabús activos arcos que no formen parte de la solución actual. Así, inicialmente se asegura que $T = \emptyset$.

– Se va a dar al parámetro maxiter_tabú el valor de *maxiter_tabú* = $2^* n^{1/2}$.
– Para el criterio de parada se toma maxiter = $10 \cdot n$.

En esta sección se ha descrito un algoritmo de búsqueda tabú básico, pero que puede ser complementado y ampliado con procedimientos basados en lo que se denomina habitualmente memoria a *largo* y *medio plazo* como *diversificación* e *intensificación*; también se puede enriquecer con la posibilidad de visitar de forma controlada soluciones infactibles, por medio de funciones de penalización (*oscilación estratégica*).

5.2. Fase de intensificación

Como señala el nombre, en esta fase se intensifica la exploración de las regiones y los vecindarios donde se hallan las mejores soluciones encontradas en fase inicial (algoritmo básico) con la esperanza de encontrar aún ligeras mejoras. Esta fase puede ser

diseñada de diferentes formas. En este caso, está inspirada en los trabajos de Rossing (1997), Rossing y otros (1997) y (1998) sobre *Concentración Heurística*, una estrategia para encontrar soluciones en dos fases.

En la primera se ejecuta varias veces un procedimiento de búsqueda local, registrándose los mejores óptimos locales obtenidos; en la segunda se construye un *conjunto de concentración*, CS, con los elementos de las mejores soluciones obtenidas en la primera, y se ejecuta un algoritmo exacto o heurístico pero considerando sólo los elementos de CS.

En este trabajo se va a tomar la idea de construir un *conjunto de concentración* con los elementos de las mejores soluciones obtenidas en el algoritmo básico. Más concretamente, para este modelo se va a considerar como elementos de las soluciones a los arcos que la componen; por ejemplo la solución compuesta por las dos rutas:

$$\textit{ruta 1:} 1-3-5-1; \quad \text{y} \quad \textit{ruta 2:} 1-4-2-1$$

que puede expresarse como la secuencia $1-3-5-1-4-2-1$ (según se vio en el apartado 2), estará formada por los arcos (1, 3), (3, 5), (5, 1), (1, 4), (4, 2) y (2, 1).

Para formar el conjunto de concentración utilizamos el siguiente procedimiento:

▶ Procedimiento Construcción_Conjunto_de_Concentracion
Ordenar las soluciones obtenidas en una lista según el valor de f (comenzando con la mejor)
Hacer i = 0 y CS = ∅
Repetir
hacer $i = i+1$
Añadir los elementos de la i-ésima solución de la lista a CS
hasta Cardinal(CS) $\geqslant$ *num_arcos*

Obsérvese que a diferencia de la propuesta por Rossing no se fija un número predeterminado de soluciones cuyos elementos se introducen, sino que se fija un número mínimo de elementos a introducir *num_arcos*.

A continuación, se va a diseñar un procedimiento que *concentre* la búsqueda de elementos en CS. Inicialmente, se define la siguiente matriz de distancias auxiliar $d1$ de la siguiente forma:

$$d1(i,j) = d(i,j) \qquad \text{si } (i,j) \in \text{CS}$$
$$d1(i,j) = d(i,j) + 10^{*} max\_d \qquad \text{si } (i,j) \notin \text{CS}$$

donde *max_d* = máx$\{d(i,j)/i,j = 1,\ldots,n\}$; además se define $f1(s)$ como el valor de

la función objetivo considerando el coste de los arcos el dado por $d1$. Se propone el siguiente procedimiento de búsqueda que tiene en cuenta ambas matrices d y $d1$:

▶ Procedimiento Búsqueda_Local_Guiada (sf)
Hacer $s_0 = sf$
Repetir
Hacer coste_anterior = f(sf)
Repetir
Buscar $s' \in N_2(s_0)/f1(s') = \min\{f1(s)/s \in N_2(s_0)\}$
En s' *Ejecutar* Búsqueda_Local_Or$(3,r')$ *y* Búsqueda_Local_Or$(3,r'')$
(considerando f1 en vez de f) donde r' *y* r'' *son las dos rutas de* s_0 *modificadas para dar lugar a* s'
Si $f1(s') < f1(s_0)$ *entonces* $s_0 = s'$
Buscar $s'' \in N_2(s_0)/f(s'') = \min\{f(s)/s \in N_2(s_0)\}$
En s'' *Ejecutar* Búsqueda_Local_Or$(3,r')$ *y* Búsqueda_Local_Or$(3,r'')$ *donde* r' *y* r'' *son las dos rutas de* s_0 *modificadas para dar lugar a* s''
Si $f(s'') < f(sf)$ *entonces* $sf = s''$
hasta $f1(s') \geqslant f1(s_0)$
Hacer $s_0 = sf$
hasta f(sf) = coste_anterior

Se trata de un procedimiento de búsqueda local anidado: en cada paso la solución actual s_0 se sustituye por otra mejor según $f1$, es decir según $d1$ y buscando por tanto soluciones que contengan elementos de CS; cuando no hay mejora en $f1$, se sustituye s_0 por sf, la mejor solución según f observada en los vecindarios explorados y se reinicia la búsqueda local. El proceso acaba cuando no hay mejora en $f(sf)$.

En definitiva, es un procedimiento de búsqueda 'guiado' por $f1$, i.e. por $d1$, hacia soluciones que contengan el mayor número de elementos de CS posibles. En cierta manera, podría ser considerado como una forma de reencadenamiento de trayectorias. Obsérvese que $d1$ 'penaliza' a los arcos no pertenecientes a CS, pero no 'impide' su elección en cada paso, ya que esto podría hacer excesivamente reducido el número de soluciones a considerar y 'encajonar' el proceso. Obviamente, si se usara un algoritmo exacto la estrategia debería ser impedir en vez de penalizar. Este procedimiento se inserta en la fase de intensificación que queda de la siguiente forma:

▶ Procedimiento intensificación
Ejecutar Construcción_Conjunto_de_Concentracion;
Desde i:=1 hasta num_soluciones hacer

Tomar s_i *la i-esima solución de la lista ordenada obtenida en*
Búsqueda_Tabú_Básico
Ejecutar Ejecutar Búsqueda_Local_Ge(s_i)
A cada ruta de s_i *aplicar* Búsqueda_Local_Or *con* $k = \infty$,
Ejecutar Búsqueda_Local_Guiada(s_i);
Si $f(s_i) < f(s^*)$ *hacer* $s^* = s_i$

Se toma $f1$ en vez de f al generar las soluciones iniciales y se ejecuta el procedimiento de *Búsqueda_Local_Guiada* dirigido por $f1$, en vez de la búsqueda local habitual. En definitiva, se 'concentra' o intensifica la búsqueda de soluciones en las regiones con elementos de CS.

Para este trabajo se ha tomado *num_soluciones* = 50. En cuanto al valor de este parámetro *num_arcos*, en el trabajo de Pacheco y Delgado (1999), se describen los resultados de diferentes experiencias que aconsejan tomar *num_arcos* como el 10 % del total de arcos (i.e. *num_arcos* = $0{,}1^*n^*(n-1)$).

Al añadir la fase de intensificación el algoritmo búsqueda tabú queda de la siguiente forma:

▶ Procedimiento Búsqueda_Tabú_Principal;
Ejecutar Búsqueda_Tabú_Básico
Ejecutar Intensificación

Se han de tener en cuenta las siguientes consideraciones: durante la ejecución de la fase básica es necesario crear y actualizar una lista con las mejores soluciones para ser usada en la fase de intensificación; para formar parte de esta lista se considera en cada iteración la mejor de todas las soluciones del vecindario analizado independientemente de contener o no elementos tabú o de cumplir el criterio de aspiración. Por otra parte se podría añadir al procedimiento Búsqueda_Tabú_Principal una fase de diversificación y volver a ejecutarlo una o varias veces. Sin embargo, en este trabajo no va a ser así para evitar tiempos de computación excesivos.

6. RESULTADOS COMPUTACIONALES

En esta sección se van a analizar los diferentes problemas o instancias del transporte escolar de secundaria en la provincia de Burgos. Cada problema viene definido por un centro escolar, por las localizaciones donde se recogen los alumnos que van a ese centro y por el número de alumnos a recoger en cada una de esas localizaciones. En

todos los casos el tiempo máximo de un alumno en ruta, *tmax*, es de 60′. Cada una de estas instancias se ha resuelto con los algoritmos anteriormente descritos.

Para el cálculo de la matriz de distancias y del camino entre cada par de puntos del problema se ha ponderado la distancia de cada tramo según el tipo de carretera, de forma que favorezca la elección de carreteras nacionales antes que autonómicas, éstas antes que carreteras sin revestimiento, etc., de forma que en muchas ocasiones los caminos obtenidos no son los más cortos, pero si los más cómodos y rápidos.

A continuación, para los problemas correspondientes al transporte de centros de secundaria se muestra la población donde está cada centro, el número de localidades donde se recogen los niños que van a esos centros y el número de niños, los resultados con los costes y los tiempos de computación de los algoritmos anteriormente descritos, así como de la solución que se usa en la realidad (siempre teniendo en cuenta la función de costes descrita en la sección 1).

Pr.	Centro (Población)	n. loc. alumn.	S. Actu.	Algoritmo inicial				B. Tabú	
				Paso 1	Paso 2	Paso 3	Paso 4	Básico	Intensif.
1º	ARANDA DE DUERO	57 429	61.177,5 12	67.123,6 10 8,72	57.973,4 13 0,61	57.973,4 13 0,03	57.973,4 13 0,40	57.973,4 13 26,87	56.196,1 14 72,78
2º	BELORADO	24 175	21.045,0 5	22.960,5 4 0,29	16.962,5 8 0,22	16.962,5 8 0,02	16.962,5 8 0,12	16.962,5 8 7,08	16.962,5 8 20,74
3º	BRIVIESCA	24 101	25.051,3 6	24.055,0 5 0,75	23.333,8 6 0,07	23.333,8 6 0,01	23.333,8 6 0,06	23.197,4 7 7,56	23.197,4 7 7,02
4º	L. Mendoza BURGOS	19 127	18.608,8 3	18.104,4 4 0,26	17.542,4 4 0,03	17.542,4 4 0,00	17.542,4 4 0,03	16.880,3 5 5,30	16.863,6 5 6,26
5º	Diego Marín BURGOS	23 59	15.902,5 4	14.773,3 3 0,35	14.614,9 3 0,02	14.614,9 3 0,00	14.614,9 3 0,03	14.614,9 3 5,25	14.614,9 3 4,43
6º	Diego Siloé BURGOS	32 141	30.720,0 4	34.831,6 5 1,81	29.742,0 6 0,18	29.742,0 6 0,01	29.742,0 6 0,08	27.410,8 6 11,65	27.410,8 6 12,29
7º	S.de Colonia BURGOS	36 158	34.835,0 6	28.147,5 5 1,98	22.820,9 7 0,31	22.820,9 7 0,01	22.820,9 7 0,17	22.539,1 7 27,71	22.539,1 7 19,12
8º	LERMA	53 302	51.112,5 9	52.730,5 9 6,80	45.152,3 11 0,46	45.152,3 11 0,02	43.152,9 10 1,40	41.006,9 10 53,77	40.542,0 11 42,49
9º	MEDINA DE POMAR	39 182	31.075,0 5	35.005,8 6 2,87	30.790,4 7 0,19	30.790,4 7 0,01	30.790,4 7 0,15	30.189,1 7 16,55	30.189,1 7 15,81
10º	MELGAR	22 71	19.402,5 6	19.758,9 4 0,44	18.887,0 5 0,02	18.887,0 5 0,01	18.887,0 5 0,05	18.847,9 6 5,27	18.847,9 6 5,16
11º	MIRANDA	13 41	16.622,5 4	19.038,1 4 0,17	18.847,9 5 0,03	18.847,9 5 0,01	18.847,9 5 0,01	18.847,9 5 2,27	18.847,9 5 3,60
12º	QUINTANAR	4 105	8.025,0 2	6.925,0 2 0,01	3.312,5 4 0,02	3.312,5 4 0,00	3.312,5 4 0,00	3.312,5 4 0,66	3.312,5 4 1,79
13º	ROA	28 207	22.975,0 6	28.072,4 5 0,51	20.237,5 7 0,26	20.237,5 7 0,01	20.200,0 7 0,17	20.200,7 7 8,83	19.437,5 7 21,85
14º	SALAS	23 81	21.488,8 5	21.371,1 5 0,55	18.586,8 4 0,04	18.586,8 4 0,00	18.586,4 4 0,03	18.512,5 4 5,26	18.512,5 4 3,27
15º	VILLADIEGO	31 128	29.088,8 7	29.604,8 6 1,75	25.212,5 8 0,10	25.212,5 8 0,01	25.212,5 8 0,12	25.000,0 7 12,22	24.981,2 8 8,71
16º	VILLARCAYO	9 23	8.252,5 2	13.205,5 2 0,05	10.847,0 2 0,01	10.847,0 2 0,01	10.847,0 2 0,00	10.847,0 2 1,61	10.847,0 2 1,75
T.	**TOTAL PROVINCIA**	**437 2.330**	**415.382’7 86**	**435.708’0 79 27,31**	**374.863’8 100 2,57**	**374.863’8 100 0,16**	**372.826’5 99 2,82**	**366.342’9 101 197,86**	**363.302’0 104 247,07**

En cada celda se muestra el coste, el número de vehículos y el tiempo de computación en segundos

A continuación se muestran dos gráficos correspondientes a los costes y tiempos de computación de cada algoritmo:

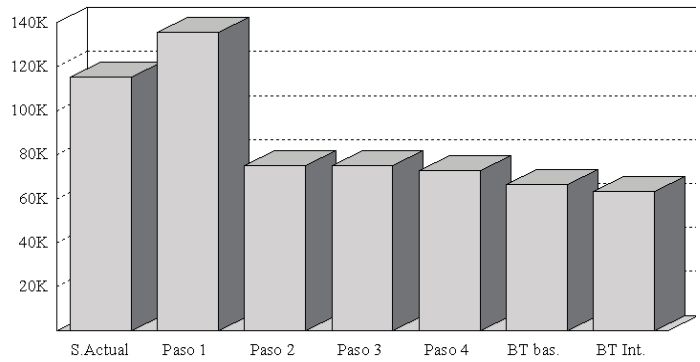


Figura 4. Costes de las soluciones obtenidas. El paso 2 supone la disminución más significativa. La búsqueda tabú también aporta reducciones importantes.

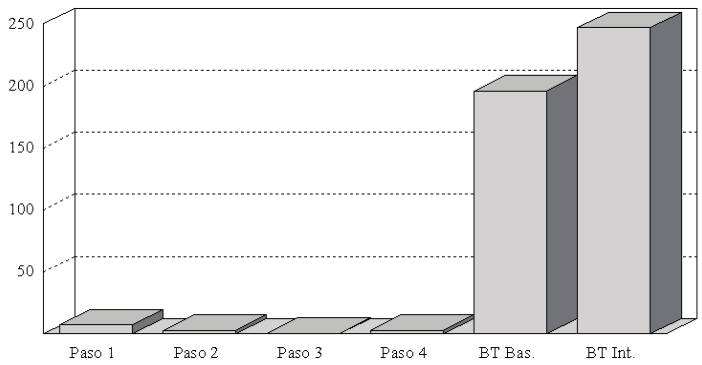


Figura 5. Tiempos de computación (en segundos) empleados. Insignificantes tiempos de las partes del algoritmo inicial, frente al empleado por la búsqueda tabú

Los algoritmos diseñados en este trabajo se han programado usando el compilador Borland Pascal 7.0 y Borland Delphi 3.0. El equipo informático donde se han programado y se han realizado las pruebas es un ordenador personal *Pentium MMX-200* Mhz.

Como se ve en la tabla de resultados, el algoritmo inicial aporta soluciones rápidamente (apenas poco más de 11 segundos para todos los problemas de secundaria), que mejoran las soluciones actuales en más de un 10 % en el coste. Si se dispone de más tiempo de computación, la búsqueda tabú aporta soluciones que mejoran algo más de un 2 % las del algoritmo inicial, y un 13 % las soluciones actuales. Resultados similares se han registrado para primaria.

7. CONCLUSIONES Y REFLEXIONES

Es claro que los algoritmos propuestos (A.Inicial y A.Inicial+B.Tabú), globalmente dan soluciones muy interesantes en cuanto a ahorro de costes. Obsérvese, sin embargo, que existen instancias, como se refleja en la tabla anterior, para las que esta mejora es escasa. Incluso en algún caso, los algoritmos descritos aparentemente aportan soluciones peores que las utilizadas actualmente (Miranda y Villarcayo). Esto se debe a las siguientes causas:

– Para el cálculo de los caminos, distancias y tiempos se ha usado la red de carreteras obtenida de la cartografía digitalizada suministrada por el CNIG (Centro Nacional de Información Geográfica). Esta cartografía es muy completa y de mucha calidad. Sin embargo, en nuestro caso se ha detectado alguna imprecisión (falta de algún tramo o cruces no detectados) que puede dar lugar a que los caminos hallados, en algún caso concreto, sean más largos y costosos que los que se podrían usar realmente.
– Muchas de las rutas de las soluciones actuales, a la hora de la verdad tienen una duración mayor que el máximo programado de 60′. (En algún caso hasta de 90′, según responsables de la propia Dirección Provincial de Educación, con el consiguiente esfuerzo de intentar ‘desdoblar’ estas rutas.) Esto se debe a que en la planificación de las rutas se supuso unas velocidades medias mayores de las que se pueden conseguir en la realidad.
– Sin embargo, en nuestro caso se han supuesto unas velocidades bastante moderadas para cada tipo de carretera; concretamente 80 km/h para autopista y autovía, 70 km/h para nacional, 60 para autonómica de 1er orden, 55 para autonómicas de 2^{o} orden, 50 para autonómicas de 3er orden, 40 para carreteras sin revestimiento, 30 para carreteras de enlace y 20 para travesías. Obviamente, estas velocidades se pueden conseguir fácilmente en la realidad e incluso mayores. Lo importante es que las soluciones planificadas teóricamente puedan ser llevadas a cabo en la realidad con cierto margen. (Al contrario de lo que pasa en algunas soluciones usadas actualmente.)

Lo señalado en el párrafo anterior hace reflexionar en el siguiente punto: la reducción de costes es algo bueno, pero esta reducción se ha de conseguir manteniendo, e incluso mejorando, la comodidad de los trayectos (menores distancias y tiempos; mejores carreteras, etc.). En otras palabras, se ha de buscar la *racionalidad* en el sentido más amplio de la palabra, que es algo socialmente muy valioso tratándose del problema del transporte escolar.

RECONOCIMIENTO

Nuestro más sincero agradecimiento a los responsables de transporte de la Delegación Provincial del Ministerio de Educación y Ciencia de Burgos, por los datos suministrados y por las facilidades dadas en general.

REFERENCIAS

Backer (de), B. Furnon, V. Kilby, P., Prosser, P. and Shaw, P. (1997). «Solving Vehicle Routing Problems using Constraint Programming and Metaheuristics». *Journal of Heuristics*, 1 - 16.

Bodin, L.D. and Golden, B.L. (1981). «Classification in Vehicle Routing and Scheduling». *Networks*, 11, 2, 97-108.

Bullheimer, B., Harti, R.F. and Strauss, C. (1997). *Applying the Ant System for the Vehicle Routing Problem*. 2nd Metaheuristics International Conference (MIC-97), Sophie-Antipolis, France, July 1997.

Campos, V. y Mota, E. (1995). «Metaheurísticos para el CVRP». *XXII Congreso Nacional de Estadística e Investigación Operativa*. Sevilla, Noviembre 1995.

Clarke, G. and Wright, J.W. (1964). «Scheduling of Vehicles from a Central Depot to a Number of Delevery Points». *Oper. Res.*, 12, (1964), 568-581.

Desrochers, M., Lenstra, J.K., Savelsbergh, M.W.P. and Soumis, F. (1988). «Vehicle Routing with Time Windows: Optimization and Approximation». In *Vehicle Routing: Methods and Studies*, (Studies in Management Sciences and Systems, vol. 16), eds: Golden, B.L. and Assad, A.A., Nort-Holland, 65-84.

Fisher, M.L. y Jaikumar, R. (1981). «A Generalized Assignment Heuristic for Vehicle Routing». *Networks*, 11, 2, 109-124.

Gendreau, M., Hertz, A. and Laporte, G. (1991). «A Tabu Search Heuristic for Vehicle Routing Problem». *Report CRT-777*. Centre de Recherche sur les Transports. Univ. Montréal.

Gendreau, M., Hertz, A. and Laporte, G. (1994). «A Tabu Search Heuristic for Vehicle Routing Problem». *Management Sci.*, 40 (10), 1276-1290.

Glover, F. (1989). «Tabú Search: Part I». *ORSA Journal on Computing*, 1, 190-206.

Glover, F. (1990). «Tabú Search: Part II». *ORSA Journal on Computing*, 2, 4-32.

Glover, F. (1996). *Búsqueda Tabú en Optimización Heurística y Redes Neuronales*. Adenso Díaz (coordinador). Paraninfo. Madrid. pp. 105-143.

Glover, F. y Laguna, M. (1993). *Tabu Search in Modern Heuristic Techniques for Combinatorial Problems*. C. Reeves, ed., Blackwell Scientific Publishing, pp. 70-141.

Glover, F. y Laguna, M. (1997). *Tabu Search*. Kluver Academic Publishers, Boston.

Glover, F. y Laguna, M. (1999). *Tabu Search, aparecerá en Handbook of Applied Optimization*, P.M. Paradalos and M.G.S. Resende (eds). Oxford Academic Press.

Haouari, M., Dejax, P. et Desrochers, M. (1990). «Les Problems de Tournées avec Contraintes des Fenênetres de Temps: L'Etat de l'Art». *Recherche Operationnelle/Operations Research*, 24, 3, 217-244.

Kilby, P., Prosser, P. and Shaw, P. (1997). *Guided Local Search for the Vehicle Routing Problem*. 2nd Metaheuristics International Conference (MIC-97), Sophie-Antipolis, France, July 1997.

Kontoravdis, G. and Bard, J.F. (1995). «A Grasp for the Vehicle Routimg Problem with Time Windows». *ORSA Journal on Computing*, 7, 10-23.

Laporte, G. (1992). «The Vehicle Routing Problem: An overview of exact and approximate algorithms». *European Journal of Operations Research*, 59, 345-358.

Laporte, G., Desrochers, M. and Nobert, Y. (1984). «Two Exact Algorithms for the Distance-Constrained Vehicle Routing Problem». *Networks*, 14, 161-172.

Laporte, G., Nobert, Y. and Desrochers, M. (1985). «Optimal routing under capacity and distance restrictions». *Operations Research*, 33, 1075-1073.

Laporte, G. and Osman, I.H. (1995). «Routing Problems: A Bibliography». *Ann. Oper. Res.*, 61, 227-262.

Lenstra, J.K. and Rinnoy Kan, A.H.G. (1981). «Complexity of Vehicle Routing and Scheduling Problems». *Networks*, 11, 2, 221-228.

Lin, S. (1965). «Computer Solutions to the Traveling Salesman Problem». *Bell Syst. Tech. Jou.*, 44, 2245-2269.

Lin, S. y Kernighan, B.W. (1973). «An Effective Heuristic Algorithm for the Traveling Salesman Problem». *Operations Researsch*, 20, 498-516.

Nurmi, K. (1991). «Traveling Salesman Problem Tools for Microcomputers». *Computers & Ops. Res.*, 18, 8, 741-749.

Or, I. (1976). *Traveling Salesman Type Combinatorial Problems y their Relations to the Logistics of Blood Banking*. Ph. Thesis, Dpt. of Industrial Enginnering y Management Sciences, Northwestern Univ.

Osman, I.H. (1993). «Metastrategy Simulated Annealing and Tabu Search Algorithms for the Vehicle Routing Problem». *Annals of Operations Research*, 41, 421-451.

Pacheco, J. y Delgado, C. (1996). *Adaptación del Algoritmo de Or al VRPTW con Carga y Descarga simultánea*. X Reunión Asepelt-España, Albacete, Junio 1996.

Pacheco, J. y Delgado, C. (1997). «Problemas de Rutas con Ventanas de tiempo y carga y Descarga simultánea: Diseño de Filtros para algoritmos de intercambio (caso de un sólo vehículo)». *Estudios de Economía Aplicada*, 7 , 79-100.

Pacheco, J. y Delgado, C. (1999). «Diseño de Metaheurísticos híbridos para Problemas de Rutas con Flota Heterogénea: Concentración Heurística». Aceptado para su publicación en *Estudios de Economía Aplicada*.

Potvin, J.Y. and Bengio, S. (1994). «A Genetic Approach to the Vehicle Routing Problem with Time Windows». *Technical Report CRT-953*, Centre de Recherche sur les Transports. Univ. Montréal.

Potvin, J.Y., Kervahut, T., García, B.L. and Rousseau, J.M. (1993). «A Tabu Search Heuristic for Vehicle Routing Problem with Time Windows». *Report CRT-777. Management Sci.*, 40 (10), 1276-1290.

Rego, C. (1998). «A Subpath Ejection Method for the Vehicle Routing Problem». *Management Science*, 44, 10, 1447-1459.

Rochat, Y. and Taillard, E.D. (1995). «Probabilistic Diversification and Intensification in Local Search for Vehicle Routing». *Journal of Heuristics*, 1 (1), 147-167.

Rosing, K.E. (1997). *Heuristic Concentration: An Introduction with Examples. The Tenth Meeting of the European Chapter on Combinatorial Optimization*. Tenerife. Spain. May, 1997.

Rosing, K.E. and Revelle, C.S. (1997). «Heuristic Concentration: Two Stage solution Construction». *European Journal of Operational Research*, 97, 75-86.

Rosing, K.E., Revelle, C.S., Rolland, E., Schilling, D.A. and Current, J.R. (1998). «Heuristic Concentration and Tabu Search: A head to head comparison». *European Journal of Operational Research*, 104, 93-99.

Solomon, M.M. (1987). «Algorithms for the Vehicle Routing and Scheduling Problem with Time Windows Constraints». *Operations Research* 35, 254-265.

Taillard, E., Badeau, P., Gendreau, M., Guertain, F. and Potvin, J.Y. (1995). *A new Neighbourhood structure for the Vehicle Routing Problem with Time Windows*. Technical Repport CRT-95-66, Centre de Recherche sur les Transports. Univ. Montréal.

Taillard, E., Badeau, P., Gendreau, M., Guertain, F. and Potvin, J.Y. (1997). «A Tabu Search heuristic for the Vehicle Routing Problem with Time Windows». *Transportation Science*, 31, 170-186.

Thangiah, S.R., Osman, I.H. and Sun, T. (1994). «Hybrid Genetic Algorithm, Simulated Annealing, and Tabu Search methods for the Vehicle Routing Problem with Time Windows». *Working paper UKC/OR94/4*, Institute of Mathematics and Statistics, University of Kent, Canterbury.

Thangiah, S.R., Vinayagamoorty, R. and Sun, T. (1993). «Vehicle Routing Problem with Time Deadlines using Genetic and Local Algorithms». In *5th International Conference on Genetic Algorithms*.

ENGLISH SUMMARY

ALGORITHM DESIGN FOR SCHOOL TRANSPORTATION. APPLICATION TO BURGOS PROVINCE

J.A. PACHECO
A. ARAGÓN
C. DELGADO
Universidad de Burgos*

The issue of school transportation in Burgos is particularly significant since it is a large province with many inhabited areas widely dispersed and only scantly populated. In this study, the authors = attempts to provide a solution to this problem in the most reasonable way possible are presented. It should be noted that, in this context, the term 'reasonable' does not refer solely to the minimization of total transport costs, but also to the consideration of certain aspects such as the amount of time students are in the vehicle, the selection of good highways, etc., particularly in light of certain recent incidents. Algorithms based principally on Local Search are proposed and described and then a Tabu Search algorithm is also suggested as an option depending on the computation time. Likewise, the results obtained from the data from the current year are also shown.

Keywords: Routes, school transportation, local search, tabu search, intensification, heuristic concentration

AMS Classification (MSC 2000): 90B20

---

* Departamento de Economía Aplicada. Universidad de Burgos. Parralillos, s/n. 09001 Burgos.

– Received June 1999.
– Accepted January 2000.

Consider the transportation problem of a group of students that need to be picked up from a series of geographically distributed places and taken to an educational center. The number 1 can indicate the educational center, 2, $\ldots, n$ points corresponding to the places where the children are picked up, and $q(i)$ the number of students that are picked up at each point $i, i = 2, \ldots, n$. In order to meet these demands, the legislation authorizes different types of vehicles, each one with a different number of seats. Each student should not be in route more than a determined maximum amount of time, which is indicated by *tmax*, and the route should be completed before the start of classes in the moment *tinicio*. The distances d_{ij} and the times t_{ij} between each pair of points i, j are known. The number of types of authorized vehicles is indicated by *ntipos* and the capacities by *capactipo(i)*, $i = 1, \ldots, ntipos$.

A group of routes needs to be designed with a minimum total cost making sure that: the driving hours and times and respected, and that the number of students to be transported on each route does not exceed the total capacity of the vehicle assigned to that route. Considered in this way, this model is a specific case of the already established Vehicle Routing Problem (VRP), or to be more precise of the Vehicle Routing Problem with Time Windows (VRPTW), since there are time restrictions.

To design an algorithm for this problem, we have chosen to use a heuristic technique which provides a good solution in a more reasonable computation time. This strategy is based on two parts: the first consists basically in a series of Local Search processes which give quick solutions; the second (optional depending on the available calculation time) is based on a process of Tabu Search that includes a new Intensification method. Therefore, the proposed algorithm is a technique made up of various parts or subalgorithms based, for the most part, on neighborhood movement.

In the following two sections, the different definitions used here of neighborhood are presented. In the four section, he general algorithm structure is described. In the fifth section an algorithm based on a Tabu Search process is described which complements the preceding algorithm and which on many occasions improves the results. Finally, in the sixth section, the results obtained by the different algorithms and subalgorithms are described, as are the computation times used for a series of experiments on real problems with data from Burgos Province.