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

Joaquín A. Pacheco (1), Alberto Aragón (2) y Cristina Delgado (3)

(1) Catedrático de Escuela del Departamento de Economía Aplicada de la Universidad de Burgos.

(2) Profesor Titular del Departamento de Economía Aplicada de la Universidad de Burgos.

(3) Profesora Asociada del Departamento de Economía Aplicada de la Universidad de Burgos.

Resumen.- 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 describe 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 termino de racionalidad no hace solo referencia a la minimización del coste total del transporte, sino también al cuidado de determinados aspectos como el tiempo de los alumnos en el vehículo, elección de carreteras cómodas, etc.. en especial a partir de determinados acontecimientos recientes. Así mismo se muestran los resultados obtenidos con los datos del actual curso y con un conjunto de problemas simulados.

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

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,..., 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,...,n*. Para cumplir estos requerimientos la legislación autoriza diferentes tipos de vehículos; cada tipo de vehículo con un número de plazas diferente. Cada alumno no debe permenecer más de un determinado tiempo máximo en ruta, 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 $i, j \in \{1,2,...,n\}$ son conocidas. El número de tipos de vehículos autorizados se denota por *ntipos* y las capacidades por *capactipo(i), i = 1,..., 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 esta superen la máxima capacidad de todos los tipos de vehículo se trataría dicho localización como dos diferentes: una con un número de alumnos igual a dicha capacidad, y otra con el resto. En cualquier caso 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 1.997), 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 = $Pr * k$ si $k < 35$
  $Pr * 35 + (k-35)*(1'33)*Pr$ si $k > 35$

  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 = $100 * (M-33)$ si $M > 33$

  donde *M* es el nº de alumnos.

- Coste por Paradas = $75 * (L1-6)$, si $L1 > 6$

  siendo $L1 = \min\{L, M/3\}$ 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 formula 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 desde donde 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 estrategia de solución, puesto que de esta forma no va a ser necesario minimizar el número de vehículos a utilizar, es más, se ha

---

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

observado que en muchos casos aumentando el número de rutas se pueden llegar a soluciones menos costosas.

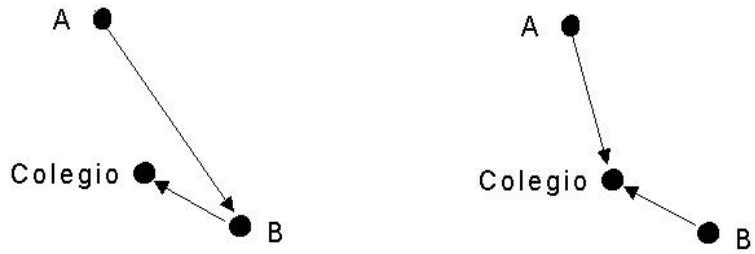


Figura 1.- Al no haber coste fijo por vehículo, y al comenzar a contar desde el primer punto de recogida (Origen y Destino no coinciden) es fácil encontrar soluciones que con más rutas son mas baratas que otras con menos. (La solución con las rutas A-Colegio y B-Colegio, recorre menos distancias que 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 del *Problema de Rutas de Vehículos con Ventanas de Tiempo* (VRPTW, Vehicle Routing Problem with Time Windows) al aparecer restricciones temporales.

Existen muchos algoritmos de solución para el VRP y el VRPTW en la literatura. Se pueden encontrar recopilaciones de los principales en trabajos como los de Bodin y Golden (1.981), Desrochers y otros (1.988), Haouri y otros (1.990), y Laporte (1.992). En los últimos años han tomado importancia el desarrollo de algoritmos basados en procesos denominados Metaheurísticos como *Temple Simulado, Búsqueda Tabú, GRASP*... especialmente a partir del trabajo de Gendreu y otros (1.991). Así se refleja en el trabajo de Osman (1.993) y más recientemente en el de Campos y Mota (1.995) o Kantoravdi, (1.995), Rego (1.998).

Al ser este un problema NP-Hard (complejidad no Polinomial, ver Lenstra et al, (1.981)), 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).

Por tanto se va a optar por el diseño de una técnica heurística, es decir, que no garantiza la obtención del óptimo pero si una buena solución en un tiempo de computación más razonable. Más concretamente, 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* usados. 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.- Vecindario tipo Or.

Se van a considerar dos tipos de soluciones vecinas:

- una basada en la idea propuesta por Or (1.976) para el *Problema del Viajante* o TSP,
- y otra más reciente que extiende la usada en los trabajos de Gendreu et al. (1.991), Osman (1.993), Campos y Mota (1.995),... que se explicará en la siguiente sección.

Se van a definir como soluciones vecinas las obtenidas por el método de intercambio propuesto por Or (1.976) que sean factibles. El método de intercambio Or es una variante de los conocidos intercambios r-óptimos desarrollados por Lin, (1.965) y Lin & Kernighan (1.973) 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 (1.991).
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:

ruta 1: 1 - 3 - 5 - 1; y *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 esta como una única secuencia.

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 otros dos. Nótese que con estos intercambios no se cambia el sentido de los diferentes tramos.

---

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

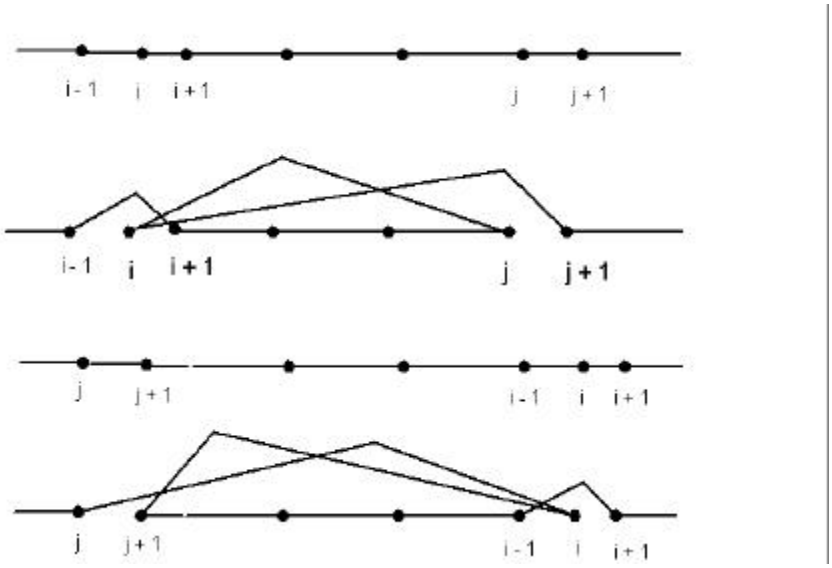


Figura 2.- Posible recolocación del elemento i hacia adelante y hacia atrás entre j y j+1.

En nuestro caso seguiremos la misma idea, pero sólo se consideraremos recolocaciones hacia adelante y no pondremos límite al tamaño de la cadena a recolocar (salvo el determinado por el tamaño del problema); 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.

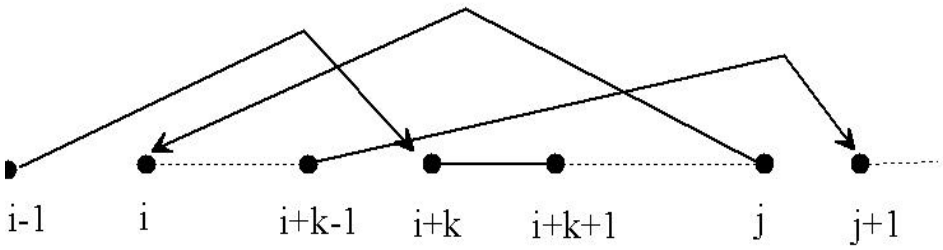


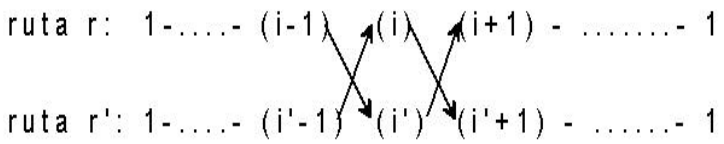
Figura 3.- Recolocación de una cadena de k elementos.

Para cada $s \in S$ se va a denotar por $N^k_1(s)$ al conjunto de soluciones factibles obtenidas por recolocaciones hacia adelante de cadenas de a lo sumo *k* elementos en *s*. Se denota por $N^\infty_1(s)$ el conjunto de soluciones factibles obtenidas por todas las recolocaciones.

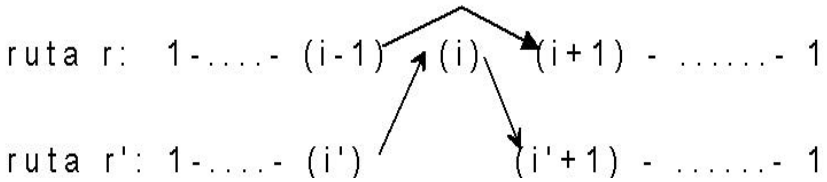
3.- Vecindarios tipo Gendreu-Clarke.

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:

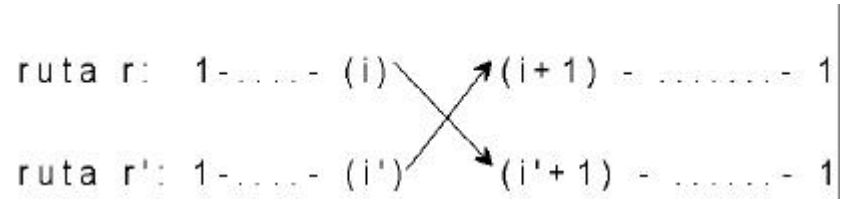
- Tipo I: Intercambio del elemento *i* de la ruta *r* y con el elemento *i'* de la ruta *r'*:
  - Eliminación de los arcos (*i*-1, *i*), (*i*, *i*+1), (*i'*-1, *i'*), (*i'*, *i'*+1)
  - Incorporación de los arcos (*i*-1, *i*'), (*i'*, *i*+1), (*i'*-1, *i*), (*i*, *i'*+1)



- 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 Gendreu et al. (1.991) y otros mencionados anteriormente; el tercero sin embargo no aparece en dichos trabajos: se basa algunas de las ideas propuestas por Clarke and Wright (1.964).

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 generados por cada uno de estos 3 tipos de intercambio como $N^1_2(s)$, $N^2_2(s)$ y $N^3_2(s)$ respectivamente; asi mismo se denota por $N_2(s)$ a la unión de estos 3 conjuntos.

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, (1.996) y (1.997b) 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.

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 $s' \in N^k_1(sf)$ *verificando* $f(s') = min\{f(s) / s \in N^k_1(sf)\}$

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

  hasta que $f(s) \geq f(sf)$, $\forall\, s \in N^k_1(sf)$.

- Procedimiento Búsqueda_Local_Ge(*sf*)

  Repetir

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

  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 $f(s) \geq f(sf)$, $\forall\, s \in N_2(sf)$.

Estos procedimientos de Búsqueda Local son análogos, aunque obsérvese como en el segundo caso cuando se realiza un intercambio (tipo Gendreu-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 Principal queda de la siguiente forma:

- Procedimiento Algoritmo_Inicial(Output s^* Î 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 (1.981) 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, (1.980)).

Tras el paso 2, cada una las rutas de la solución *s** obtenida es óptimo local con respecto a N^3_1, 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 Metaheurística 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 (1.989) y (1.990), y que esta teniendo grandes exitos y mucha aceptación en los últimos años. Según su creador, es un procedimiento que "*explora el espacio de soluciones más alla del óptimo local*", (Glover y Laguna, (1.993)). 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 *tabús* durante las siguientes iteraciones para evitar que se vuelvan a soluciones anteriores y el algoritmo cicle. El termino *tabú* hace referencia a un tipo de inhibición a algo debido a connotaciones culturales o historicas y que puede ser superada en determinadas condiciones. (Glover, (1.996)).

5.1.- El Algoritmo Básico

El algoritmo que se propone basicamente 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 $s_0 = s^*$; $T = \varnothing$, *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 $\geq$ *maxiter*

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

El objeto de aplicar el *criterio de aspiración* es determinar en que 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 produzca 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 va 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 que atributos tabús (arcos) van a estar activos se define *arco_tabú* una matriz *n* * *n* de la siguiente forma:

arco_tabú(r,l) = 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:

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

siendo *maxiter_tabu* 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)* = - *iter_tabu* (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 = ∅.

En cuanto al valor del parámetro *maxiter_tabu* (también denominado 'tamaño de la lista'), siguiendo la idea de Osman (1.993) este debe ser función del logaritmo del número de atributos tabú. Concretamento se toma:

$$maxiter\_tabu = 10*\ln(n)$$

Para el criterio de parada se toma:

$$maxiter = 10*n$$

En esta sección se ha descrito un algoritmo de Busqueda 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*). (Una amplia descripción de elementos que potencian la Búsqueda Tabú se puede encontrar en Glover y Laguna, (1.997) y (1.998)). A continuación se describe un procedimiento de Intensificación para complementar el algoritmo básico.

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 esta inspirada en los trabajos de Rossing, (1.997),

Rossing y otros, (1.997) y (1.998) 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:

ruta 1: 1 - 3 - 5 - 1; y 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_deConcentracion
  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) ≥ 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:

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

donde $max\_d = \max\{d(i, j) / i, j = 1, \ldots, n\}$; 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 *d*1:

- 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''*) *(considerendo f1 en ves 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 $f1(s') \geq f1(s_0)$

    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 *f*1, es decir según *d1* y buscando por tanto soluciones que contengan elementos de CS; cuando no hay mejora en *f*1, 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 *f*1, i.e. por *d1*, hacia soluciones que contengan el mayor número de elementos de CS posibles. 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_deConcentracion*;*
  Desde i:=1 to 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 *f*1 en vez de *f* al generar las soluciones iniciales y se ejecuta el procedimiento de *Búsqueda_Local_Guiada* dirigido por *f*1, 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, (1.998), 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 Algortimo Búsqueda Tabú queda de la siguiente forma:

- Procedimiento Búsqueda_Tabú_Principal;
  Ejecutar Búsqueda_Tabú_Básico
  Ejecutar Intensificación

Se ha de tener en cuenta las siguiente 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 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 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 uno de estas instancias se han 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, estas 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).

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 en un ordenador personal *Pentium MMX*-200 Mhz.

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	57,00 429	61.177,50 12	67.123,60 10 8,72	57.973,40 13 0,61	57.973,40 13 0,03	57.973,40 13 0,40	57.973,40 13 26,87	56.196,10 14 72,78
2º	BELORADO	24,00 175	21.045,00 5	22.960,50 4 0,29	16.962,50 8 0,22	16.962,50 8 0,02	16.962,50 8 0,12	16.962,50 8 7,08	16.962,50 8 20,74
3º	BRIVIESCA	24,00 101	25.051,30 6	24.055,00 5 0,75	23.333,80 6 0,07	23.333,80 6 0,01	23.333,80 6 0,06	23.197,40 7 7,56	23.197,40 7 7,02
4º	L. Mendoza BURGOS	19,00 127	18.608,80 3	18.104,40 4 0,26	17.542,40 4 0,03	17.542,40 4 0,00	17.542,40 4 0,03	16.880,30 5 5,30	16.863,60 5 6,26
5º	Diego Marín BURGOS	23,00 59	15.902,50 4	14.773,30 3 0,35	14.614,90 3 0,02	14.614,90 3 0,00	14.614,90 3 0,03	14.614,90 3 5,25	14.614,90 3 4,43
6º	Diego Siloé BURGOS	32,00 141	30.720,00 4	34.831,60 5 1,81	29.742,00 6 0,18	29.742,00 6 0,01	29.742,00 6 0,08	27.410,80 6 11,65	27.410,80 6 12,29
7º	S.de Colonia BURGOS	36,00 158	34.835,00 6	28.147,50 5 1,98	22.820,90 7 0,31	22.820,90 7 0,01	22.820,90 7 0,17	22.539,10 7 27,71	22.539,10 7 19,12
8º	LERMA	53,00 302	51.112,50 9	52.730,50 9 6,80	45.152,30 11 0,46	45.152,30 11 0,02	43.152,90 10 1,40	41.006,90 10 53,77	40.542,00 11 42,49
9º	MEDINA DE POMAR	39,00 182	31.075,00 5	35.005,80 6 2,87	30.790,40 7 0,19	30.790,40 7 0,01	30.790,40 7 0,15	30.189,10 7 16,55	30.189,10 7 15,81
10º	MELGAR	22,00 71	19.402,50 6	19.758,90 4 0,44	18.887,00 5 0,02	18.887,00 5 0,01	18.887,00 5 0,05	18.847,90 6 5,27	18.847,90 6 5,16
11º	MIRANDA	13,00 41	16.622,50 4	19.038,10 4 0,17	18.847,90 5 0,03	18.847,90 5 0,01	18.847,90 5 0,01	18.847,90 5 2,27	18.847,90 5 3,60
12º	QUINTANAR	4,00 105	8.025,00 2	6.925,00 2 0,01	3.312,50 4 0,02	3.312,50 4 0,00	3.312,50 4 0,00	3.312,50 4 0,66	3.312,50 4 1,79
13º	ROA	28,00 207	22.975,00 6	28.072,40 5 0,51	20.237,50 7 0,26	20.237,50 7 0,01	20.200,00 7 0,17	20.200,70 7 8,83	19.437,50 7 21,85
14º	SALAS	23,00 81	21.488,80 5	21.371,10 5 0,55	18.586,80 4 0,04	18.586,80 4 0,00	18.586,40 4 0,03	18.512,50 4 5,26	18.512,50 4 3,27
15º	VILLADIEGO	31,00 128	29.088,80 7	29.604,80 6 1,75	25.212,50 8 0,10	25.212,50 8 0,01	25.212,50 8 0,12	25.000,00 7 12,22	24.981,20 8 8,71
16º	VILLARCA-YO	9,00 23	8.252,50 2	13.205,50 2 0,05	10.847,00 2 0,01	10.847,00 2 0,01	10.847,00 2 0,00	10.847,00 2 1,61	10.847,00 2 1,75
T.	**TOTAL PROVINCIA**	**437,00 2.330**	**415.382,78 6**	**435.708,00,0 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

7.- Conclusiones y Reflexiones.

Como se ve en la tabla de resultados, el Algoritmo Inicial aporta soluciones rápidamente (apenas poco más de 11 segundos) que mejoran las soluciones actuales en más de un 10% en el coste. Si se dispone de mas 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.

Observese 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 han detectado alguna imprecisión (falta de algún tramo o cruces no detectados) que pueden dar lugar a que los caminos hallados, en algún caso concreto, sean mas 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 ha supuesto unas velocidades bastante moderadas para cada tipo de carretera; concretamente 90 km/h para Autopista y Autovía, 75 km/h para Nacional, 65 para Autonómica de 1er orden, 60 para Autonómicas de 2º orden, 55 para Autonómicas de 3º, 45 para Carreteras Sin Revestimeinto, 40 para carreteras de Enlace y 20 para Travesías. Obviamente, estas velocidades se pueden conseguir fácilmente en la realidad e incluso mayores. Lo importente 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...). 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.[3]

[3] **Reconocimiento**

Nuestro mas 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.

8.- Bibliografía.

BODIN L. D. and GOLDEN B. L. (1.981): "Classification in Vehicle Routing and Scheduling". *Networks*, vol.11, nº 2, 97-108.

CAMPOS V y MOTA E. (1.995): "Metaheurísticos para el CVRP". *XXII Congreso Nacional de Estadística e Investigación Operativa*. Sevilla, Noviembre 1.995.

CLARKE,G. and WRIGHT,J.W. (1.964): "Scheduling of Vehicles from a Central Depot to a Number of Delevery Points". *Oper.Res*., 12, (1.964), 568-581.

DESROCHERS M., LENSTRA J. K. SAVELSBERGH M. W. P. and SOUMIS F. (1.988): "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. (1.981). "A Generalized Assignment Heuristic for Vehicle Routing". *Networks*, vol.11, nº 2, 109-124.

GENDREU M., HERTZ A. and LAPORTE G. (1.991): "A Tabu Search Heuristic for Vehicle Routing Problem". Report CRT-777. *Centre de Recherche sur les Transports*. Univ. Montréal.

GLOVER F (1.989). Tabú Search: Part I. *ORSA Journal on Computing*, Vol 1, pp. 190-206. 1.989.

GLOVER F (1.990). Tabú Search: Part II. *ORSA Journal on Computing*, Vol 2, pp. 4-32. 1.990.

GLOVER F (1.996). Búsqueda Tabú en *Optimización Heurística y Redes Neuronales*. Adenso Díaz (coordinador). Paraninfo. Madrid. pp. 105-143.

GLOVER F y LAGUNA M. (1.993) Tabu Search in *Modern Heuristic Techniques for Combinatorial Problems*. C. Reeves, ed., Blackwell Scientific Publishing, pp, 70-141.

GLOVER F y LAGUNA M. (1.999) Tabu Search, aparecerá en *Handbook of Applied Optimization*, P.M.Paradalos and M.G.S.Resende (eds). Oxford Academic Press, (1.999)

HAOUARI M., DEJAX P. et DESROCHERS M. (1.990): "Les Problems de Tournées avec Contraintes des Fenênetres de Temps: L'Etat de l'Art". *Recherche Operationnelle/Operations Research*, vol. 24, nº 3, 217-244.

KONTORAVDIS G. and BARD J. F. (1.995): "A GRASP for the Vehicle Routimg Problem with Time Windows". *ORSA Journal on Computing*, 7: 10-23.

LAPORTE G. (1.992): "The Vehicle Routing Problem: An overview of exact and approximate algorithms". *European Journal of Operations Research*, 59, 345-358.

LENSTRA J. K. and RINNOY KAN A. H. G. (1.981): "Complexity of Vehicle Routing and Scheduling Problems". *Networks*, vol.11, nº 2, (1.981), 221-228. LIN,S. (1.965): "Computer Solutions to the Traveling Salesman Problem". *Bell Syst.Tech.Jou.*, vol 44, 2245-2269.

LIN S. y KERNIGHAN B. W. (1.973): "An Effective Heuristic Algorithm for the Traveling Salesman Problem". *Operations Researsch*, vol.20, 498-516.

NURMI K. (1.991): "Traveling Salesman Problem Tools for Microcomputers". *Computers & Ops.Res.* Vol. 18, nº 8, 741-749. OR,I. (1.976). 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. (1.993): "Metastrategy Simulated Annealing and Tabu Search Algorithms for the Vehicle Routing Problem". *Annals of Operations Research*, 41, pg. 421-451.

PACHECO J. y DELGADO C. (1.996). Adaptación del Algoritmo de Or al VRPTW con Carga y Descarga simultánea. *X Reunión ASEPELT-ESPAÑA* , Albacete, Junio 1.996.

PACHECO J. y DELGADO C. (1.997b). "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,* nº 7 , pgs. 79-100.

PACHECO J. y DELGADO C. (1.998). "Diseño de Metaheurísticos híbridos para Problemas de Rutas con Flota Heterogénea: Concentración Heurística". Trabajo sin publicar. Burgos. Septiembre 1.998.

REGO C. (1.998): "A Subpath Ejection Method for the Vehicle Routing Problem". *Management Science,* vol.44, nº 10, pg. 1447-1459.

ROSING K. E. (1.997). "Heuristic Concentration: An Introduction with Examples". *The Tenth Meeting of the European Chapter on Combinatorial Optimization*. Tenerife. Spain. May, 1.997.

ROSING K. E. and REVELLE C. S. (1.997). "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. (1.998). "Heuristic Concentration and Tabu Search: A head to head comparison". *European Journal of Operational Research* 104, 93-99.