NUEVAS TÉCNICAS METAHEURÍSTICAS: APLICACIÓN AL TRANSPORTE ESCOLAR.

Cristina Rocío Delgado Serna

NUEVAS TÉCNICAS METAHEURÍSTICAS: APLICACIÓN AL TRANSPORTE ESCOLAR.

Cristina Rocío Delgado Serna

Universidad de Burgos

DPTO. Economía Aplicada

Facultad de Ciencias Económicas y Empresariales

NUEVAS TÉCNICAS METAHEURÍSTICAS:

APLICACIÓN AL TRANSPORTE ESCOLAR.

Autora: Cristina R. Delgado Serna

Director: Joaquín A. Pacheco de Bonrrostro

de entre los cuales, los más utilizados en los últimos tiempos son

INTRODUCCIÓN

Los primeros contactos de la autora con problemas relacionados con el transporte terrestre, en aquel caso de mercancías, tienen su origen en la participación, dentro del Grupo SAAT (grupo de empresas dedicadas a la logística integral), en el proyecto tecnológico SILT (Sistemas Integrados de Logística del Transporte). Dicho proyecto, desarrollado entre los años 1.995-1.996, fue aprobado y cofinanciado por el CDTI con número de referencia 950107 y uno de sus dos objetivos era el desarrollo de un programa para el transporte de mercancías en Alemania.

Las tareas realizadas en el mencionado proyecto fueron principalmente las siguientes:

- Recopilación e integración de experiencias reales y normas legales en el programa de rutas desarrollado.
- Realización de pruebas e informes.
- Mejora del entorno gráfico, pasando el programa de Turbo Pascal a Delphi 1.0.
- Coordinación entre investigadores-programadores, usuarios e inspectores del CDTI.

Los tres grupos de metaheurísticos, se estudian respectivamente en los apartados 1.4, 1.5 y 1.6 y son los siguientes:

- *Metaheurísticos basados en Búsqueda por Entornos*; generan soluciones a partir de una solución s_0 mediante una operación denominada movimiento; las soluciones generadas integran lo que se denomina *vecindario* o *entorno* de esa solución.

- *Metaheurísticos Evolutivos o Basados en Población*; generan nuevas soluciones a partir de una, dos o más soluciones. En este grupo se han introducido los *Algoritmos Meméticos* y *Scatter Search* que en realidad son híbridos de los Metaheurísticos Evolutivos y de los basados en Búsqueda por Entornos.

- El tercer grupo "*Otros Metaheurísticos*", engloba metaheurísticos con otras formas de generar soluciones. En algunos de estos metaheurísticos es difícil observar el esquema común que decíamos que tienen los dos primeros grupos; este es el caso de Colonia de Hormigas.

Tradicionalmente, antes de la aparición de los metaheurísticos, la Búsqueda por Entornos abarcaba tan sólo la *Búsqueda Local*. Búsqueda Local es una estrategia descendente, de modo que la solución que se escoge es siempre mejor que la actual. Dada su importancia en muchos de los metaheurísticos utilizados (los del primer grupo y los híbridos que hemos incluido en el segundo), se

dedica el capítulo 2 a su estudio y se implementan y hacen pruebas con tres tipos *diferentes* de *entornos*. También es objeto de estudio en este capítulo la estrategia a seguir para elegir soluciones dentro del entorno. Se realizan pruebas con la estrategia de *Mayor Descenso* (explorar todo el entorno y elegir la mejor solución encontrada) y con la de *Primer Descenso* (elegir la primera solución que se encuentre que mejore la actual). Para reducir el tiempo de computación empleado para la búsqueda de soluciones en los vecindarios se ha adaptado a problemas de rutas la estrategia de *Búsqueda Local Rápida* propuesta por Bentley J.L. (1.992) y utilizada por también por Voudouris C. y Tsang E. (1.995) para el TSP.

En el capítulo 3 se implementa un algoritmo para el VRPTW utilizando *GRASP* y *Concentración Heurística* cuyo fin es comprobar la utilidad del uso de la filosofía de Concentración Heurística. Posteriormente se desarrolla un algoritmo de *Búsqueda Tabú* que utiliza *vecindarios* descritos en el capítulo dos, y un *conjunto de concentración* en la fase de intensificación. En su fase básica se implementa la estrategia de *Oscilación Estratégica*, para dar mayor robustez al algoritmo.

Se implementa un algoritmo en el capítulo 4 para el transporte escolar, eligiendo las estrategias que mejor resultado nos han ofrecido en las pruebas realizadas para el VRPTW. Se establece en como objetivo el apartado 4.2, la minimización del coste; en el 4.3

se aborda un doble objetivo: el social (reducción del tiempo que los escolares pasan en el autobús) y el económico (fijado un tiempo máximo, lo más reducido posible, minimizar el coste del transporte). Al final del capítulo 4, apartados 4.3.4, 4.3.5 y 4.3.6 se analizan los resultados obtenidos y se comparan con los presentados en otros trabajos (Corberán A. y otros (2.000)) y con las rutas que se llevaron a cabo en la realidad.

En el capítulo 5 se hace una síntesis de las aportaciones que se consideran importantes en este trabajo y se comentan una serie de reflexiones realizadas, así como las líneas de investigación que han quedado abiertas.

El capítulo 6 se dedica a apéndices y por último el 7 contiene las referencias bibliográficas utilizadas y/o referenciadas.

Expresar el agradecimiento en primer lugar al Grupo SAAT que me permitió conocer este interesante campo de investigación y sobre todo ver su aplicación práctica. En segundo lugar al investigador al frente del proyecto SILT y director de este trabajo, el catedrático Joaquín Pacheco Bonrostro por la ayuda que constantemente he recibido de él. En tercer lugar al director del departamento, Catedrático Alfredo García Güemes, por el apoyo que en todo momento me ha prestado. También quiero expresar mi agradecimiento a los responsables del transporte escolar en la provincia de Burgos por los datos e información facilitada para el

desarrollo de la aplicación. Agradezco también a Rafael Martí y Manuel Laguna por la ayuda prestada y sobre todo por la motivación e ilusión que personas de su relevancia inspiran. En último lugar y no menos importante a mis padres y a mis amigos por estar ahí.

Indice

CAPÍTULO I

ESTADO DEL ARTE

1.1 Optimización Combinatoria. Transporte Escolar.

1.2 Método de Mejora de Soluciones: Búsqueda Local.

1.3 Resolución de Problemas por Aproximación: Metaheurísticos.

1.4 Metaheurísticos Basados en Búsqueda por Entornos:
- 1.4.1 Temple Simulado.
- 1.4.2 Búsqueda Tabú.
- 1.4.3 GRASP
- 1.4.4 Concentración Heurística.

1.5 Metaheurísticos Evolutivos o Basados en Población:
- 1.5.1 Algoritmos Genéticos
- 1.5.2 Algoritmos Meméticos.
- 1.5.3 Búsqueda Dispersa.

1.6 Otros Metaheurísticos:
- 1.6.1 Colonia de Hormigas.

1.1 Optimización Combinatoria. Transporte Escolar.

OPTIMIZACIÓN COMBINATORIA.

En la vida empresarial se presentan problemas de decisión: existen recursos escasos (capital, mano de obra, tiempo, etc.) y requisitos mínimos que cumplir (fechas de entrega, producción necesaria, horas de descanso, etc.) que condicionan la elección de la estrategia más adecuada. El objetivo al tomar la decisión consiste en llevar a cabo el plan propuesto de una manera óptima (minimizar el coste o maximizar el beneficio). El problema consiste en la determinación del valor de unas magnitudes para que otra u otras alcancen un valor óptimo, máximo o mínimo. Estos problemas podrían plantearse matemáticamente de la siguiente forma:

minimizar/maximizar f(s)
sujeto a
$s \in S$
(S: conjunto de soluciones factibles)

Para el caso particular en que todas las funciones, objetivo y restricciones, son lineales, pronto apareció una nueva disciplina, la programación lineal, y un algoritmo eficiente, el simplex. Pero si las variables que intervienen son muchas el algoritmo puede no resultar eficiente, dado que el tiempo de cálculo necesario es

excesivamente largo; el tiempo de respuesta en ese caso no es operativo.

Existe un tipo concreto de problemas de optimización que en nuestro caso resulta especialmente interesante: problemas de **optimización combinatoria**. En ellos las variables de decisión son enteras y por lo general el espacio de soluciones está formado por ordenaciones o subconjuntos de números naturales.

Los dos problemas combinatorios más famosos quizás sean el *Problema de la Mochila* y el *Problema del Viajante* (*Knapsack Problem y Traveling Salesman Problem*). El primero de ellos consiste en seleccionar de entre un conjunto de n objetos, cada uno con un valor c_i y un volumen v_i, aquellos que quepan en un recipiente con capacidad V, y que tengan el mayor valor posible.

El Problema del Viajante (*TSP*), trata de determinar en que orden deben visitarse n ciudades distribuidas geográficamente, de forma que partiendo de una cualquiera, se recorra el menor número de kilómetros posible, y se vuelva al punto de partida tras visitar exactamente una vez cada una de ellas. Se trata, no de seleccionar un subconjunto de elementos, sino de elegir una permutación de las n ciudades de forma que las distancias recorridas sean mínimas. La importancia del Problema del Viajante, además del gran número de aplicaciones, estriba en el hecho de que combina las características típicas de un gran número de problemas de optimización

combinatoria, contiene dos elementos que hacen atractivo un problema: planteamiento sencillo y dificultad de resolución.

En muchas ocasiones se ha tratado de resolver este tipo de problemas combinatorios formulándolos como modelos de programación lineal entera, donde la pertenencia o no de una variable al subconjunto buscado se representa a través de su "función característica". En el caso del Problema de la Mochila tendríamos:

$$max \sum_{i=1}^{n} c_i x_i$$

con la restricción

$$\sum_{i=1}^{n} v_i x_i \leq V$$

$$x_i \in \{0,1\} \quad \forall i \in \{1,.., n\}$$

En el caso del TSP:

$$min \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{n} c_{ij} x_{ijk}$$

con las restricciones

$$\sum_{\substack{j=1 \\ j \neq i}}^{n} \sum_{k=1}^{n} x_{ijk} = 1 \quad i = 1,..., n$$

(de cada ciudad se sale una vez)

$$\sum_{\substack{i=1 \\ j \neq i}}^{n} \sum_{j=1}^{n} x_{ijk} = 1 \quad k = 1,..., n$$

(cada tramo k une sólo dos ciudades)

$$\sum_{i=1}^{n}\sum_{\substack{k=1\\ i\neq j}}^{n} x_{ijk} = 1 \quad j = 1,\ldots, n$$

(a cada ciudad sólo se llega una vez)

$$\sum_{\substack{i=1\\ i\neq j}}^{n} x_{ijk} = \sum_{\substack{r=1\\ r\neq j}}^{n} x_{jrk+1} \quad j = 1,\ldots, n; \quad k = 1,\ldots, n$$

(el punto de llegada del tramo k es el punto de partida del tramo k+1)

$$\sum_{j=2}^{n} x_{1j1} = 1$$

(en el primer paso se sale de n)

$$\sum_{i=2}^{n} x_{i1n} = 1$$

(en el último paso se llega a 1)

$$x_{ijk} \in \{0,1\} \ \forall i, j, k \in \{1,\ldots, n\}$$

En los problemas de tipo combinatorio existe siempre un procedimiento elemental para determinar la solución óptima buscada: realizar una enumeración completa del conjunto de soluciones, es decir, generar todas las soluciones que satisfagan las restricciones (soluciones factibles), calcular para cada una el coste asociado y elegir finalmente la que haya dado lugar al mejor de ellos. El problema es que aunque este método teóricamente nos lleva siempre a la solución óptima buscada, no es eficiente, pues el tiempo de cálculo necesario crece de forma superpolinomial (normalmente exponencial) con el número de items del problema. Por consiguiente, en muchos casos, la solución óptima no puede ser obtenida en un tiempo razonable.

Se distinguen dos tipos de problemas combinatorios atendiendo a su complejidad computacional:

- los pertenecientes a la *clase P*, para los que se conocen algoritmos y se considera que son resolubles eficientemente; necesitan un tiempo polinomial para ofrecer la solución óptima. Una aplicación es de complejidad polinómica o polinomial (*clase P*) si se puede resolver en un tiempo de computación polinomial que dependerá del tamaño del problema, $O(p(n))$: existe un algoritmo para el que se puede encontrar un polinomio $p(n)$ tal que este algoritmo puede resolver cualquier ejemplo de tamaño n en un tiempo $O(p(n))$.

- los que pertenecen a la *clase NP (non-deterministic polynomial)*, que son la mayoría de los problemas que aparecen en la gestión empresarial e ingeniería; no pueden ser resueltos en tiempo polinomial (no se conoce un algoritmo polinomial de resolución), aunque sea posible dada una solución comprobar en tiempo polinomial, si su coste es mejor que un determinado valor. En este tipo de problemas un consumo masivo de memoria no mejora el tiempo de ejecución.

Dentro de los problemas *NP* se distingue entre *NP-hard* que es el problema de optimización y *NP-completo* que es el problema de decisión correspondiente. En la práctica ambos conceptos se

confunden y se utilizan indistintamente. Cuando se quiere resolver problemas *NP- hard* de gran tamaño se debe escoger entre dos opciones: elegir un algoritmo que asegure la obtención del óptimo, lo que como se ha comentado tiene el riesgo de emplear un tiempo de computación excesivo, a veces no disponible; o elegir un algoritmo que proporcione una buena solución rápidamente (tiempo de computación polinomial), pero sin la garantía de que ésta sea la óptima.

Dada la dificultad para resolver de forma exacta una serie de problemas combinatorios de interés práctico, comenzaron a aparecer métodos que proporcionan soluciones factibles, que aunque no optimizan la función objetivo, se acercan en un tiempo computacional razonable a un valor cercano al óptimo; este tipo de métodos se denominan *Heurísticos o Aproximados*. El término *heurístico* deriva de la palabra griega *heuriskein* que significa encontrar o descubrir; se ha adoptado este término en el ámbito de la optimización para esta forma de resolución de problemas que no garantiza el óptimo, pero sí garantiza una buena solución.

Aunque los heurísticos en un principio no fueron bien vistos, se han ido aceptando dada su utilidad al dar soluciones a problemas reales, más aun a partir de los setenta con la proliferación de resultados en el campo de la complejidad computacional. El crecimiento espectacular en el desarrollo de métodos heurísticos puede ser constatado examinando el gran número de artículos en las principales revistas de Investigación Operativa que tratan sobre

estos. Además, han aparecido publicaciones específicas como el *Journal of Heuristics* publicado por Kluver Academic Press.

Los métodos heurísticos tienen la ventaja añadida de que pueden ser más fácilmente adaptables para solucionar modelos más complejos. Es interesante su utilización cuando no existe un método exacto de resolución o éste requiere mucho tiempo de cálculo o memoria, cuando no se necesita la solución óptima, cuando los datos son poco fiables, cuando hay limitaciones de tiempo (es frecuente en la vida real la necesidad de una respuesta rápida) o espacio, como paso intermedio en la aplicación de otro algoritmo, etc.

Según Zanakis S.H. y Evans J.R. (1.981) se pueden definir como *"procedimientos simples, a menudo basados en el sentido común, que se supone ofrecerán una buena solución (aunque no necesariamente la óptima) a problemas difíciles, de un modo fácil y rápido"*.

Existen diferentes **tipos de heurísticos**, según el modo en que buscan y construyen sus soluciones. Una posible clasificación basada en Silver E.A, Vidal R.V y Werra (de) D. (1.980) es la siguiente:

A. Métodos constructivos.
B. Métodos de descomposición.
C. Métodos de reducción.

D. Métodos de manipulación del modelo.

E. Métodos de búsqueda por entornos.

A. MÉTODOS CONSTRUCTIVOS

Consisten en ir añadiendo paulatinamente componentes individuales a la solución, hasta que se obtiene una solución factible. El más popular de estos métodos lo constituyen los algoritmos golosos o devoradores (greedy), los cuales construyen paso a paso la solución seleccionando en cada paso la mejor opción.

B. MÉTODOS DE DESCOMPOSICIÓN

Se trata de dividir el problema en subproblemas más pequeños, siendo el output de uno el input del siguiente, de forma que al resolverlos todos obtengamos una solución para el problema global, "divide y vencerás".

C. MÉTODOS DE REDUCCIÓN

Tratan de identificar alguna característica que presumiblemente deba poseer la solución óptima y de ese modo simplificar el problema.

D. MANIPULACIÓN DEL MODELO

Modifican la estructura del modelo con el fin de hacerlo más sencillo de resolver, deduciendo a partir de su solución la solución

del problema original. Pueden consistir en reducir el espacio de soluciones o incluso aumentarlo.

E. MÉTODOS DE BÚSQUEDA POR ENTORNOS

Parten de una solución factible inicial y mediante alteraciones de esa solución van pasando de forma iterativa, y mientras no se cumpla un determinado criterio de parada, a otras soluciones factibles de su entono (soluciones vecinas), almacenando la mejor de las soluciones visitadas. Tradicionalmente la búsqueda por entornos abarcaba sólo a los *algoritmos de búsqueda local*: parten de una solución inicial y en cada paso buscan y seleccionan una solución vecina de la actual que la mejore; se sustituye la solución actual por esta nueva solución y se repite el proceso hasta que no se pueda encontrar ninguna solución vecina mejor que la actual; en este caso se dice que existe un mínimo/máximo vecinal o local. La utilización de Búsqueda Local da siempre como solución final un óptimo local que, en ocasiones, puede ser global.

TRANSPORTE ESCOLAR.

El Problema del Viajante (TSP) comentado al comienzo de este apartado, puede ir modificándose a necesidades concretas y dar lugar a otros problemas más complejos como el denominado *Problema de Rutas de Vehículos (VRP)*. Se trata de satisfacer la demanda conocida de *n-1* clientes repartidos geográficamente, utilizando *m* vehículos de capacidad también conocida y

recorriendo la mínima distancia posible, de forma que cada cliente sea visitado por un solo vehículo. Cuando los clientes llevan asociados unos intervalos de tiempo en los que deben de ser visitados, se tiene el VRPTW (problema de rutas de vehículos con ventanas de tiempo).

El problema estudiado en este trabajo es el del transporte escolar y se considera como un VRPTW o como un VRP con restricciones de duración de las rutas. La analogía planteada es la siguiente:

Transporte Escolar	VRP(TW*)
√ Conjunto de Localizaciones	√ Localización de los Clientes
√ Número de alumnos a recoger en cada localización	√ Mercancía a entregar
√ Autobuses escolares disponibles	√ Vehículos disponibles
√ Tiempo máximo de duración de la ruta √ Tiempo máximo de llegada al centro escolar	√ (*)Ventanas de Tiempo o restricciones de duración de la ruta

Fig 1.1 Analogía entre el problema del Transporte Escolar y el VRP(TW)*

Con el fin de encontrar una buena solución, se desarrolla en el capítulo tres un heurístico para el VRPTW. La calidad del algoritmo desarrollado se comprueba comparando los resultados

que aporta para librerías de problemas disponibles en la red y resueltos por medio de otros heurísticos.

En el capítulo cuatro se adapta y aplica a los datos del transporte escolar de la provincia de Burgos en el curso 98-99. Como comprobaremos dicho heurístico aporta una solución mejor, tanto en coste como en duración de las rutas, que la que se utilizó en el mencionado curso. Además se realiza una comparativa con otro heurístico desarrollado por Corberán A. y otros (2.000) para el mismo problema.

1.2 Método de mejora de soluciones: Búsqueda Local.

Un método básico de mejora de soluciones, que además es de gran importancia en muchas de las técnicas heurísticas más recientes, es la Búsqueda Local. Los procedimientos de Búsqueda Local, también llamados de mejora, se basan en explorar el *entorno o vecindario* de una solución con el fin de encontrar otra mejor. Utilizan una operación básica llamada *movimiento*, que aplicada sobre los diferentes elementos de una solución, proporciona las soluciones de su entorno. Un procedimiento de búsqueda local

parte de una solución inicial s_0, explora su entorno, y escoge en él una nueva solución s que mejore la actual.

Vamos a considerar un problema de optimización combinatoria en el que se trata de minimizar el valor de la función objetivo. Sea S el conjunto de soluciones factibles y f: $S \rightarrow R$ la función a minimizar[1], se denota por $N(s)$ el conjunto de soluciones vecinas de una determinada solución $s \in S$ (vecindario o entorno de s). El procedimiento de Búsqueda Local en pseudocódigo se puede escribir como sigue:

Procedimiento Búsqueda_Local

Seleccionar una solución inicial $s_0 \in S$

Repetir

Seleccionar $s \in N(s_0) / f(s) < f(s_0)$ *por un método preestablecido*

Reemplazar s_0 *por* s

Hasta que $f(s) \geq f(s_0), \ \forall s \in N(s_0)$

Un procedimiento de Búsqueda Local queda completamente determinado al especificar un entorno y el criterio de selección de una solución dentro del entorno.

1 Mientras no se especifique lo contrario, se considerarán problemas de minimización, donde "f" será la función objetivo y "S" el conjunto de soluciones factibles.

Las formas más usuales de seleccionar $s \in N(s_0)$ son: explorar en todo $N(s_0)$ y tomar el correspondiente al menor valor de f (mayor descenso), o seleccionar el primero que mejora la solución actual (primer descenso). La búsqueda se detiene cuando la solución no puede ser mejorada. A la solución encontrada se la denomina *óptimo local respecto al entorno definido*.

Por lo general, la complejidad computacional del procedimiento de Búsqueda Local depende del tamaño del vecindario y también del tiempo necesario para evaluar cada movimiento.

El inconveniente de esta estrategia siempre descendente es que, en la mayoría de los casos, se converge a mínimos locales que no son globales. Para ilustrar este problema considérese la figura 1.2.

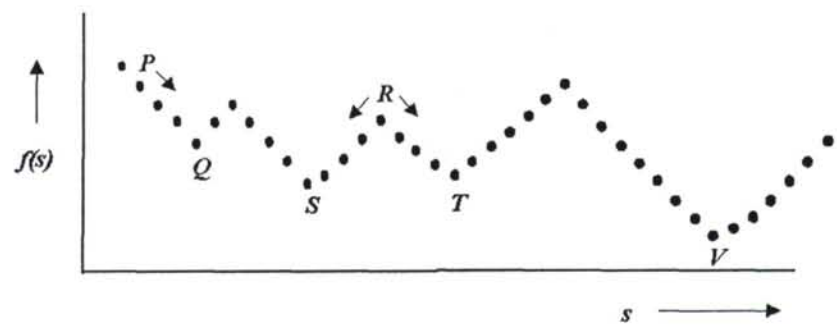


Fig. 1.2 Búsqueda Local. Inconvenientes. (Tomada de Dowsland (1.993)).

Se muestra un simple ejemplo de una función en la que cada solución tiene dos soluciones vecinas, representadas por los puntos

que tiene a la izquierda y a la derecha. Una estrategia descendente siempre se dirige hacia el fondo del "valle" que contiene al punto de salida. Por ejemplo, si la solución inicial es *P*, siempre se finalizará en el punto *Q*. La única excepción se da cuando el punto inicial es un máximo local (por ejemplo *R*), en cuyo caso se puede llegar al fondo de diferentes valles (*S* y *T*). Otro inconveniente que se observa es la dependencia de la solución inicial de la que se parte: el punto inicial determina en que óptimo local se cae. Estas observaciones se pueden extender a estructuras más complicadas o con más dimensiones.

Para evitar estos problemas se han sugerido algunas posibles soluciones: repetir el algoritmo con diferentes puntos de partida, o considerar una estructura vecinal más compleja (aumentar el vecindario). Pero ninguna de estas variantes ha resultado ser totalmente satisfactoria, bien por la escasa mejoría en las soluciones finales o por el excesivo aumento en el tiempo de computación.

En la figura anterior (figura 1.2) se observa que para evitar caer en mínimos locales de los que no se pueda salir, deberían ser permitidos algunos movimientos hacia arriba, aunque empeoren momentáneamente la solución actual. Como se verá más adelante, algunos métodos heurísticos, permiten estos movimientos hacia arriba de forma controlada, mejorándose en muchos casos el valor de la solución final.

1.3 Resolución de problemas por aproximación: Metaheurísticos.

Según Glover F. y Laguna M. (1.997):

"*... metaheurística se refiere a una estrategia maestra que guía y modifica otras heurísticas para producir soluciones más allá de aquellas que normalmente se generan en una búsqueda de óptimos locales.*"

Los heurísticos guiados por tales meta-estrategias pueden ser procedimientos de alto nivel o pueden incorporar nada más que una descripción de movimientos disponibles para transformar una solución en otra, junto con una norma de evaluación asociada.

Osman I.H. y Kelly J.P. (1.996), introducen la siguiente definición:

"*Los procedimientos Metaheurísticos son una clase de métodos aproximados que están diseñados para resolver problemas difíciles de optimización combinatoria, en los que los heurísticos clásicos no son ni efectivos ni eficientes. Los Metaheurísticos proporcionan un marco general para crear nuevos algoritmos híbridos combinando diferentes conceptos derivados de: inteligencia artificial, evolución biológica y mecanismos estadísticos*".

Se pueden encontrar más definiciones en correos dirigidos a Jin-Kao Hao y que están disponibles en la WEB del foro "Modernos Heurísticos", mantenido por I. Osman (*http://www.mailbase.ac.uk/lists-k-o/modern-heuristics/*).

- Pablo Moscato: " *Estrategia de alto nivel que guía un heurístico ad-hoc inferior para un problema dado*".

- I.H. Osman, Jim P. Kelly: "*Un meta-heurístico es un proceso generacional iterativo que guía un heurístico subordinado combinando inteligentemente diferentes conceptos para explorar y explotar los espacios de búsqueda usando estrategias de información con el fin de encontrar eficientemente soluciones casi-óptimas* ".

- Eric Taillard: " *Un meta-heurístico es un conjunto de conceptos concebidos para la optimización combinatoria que permiten ayudar a la concepción de métodos heurísticos para un problema genérico de optimización*".

- Stefan Voss: "*Un metaheurístico es un proceso maestro iterativo que guía y modifica las operaciones de un heurístico inferior para producir eficientemente soluciones de alta calidad. Puede manipular una solución simple completa (o incompleta) o una colección de soluciones en cada iteración. El*

heurístico inferior puede ser un procedimiento de alto (o bajo) nivel o una búsqueda local simple o un método de construcción"

- Rachid Chelouah: "*La palabra "heurística" viene del griego heurein (descubrir) y califica todo lo que sirve al descubrimiento, a la invención y a la investigación* ".

El contraste entre la orientación de *metaheurísticos* y la orientación de *óptimos locales* es significativo. Durante varios años, la concepción primera de un procedimiento heurístico (concepción que prevalece aún hoy) era representada tanto por reglas claras de "hojeo" como por unas reglas iterativas que terminaban en el momento en que no se podía encontrar una solución inmediatamente accesible que mejorara la última encontrada. Tales heurísticos iterativos se refieren a menudo a métodos de búsqueda local. En consecuencia la emergencia de métodos que se apartaban de este diseño clásico -y que lo hacían por medio de diseños maestros organizados- constituyeron un importante avance. Un conocimiento general de estos avances surgieron en los ochenta, aunque su semilla retrocede más allá.

La evolución de los metaheurísticos en los años 90 ha mejorado de forma explosiva. Los metaheurísticos en sus formas modernas están basados en una variedad de interpretaciones de lo que constituye la búsqueda inteligente. Esas interpretaciones se pueden

usar con propósitos de clasificación. Es posible encontrar diferentes clasificaciones por ejemplo en Díaz A. y otros (1.996) o en Glover F. y Laguna M. (1.997), pero tal y como estos últimos afirman "*una clasificación rigurosa de diferentes metaheurísticos es una empresa difícil y peligrosa ya que los principales partidarios de métodos alternativos a menudo difieren entre ellos mismos en la naturaleza esencial de los métodos a los que se adhieren*".

Entre las técnicas metaheurísticas que más éxito han tenido en la resolución de problemas de tipo combinatorio se pueden destacar las siguientes (indicando para cada una los trabajos pioneros):

- Algoritmos Genéticos: John Holland (1.975).
- Temple Simulado: Kirkpatrick S. y otros (1.983).
- Búsqueda Tabú: Fred Glover (1.989) y (1990-a).
- Algoritmos Meméticos: Pablo Moscato (1.989).
- GRASP: Feo T.A. y Resende M.G.C. (1.989) y (1.995).
- Búsqueda Reactiva: Roberto Battiti (1.996).
- Colonias de Hormigas: Marco Dorigo y otros (1.996).
- Concentración Heurística: Rosing K.E (1.997) y Rosing K.E. y Revelle C.S. (1.997).
- Búsqueda Dispersa: Fred Glover (1.998), Manuel Laguna (1.999).
- Búsqueda Local Guiada: Voudouris C. y Tsang E. (1.999).

1. ALGORITMO GENÉTICO (GA)

Está basado en los *procesos biológicos de evolución genética*; es un algoritmo poblacional pionero en la implementación de métodos que explotan la idea de combinación de soluciones. Tiene un componente de búsqueda *aleatoria*.

2. TEMPLE SIMULADO (SA)

Creado por analogía con los *procesos físicos de templado de sólidos*; permite movimientos que empeoran la solución actual con una determinada *probabilidad*, facilitando así la salida de los óptimos locales.

3. BÚSQUEDA TABÚ (TS)

Procedimiento en el que, una vez que se llega a un óptimo local, se permiten *movimientos* que empeoran la solución actual, para escapar de dicho óptimo. Simultáneamente los últimos movimientos son calificados como *tabús* durante un determinado número de iteraciones para evitar que se vuelva a soluciones anteriores y el algoritmo cicle.

4. GRASP

Proceso iterativo; cada iteración consta de dos fases: en la primera se construye una solución mediante un procedimiento *ávido-aleatorio* y en la segunda se mejora la solución obtenida mediante *Búsqueda Local*.

5. ALGORITMOS MEMÉTICOS (MA).

Son híbridos de *Algoritmos Genéticos* y heurísticos basados en *Búsqueda Local*; las combinaciones de GAs con heurísticos constructivos o métodos exactos pertenecen también a los MA.

6. COLONIAS DE HORMIGAS (AC)

Algoritmo que simula el comportamiento *retroalimentado* de una colonia de hormigas cuando van en busca de alimentos, siguiendo el reguero de *pheromone* que depositan sus compañeras al caminar.

7. BÚSQUEDA REACTIVA (RS)

La búsqueda reactiva propone la integración de procedimientos de autoajuste de parámetros, en algoritmos basados en búsqueda por entornos.

8. CONCENTRACIÓN HEURÍSTICA (HC)

Procedimiento que consta de dos fases: en la primera se generan múltiples soluciones según diferentes criterios; en la segunda, con los elementos de las mejores soluciones se construye el denominado *Conjunto de Concentración* y se resuelve el problema original pero restringiendo la selección de elementos a los del Conjunto de Concentración.

9. BÚSQUEDA DISPERSA (SS)

Construyen soluciones mediante combinaciones lineales de otras que constituyen el llamado *Conjunto de Referencia*. Dicho conjunto se renueva con las nuevas soluciones generadas y está

formado por soluciones consideradas "buenas" según el valor de la función objetivo o el grado de dispersión que aporten al conjunto.

10. BÚSQUEDA LOCAL GUIADA (GLS)

Secuencia de procedimientos de *Búsqueda Local*; al finalizar cada uno de ellos se modifica la función objetivo *penalizando* determinados elementos que aparecen en el último óptimo local, estimulando de esta forma la *diversificación* de la búsqueda.

En la figura 1.3 se muestra una tabla con las diferentes principales características de cada uno de los Metaheurísicos descritos anteriormente[2]. En apartados posteriores se explican algunas de estas técnicas con más detalle. Para ello, se han considerado por un lado los metaheurísticos basados en búsqueda por entornos (apartado 1.4), por otro los metaheurísticos evolutivos o basados en población (apartado 1.5) y por último, otros metaheurísticos (apartado 1.6). Los primeros modifican determinadas características de una solución para generar a partir de ella un conjunto de soluciones, denominadas soluciones vecinas, que constituyen lo que se denomina entorno o vecindario; los segundos combinan soluciones para crear otras nuevas; los terceros son el resultado de nuevas ideas, muchas de ellas inspiradas por la continua investigación en los dos anteriores.

---

[2] Características de las estrategias o principios básicos en cada caso.

Algoritmo	Determinístico /Aleatorio	Uso de memoria	Poblacional	Uso de movimientos por entornos	Basado en procesos
GENÉTICOS	Aleatorio	No	Si	No	Biológicos
TEMPLE SIMULADO	Aleatorio	No	No	Si	Físicos
BÚSQUEDA TABÚ	Determinístico	Si	No	Si	-
GRASP	Aleatorio	No	No	Si	-
MEMÉTICOS	Aleatorio	No	Si	Si	Biológicos
COLONIA DE HORMIGAS	Aleatorio	Si	No	No	Biológicos
BÚSQUEDA REACTIVA	Determinístico	Si	No	Si	-
CONCENTRACIÓN HEURÍSTICA	Aleat/Deter	Si	No	Si	-
BÚSQUEDA DISPERSA	Determinístico	Si	Si	Si	-
BÚSQUEDA LOCAL GUIADA	Determinístico	Si	No	Si	-

Fig. 1.3 Tabla de características de diferentes Metaheurísticos.

1.4 Metaheurísticos basados en Búsqueda por Entornos.

La Búsqueda por Entornos utiliza una operación básica denominada *movimiento*, que consiste en la modificación de características o elementos de una solución. Esta operación aplicada sobre los diferentes elementos de la solución, proporciona las soluciones de su entorno, las cuales constituyen el *vecindario* o *entorno* de dicha solución.

1.4.1 Temple Simulado.

Kirkpatrick S. y otros (1.983), e independientemente Cerny V. (1.985), proponen un procedimiento para obtener soluciones aproximadas llamado Temple Simulado (*Simulated Annealing* o *SA*, en inglés). Se pueden considerar como una variante de los métodos de búsqueda local, en la que se permite, como se verá más adelante, empeoramientos en la solución actual aunque de forma controlada.

Los autores mencionados introdujeron el concepto de *templado* en optimización combinatoria. Este concepto está basado en una

estrecha analogía entre el proceso físico de *templado* y los problemas combinatorios.

La idea original que dio lugar a esta metaheurística es el denominado "*algoritmo de Metrópolis*", Metrópolis y otros (1.953), bien conocido en el mundo de la Química-Física. Para estudiar las propiedades de equilibrio, Metrópolis utilizó *el "método de Montecarlo"*, que es el más usado en Mecánica Estadística para estudiar el comportamiento microscópico de los cuerpos.

Los puntos que a continuación se desarrollan en este apartado son los siguientes:

1. Proceso Físico de Templado.
2. El Algoritmo de Metrópolis.
3. El Temple Simulado.
4. Diferentes Planes de Templado.
5. Parámetro Inicial c_0 (temperatura inicial).
6. Función de Decrecimiento (velocidad de enfriamiento).
7. Criterio de Parada.
8. Longitud de los Ciclos de Procesamiento.
9. Propuesta de un Híbrido.
10. Aplicaciones del Temple Simulado a Problemas de Rutas.

1.4.1.1 Proceso Físico de Templado.

Las moléculas de una sustancia pueden tener distintos niveles de energía. El menor de estos niveles es el llamado "estado fundamental", ε_0. A una temperatura de 0° *K* todas las moléculas están en su estado fundamental, pero se sabe que un trozo de sustancia a alta temperatura probablemente posea un estado de energía más alto que otro idéntico a temperatura menor.

Cada una de las maneras en que las moléculas pueden estar distribuidas entre los distintos niveles de energía recibe el nombre de *microestado*. Se denomina Ω al conjunto de todos los posibles microestados y *número de ocupación*, n_i al número de partículas en el nivel de energía *i*. En la figura 1.4, se observa un ejemplo de Templado; como se aprecia en la figura, el número de moléculas en los estados superiores decrece para una temperatura *T* fija.

Para reducir la energía de la sustancia al menor valor posible, bajar simplemente la temperatura al cero absoluto no asegura necesariamente que la sustancia alcance su configuración energética más baja posible. En *física termodinámica*, se conoce como *templado* a un proceso termal para obtener los estados de más baja energía de un sólido en un recipiente. El proceso contiene los siguientes pasos (tomados de Barker J.A. y Henderson D. (1.976)):

- Elevar la temperatura del recipiente, al menos hasta conseguir que el sólido se funda.
- Bajar la temperatura del recipiente muy suavemente hasta que las partículas se estabilicen, es decir, hasta llegar al estado sólido; entonces se dice que se ha producido *congelación.*

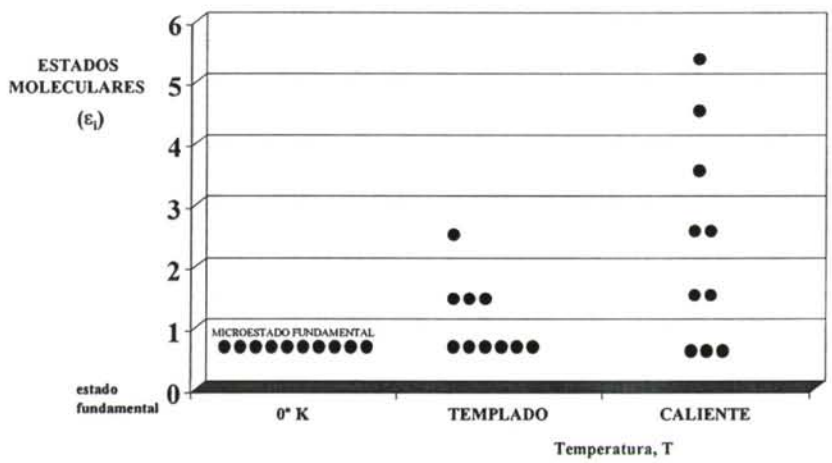


Fig. 1.4 Diferentes Microestados según la temperatura. (Tomada de Adenso Díaz y otros, 1.996).

Durante la fase líquida todas las partículas del material se mueven de forma aleatoria. Cuando se llega al estado sólido las partículas están ordenas en una estructura *enrejada* con energía mínima. Esta estructura se consigue solamente si la temperatura inicial es suficientemente alta, y el enfriamiento se hace de forma suficientemente lenta; de lo contrario, el material alcanza una estructura meta-estable con mayor valor energético.

El opuesto del templado es un proceso conocido como *oscurecimiento*, (*quenching*) en el que se hace descender la temperatura del recipiente instantáneamente. El resultado de este proceso son estados meta-estables.

Considérese el ejemplo de un bloque de silicio que se está cultivando en un horno para utilizarlo como sustrato en dispositivos de circuitos integrados. Para ello es importante que la estructura cristalina sea una red perfecta, regular a temperatura ambiente (esta configuración posee el valor energético mínimo). Una vez que el bloque de silicio se ha fundido, debe enfriarse adecuadamente para asegurar que se forme correctamente la red cristalina. Un enfriamiento rápido puede dar lugar a muchas imperfecciones en la estructura cristalina, o a una estructura vítrea, en absoluto regular. Estas dos configuraciones poseen una energía mayor que la red cristalina perfecta: representan mínimos locales de energía.

Para entender el proceso de templado obsérvese la figura 1.5 que ilustra un argumento intuitivo, empleado por Hinton G.E. y Sejnowski T.J. (1.986) en su descripción del temple simulado.

Se muestra un paisaje energético sencillo con dos mínimos: un mínimo local *Ea* y un mínimo global *Eb*. El sistema comienza con una cierta energía *Es*. Se puede establecer una analogía con una bola que rodase colina abajo. La bola rueda hacia el mínimo local *Ea*, pero inicialmente no tiene energía suficiente para rodar hasta

el otro lado y bajar al mínimo global. Si sacudimos todo el sistema es posible que la bola reciba el empujón necesario para que suba la colina. Por otra parte, una sacudida muy fuerte podría también sacar la bola del valle del mínimo global y devolverla al lado del mínimo local.

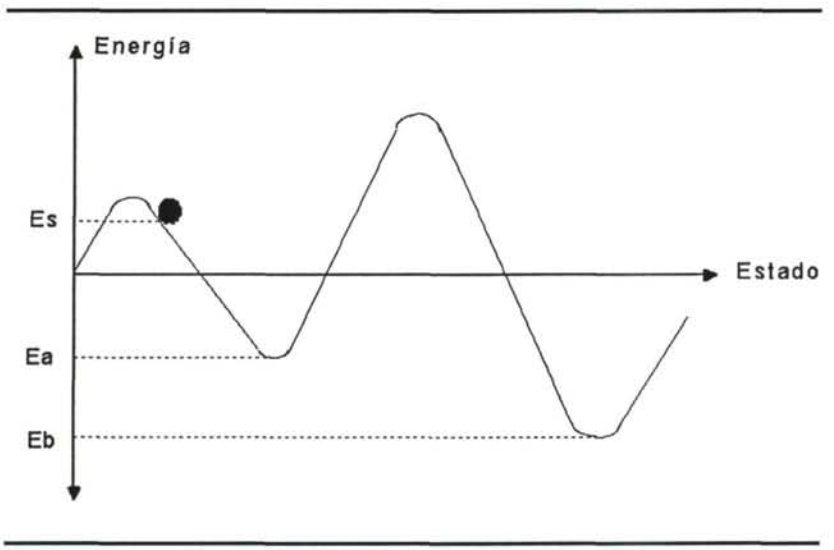


Fig. 1.5 Ilustración del proceso físico de templado.

El templado representa un compromiso entre una sacudida fuerte y una sacudida suave. A temperaturas altas la gran energía termal corresponde a una sacudida muy fuerte; a temperaturas bajas se corresponde con una sacudida suave. Para templar un objeto elevamos la temperatura y a continuación la disminuimos gradualmente hasta temperatura ambiente, permitiendo al objeto alcanzar el *equilibrio termal* en cada temperatura. La técnica de disminución gradual de la temperatura es el mejor medio para asegurar evitar un mínimo local de energía que no sea global, ya

que conforme disminuye esta, disminuye rápidamente la probabilidad de que se acepte una solución peor que la actual.

El equilibrio termal está caracterizado por la *distribución de Boltzmann*, ver Toda M. y otros (1.983), que indica la probabilidad de que un determinado material alcance un estado i con una energía E_i a la temperatura T:

$$P_T(X=i)=\frac{1}{Z(T)}\exp\left(\frac{-E_i}{k_BT}\right) \qquad (1.4.1)$$

donde X, es la variable aleatoria que indica el estado actual del sólido, T es la temperatura del recipiente, k_B una constante física conocida como *constante de Boltzmann*, y además:

$$Z(T)=\sum_j \exp\left(\frac{-E_j}{k_BT}\right) \qquad (1.4.2)$$

donde el índice del sumatorio j se extiende a todos los estados posibles. A la función $Z(T)$ se le llama también *función de partición.*

1.4.1.2 El Algoritmo de Metrópolis.

El proceso físico de templado, como otros métodos de física de materiales condensados, puede ser modelado adecuadamente para simularse en ordenador, Binder K. (1.978). Concretamente Metrópolis N. y otros (1.953), diseñaron un algoritmo sencillo para simular la evolución del material en el recipiente hasta llegar al

equilibrio térmico. Dicho algoritmo esta basado en técnicas de MonteCarlo, y en el se generan una secuencia de estados del material de la siguiente manera:

- sea i el actual estado del material, con energia E_i;
- el siguiente estado j con energía E_j, se genera aplicando un mecanismo de perturbación en el estado actual, por ejemplo desplazamiento de una partícula;
- si la diferencia de energía E_j-E_i es negativa o 0, el estado j se acepta como nuevo estado;
- si la diferencia de energía es positiva, el estado j se acepta con una probabilidad de:

$$P = \exp\left(\frac{E_i - E_j}{k_B T}\right) \qquad (1.4.3)$$

Este criterio de aceptación, basado en las leyes de la termodinámica, se conoce como *criterio de Metrópolis*, y el algoritmo descrito *algoritmo de Metrópolis*.

En el algoritmo de Metrópolis el equilibrio termal se alcanza generando un gran número de perturbaciones para cada valor de la temperatura.

1.4.1.3 El Temple Simulado.

Como se ha comentado anteriormente, Kirpatrick y otros (1.983) e independientemente Cerny (1.985) mostraron que el algoritmo de Metrópolis puede ser aplicado a problemas de optimización combinatoria.

Para ello, establecieron una biyección entre los elementos del proceso de enfriamiento físico, y los elementos de los problemas de optimización combinatoria, que puede observarse en la figura 1.6.

Simulación Termodinámica	**Optimización combinatoria**
Estados del material	Soluciones factibles S
Energía	Función f
Estados surgidos por mecanismo de perturbación	Soluciones vecinas
Estados metaestables	Mínimo local
Estado de congelación	Solución final

Fig. 1.6 Correspondencia entre los elementos de simulación termodinámica y optimización combinatoria.

Además se define el siguiente *criterio de aceptación*:

Definición 1.

Sea un problema de optimización combinatoria definido por el par (S,f) y sean s_0 y s dos soluciones de S. El *criterio de aceptación* determina la probabilidad con que se acepta la solución s a partir de la solución s_0; esta viene dada por la siguiente expresión:

$$P_c\{\text{aceptar } s \text{ a partir de } s_0\} = \begin{cases} 1 & \text{si } f(s) \leq f(s_0) \\ \exp\left(\dfrac{f(s_0)-f(s)}{c}\right) & \text{si } f(s) > f(s_0) \end{cases} \qquad (1.4.4)$$

donde c se define como un *parámetro de control*.-

Obsérvese que este criterio es análogo al *Criterio de Metrópolis* expuesto anteriormente (1.4.3).

De esta forma, cada método o algoritmo de búsqueda local puede transformarse en un algoritmo de templado, seleccionando aleatoriamente las soluciones vecinas y aplicando el criterio de aceptación. El esquema de un algoritmo de tipo temple simulado es el siguiente:

Algoritmo Temple_Simulado

Seleccionar una solución inicial s_0 ;

Dar un valor inicial al parámetro c ;

Repetir

niter = 0;

Repetir

niter = niter+1;

Seleccionar aleatoriamente $s \in N(s_0)$;

Si $f(s) \leq f(s_0)$ *entonces hacer* $s_0 = s$

En caso contrario

Generar aleatoriamente x uniformemente en (0,1);

Si $x < \exp\left(\frac{f(s_0) - f(s)}{c}\right)$ *entonces hacer* $s_0 = s$;

Hasta que niter = Long(c);

Disminuir el valor de c siguiendo un criterio previo;

Hasta alcanzar una condición de parada.

Long(c) es una función preestablecida que indica el número de iteraciones o perturbaciones para cada valor de *c*. Obsérvese que *c* es simplemente un parámetro de control y no tiene analogía física, ya que la constante de Boltzman se ha eliminado en la probabilidad del criterio de aceptación. Sin embargo es usual referirse a *c* como *temperatura* y la manera en la que se reduce *programa de enfriamiento*. Por otra parte el conjunto de iteraciones u operaciones realizadas en cada valor de *c* se llama *ciclo de procesamiento*.

Así mismo, también es habitual decir que el sistema *ha helado* o que se ha llegado a un *estado de congelación* cuando se alcanza el criterio de parada.

Una característica llamativa del temple simulado es que además de aceptar mejoras (descensos) en el valor de función objetivo también permite cambios que empeoran (aumentan) dicho valor. Inicialmente, para valores grandes de c, serán aceptados grandes aumentos, a medida que c decrece sólo se irán aceptando aumentos menores, y finalmente, cuando c este próximo a cero, no se aceptará ningún aumento. Esta característica permite que, a diferencia de los algoritmos de búsqueda local, puedan salir de mínimos locales y sigan siendo sencillos y de general aplicación.

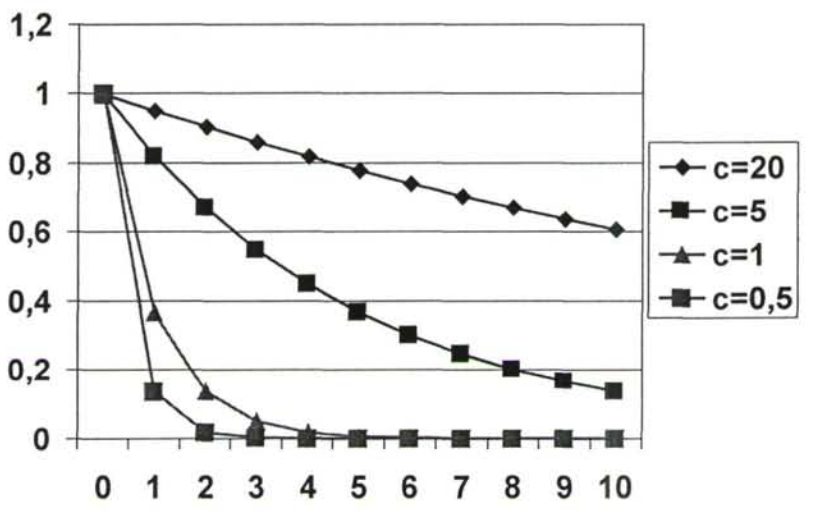


Fig. 1.7 Evoloución de exp(-d/c) en función de d.

Sea $d = f(s)-f(s_0)$, la figura 1.7 muestra la evolución de exp(-d/c) en función de d, para diferentes valores de c. Obsérvese que a medida que aumenta d disminuye la probabilidad de aceptar el cambio; por otra parte para un valor fijo de d es mas probable aceptar el cambio cuanto más alto sea c.

Los algoritmos de temple simulado se pueden considerar como una generalización de los de búsqueda local: dando a c el valor cero permanentemente, sólo se aceptarían mejoras, como en los de búsqueda local. En la analogía con los procesos físicos, la búsqueda local puede ser comparada con el proceso de *quenching* en el que, como se ha comentado anteriormente, se hace descender la temperatura bruscamente. Desde un punto de vista algorítmico en la mayoría de las aplicaciones el temple simulado llega a mejores soluciones que la búsqueda local incluso cuando esta usa diferentes soluciones iniciales.

La rapidez de la convergencia de estos algoritmos está determinada por el número de transiciones[3] o iteraciones para cada valor de c (función *long(c)*), y la forma en que se reduce este parámetro. Estos algoritmos convergen asintóticamente al óptimo (ver apéndice 6) aunque para conseguir esto se requiere un número de iteraciones excesivo. Por tanto, en la mayoría de las aplicaciones se opta por un número menor de iteraciones, que

3 Formalmente se denomina transición a la combinación de dos acciones secuencialmente: aplicación del mecanismo de elección de una solución vecina y aplicación del criterio de aceptación.

aunque no aseguran el óptimo, si dan como resultados buenas soluciones comparándolos con otros heurísticos.

1.4.1.4 Diferentes Planes de Templado.

Aunque se pueda considerar el temple simulado como un algoritmo exacto, esto sería a costa de utilizar un número de operaciones prohibitivo para gran parte de las aplicaciones. Es por ello que en la práctica se diseña en forma de heurístico utilizando un número de operaciones más asequible. Para ello se debe especificar un conjunto de parámetros y criterios que determinen la convergencia del algoritmo a la solución final. Estos parámetros y criterios constituyen el *programa de templado o enfriamiento* y son los siguientes:

- Una secuencia de valores c_k del parámetro de control, es decir:
  - Un valor inicial c_0.
  - Una función o forma de decrecimiento o descenso de dicho valor.
  - Un criterio de parada (habitualmente determinado por el valor final del parámetro c).
- Un número finito de iteraciones en cada valor de c (valor de $long(c)$).

La búsqueda de adecuados programas de templado ha sido objeto de muchos trabajos en los últimos años. Así, tenemos

artículos recopilatorios en los trabajos de Collins N.E. y otros (1.988), o también en el trabajo de Dowsland K.A. (1.996). No existe una norma general que determine que plan es mejor, dependiendo esto del problema concreto. Es a través de la experimentación como se determina el plan adecuado para cada caso. A continuación se describen las formas más habituales de establecer los parámetros y criterios que constituyen el programa de templado.

1.4.1.5 Parámetro Inicial c_0 (temperatura inicial).

El valor de c_0 debe ser lo suficientemente alto para permitir virtualmente que todos los cambios sean aceptados. En analogía con los procesos físicos esto corresponde a elevar la temperatura del material hasta la fase líquida en la que todas las partículas se mueven libremente. Obsérvese que de esta forma se asegura la independencia de la solución inicial.

Para ello en algunos casos, por ejemplo en Osman I.H. (1.993) y (1.995), se calculan los valores mayor y menor de f en el conjunto de soluciones vecinas de la solución inicial ($fmax$ y $fmin$ respectivamente) y se toma c_0 como la diferencia entre estos valores:

$$c_0 = fmax - fmin$$

En otros casos, por ejemplo en Aarts E.H.L. y Korst J.H.M. (1.990), se realiza el siguiente proceso:

- comenzar con un valor pequeño de c_0.
- realizar un conjunto de iteraciones y determinar que proporción de cambios son aceptados (*radio de aceptación*).
- multiplicar por una constante mayor que uno a c_0.
- repetir los dos últimos pasos hasta que el *radio de aceptación* sea lo suficientemente alto.

1.4.1.6 Función de Decrecimiento (velocidad de enfriamiento).

La forma en que c decrece es muy importante en la calidad de la solución final obtenida. La forma más comúnmente usada de *enfriamiento* o reducción de estos valores es considerar una velocidad geométrica de decrecimiento:

$$c_{k+1} = \alpha\, c_k$$

habitualmente con $\alpha \in [0'8, 0'99]$ (enfriamiento lento, Kuik R. y Salomon M. (1.990)). Valores altos de α corresponden a enfriamientos más lentos y dan lugar a mejores soluciones.

Otra forma de reducción usada con cierta frecuencia es la propuesta por Lundy M. y Mees A. (1.986); realiza una sola iteración para cada temperatura:

$$c_{k+1} = \frac{c_k}{1 + \beta \cdot c_k}$$

donde β es un parámetro suficientemente pequeño ($\beta \in (0'1, 0'3)$).

Dowsland K.A. (1.993), sugiere un proceso más gradual que el anterior, basado en el hecho de que a medida de que desciende la temperatura es más difícil salir de un mínimo local, por tanto sugiere la posibilidad de aumentos de temperatura cuando la búsqueda según el enfriamiento seguido parece infructuosa; más concretamente:

- cada vez que un movimiento es aceptado se reduce el valor de c de la siguiente forma:

$$c_{k+1} = \frac{c_k}{1 + \beta \cdot c_k}$$

- y cada vez que se rechaza aumentarlo de la forma:

$$c_{k+1} = \frac{c_k}{1 - \gamma \cdot c_k}$$

donde, se toma $\beta = \gamma \cdot r$. De esta modo harán falta r calentamientos para *compensar* un enfriamiento. Esta programación ha sido aplicada con éxito a problemas de empaquetamiento.

En cualquier caso todos los expertos señalan que no es tan importante la función elegida para el descenso de los valores de c, sino que dicho descenso sea lo suficientemente suave.

1.4.1.7 Criterio de Parada.

En teoría, el proceso ha de finalizar cuando el valor de *c* llegue a 0 (sistema frío). Sin embargo, en la práctica no es necesario llegar a este punto, ya que bastante antes de llegar a ese valor es prácticamente nula la probabilidad de que se acepte un movimiento hacia una solución peor:

- En algunos casos el valor mínimo de *c* está determinado por la precisión del compilador donde se ejecute el algoritmo.

- En otros casos se establece el criterio de parada cuando *c* alcanza valores que hagan insignificante la probabilidad de cambios hacia arriba en el valor de la función *f*.

- Otros autores suavizan esta condición y establecen el criterio de parada cuando han transcurrido un determinado número de iteraciones sin cambio alguno: concretamente Osman I.H. (1.993), define la *congelación* cuando ha transcurrido sin cambios un ciclo de procesamiento.

La experiencia indica que en ocasiones, antes de que se den estas situaciones, el algoritmo puede gastar mucho tiempo de computación (varias iteraciones o incluso ciclos de procesamiento) "circulando" o moviéndose en un conjunto de soluciones cercanas, entorno a un mínimo local. Al producirse cambios, no hay *congelación*, sin embargo no se consigue salir del valle

correspondiente. En la gráfica de la figura 1.8 se ilustra esta situación.

Para evitar este inconveniente en ocasiones se establece un criterio de parada más suave: el algoritmo finaliza cuando el número de movimientos aceptados baja de una determinada proporción.

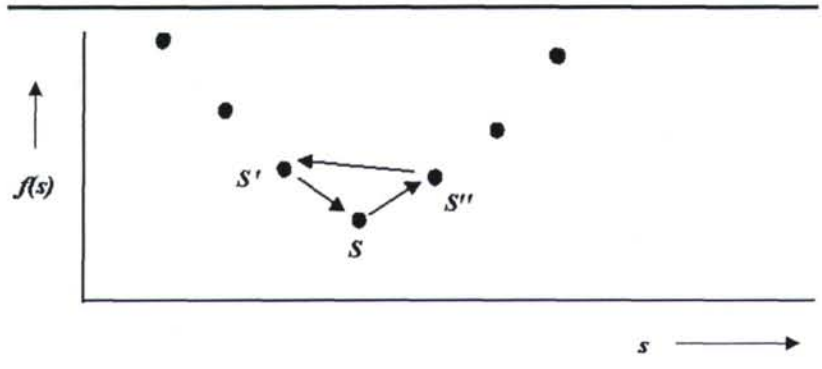


Fig. 1.8 La solución actual S sólo se mueve en el mínimo local S, y sus vecinas S' y S''.

Otra solución alternativa propuesta por Pacheco J. (1.997-b) y Pacheco J. y Delgado C. (1.997-a), consiste en *identificar* o hallar, cada cierto número de iteraciones el mínimo local correspondiente al valle donde se encuentra la solución actual; el algoritmo se para cuando se han realizado un número prefijado de exploraciones sin que cambie este mínimo, en otras palabras sin salir del mismo valle.

Algunos autores como Osman I.H. (1.993) y (1.995), sugieren la posibilidad de *recalentamiento* una vez que el sistema ha helado, y volver a repetir el proceso. El algoritmo finaliza cuando han transcurrido un número determinado de recalentamientos sin cambiar la mejor solución obtenida hasta ese momento.

1.4.1.8 Longitud de los Ciclos de Procesamiento.

El número de iteraciones de cada ciclo de procesamiento esta muy estrechamente relacionado con la forma de *enfriamiento*: si este es lento, la longitud de los ciclos no debe ser alta para evitar que el tiempo de computación sea excesivo; si el enfriamiento es rápido se puede utilizar mas iteraciones en cada valor de c.

En cualquier caso, está admitido de forma casi generalizada, que el valor de *long(c)* debe crecer a medida que decrezca c, ya que como se ha comentado, según disminuye c es más difícil salir de mínimos locales (se hacen necesarios más intentos para aceptar cambios hacia arriba).

Aarts E.H.L. y Korst J.H.M. (1.990) sugieren realizar en cada ciclo de procesamiento, tantas iteraciones como sean necesarias para que se acepten un número predeterminado de cambios. Estos autores, afirman que *"se puede aceptar la idea intuitiva de que se alcanzan distribuciones más cercanas a la estacionaria a medida que se acepten más cambios"*. Con este criterio se tiene que:

$$\text{Si } c \downarrow 0 \Rightarrow long(c) \rightarrow \infty \ ;$$

en este caso se hace necesario establecer una cota superior L a la longitud de los ciclos de procesamiento.

1.4.1.9 Propuesta de un Híbrido.

Como se ha comentado, uno de los problemas de la aplicación de los procesos de Temple Simulado es que, en general, no es fácil identificar cuando el proceso se ha estabilizado, ya que la solución actual puede estar moviéndose en torno a un mínimo local y soluciones vecinas.

En la descripción de un plan de templado se explicaba una idea propia para identificar estas situaciones. Sin embargo, la experiencia muestra que este método no consigue evitar movimientos entre soluciones no vecinas pero cercanas, como se ilustra en la figura 1.9.

En este apartado se propone aprovechar la estrategia tabú, descrita en el apartado siguiente, 1.4.2, para evitar ciclos una vez que el proceso de Temple Simulado baja de una cierta temperatura, en la que se empiezan a ser difíciles movimientos hacia arriba significativos.

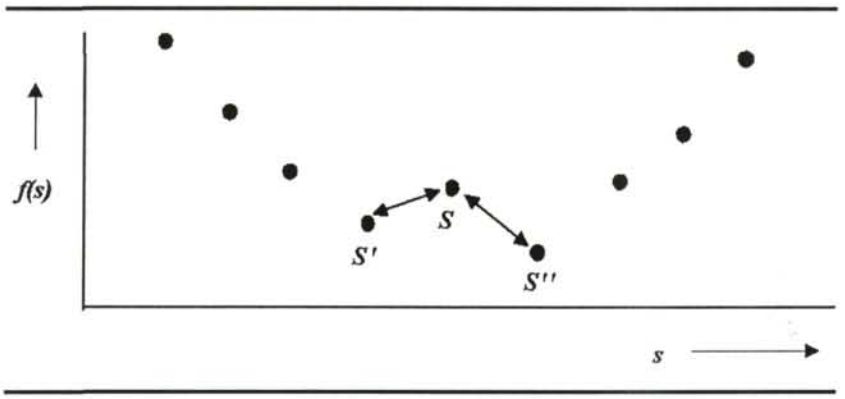


Fig. 1.9 Un ciclo puede suponer movimientos entre soluciones de valles diferentes.

El algoritmo finaliza cuando transcurren, desde que se incorpora la estrategia tabú, un determinado número de iteraciones sin cambio en el mejor valor de la función objetivo (almacenado en la variable *min1*). La primera solución a tener en cuenta es la que se esté evaluando en el momento en que se incorpora la estrategia tabú. El algoritmo híbrido propuesto es el siguiente:

Algoritmo Hibrido_Temple_Simulado_Búsqueda_Tabu

Seleccionar una solución inicial s_0 ;

Dar un valor inicial al parámetro c ; Hacer $c_0 = c$

Hacer $s^* = s_0$; $T = \varnothing$, *niter = 0, kiter = inf, min1:= inf*

Repetir

niter = niter+1;

Seleccionar aleatoriamente $s \in N(s_0)$

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

Si $f(s) \geq f(s^*)$ *y* $s \notin T$ *entonces*

Si $f(s) \leq f(s_0)$ *entonces hacer* $s_0 = s$

En caso contrario entonces

Generar aleatoriamente x uniformemente en (0,1);

Si $x < exp((f(s_0)-f(s))/c)$ *entonces hacer* $s_0 = s$;

Si $c < c_0/10$ *entonces*

Actualizar T

Si $f(s) < min1$ *entonces*

Hacer $min1 = f(s)$

kiter = niter

Hacer $c = \alpha \cdot c$

Hasta (niter - kiter > maxiter)

Las variables que determinan cuando se cumple el criterio de parada son *min1* y *kiter*; *T* es la lista de movimientos tabús.

De esta forma se intenta conseguir:

a) Evitar ciclos que impidan reconocer una congelación, es decir, que hagan gastar tiempo de computación.
b) "Forzar" al proceso a tomar o explorar caminos alternativos.

Para el VRPTW Mixto, en Pacheco J. y Delgado C.R. (1.998), siendo los resultados idénticos, se muestra como esta idea puede ahorrar un 50% de iteraciones frente a Temple Simulado con criterios de parada descritos anteriormente, una vez que se empieza a chequear la congelación ($c < c_0/10$).

1.4.1.10 Aplicaciones del SA a Problemas de Rutas.

Dos de las primeras implementaciones de Temple Simulado para el VRP son las de Robusté F. y otros (1.990) y Alfa A. y otros (1.991). En la primera, los autores definen una estructura de vecindario mediante la combinación de varios mecanismos: inversión de parte de una ruta, traslado de parte de una ruta a otra parte de esa misma ruta, intercambio de vértices entre dos rutas. Se probó el algoritmo con cuatro tamaños diferentes de problemas (80, 100, 120 y 500 puntos de visita) pero no fue comparado con otras alternativas disponibles.

En Alfa A. y otros (1.991), se usó un heurístico basado en el método *ruta primero, cluster segundo* (se crea una ruta que incluye todos los puntos de visita del problema y luego se divide en rutas más pequeñas para conseguir una solución factible, Beasley J.E. (1.983)). Construida la solución inicial se mejora mediante un intercambio 3-optimo (Lin S. (1.965); se verá con detalle en el capítulo 2). El método se aplicó a tres tamaños de problemas (30, 50 y 75 puntos de visita) sin producir resultados competitivos.

La implementación de Osman I.H. (1.993), posterior a las dos anteriores, es más compleja y más exitosa. Usa mejores soluciones iniciales, ajusta algunos parámetros en una fase de prueba, explora vecindarios más "ricos" y el programa de enfriamiento es más refinado. La estructura de vecindario de este algoritmo usa un

mecanismo de generación de nuevas soluciones denominado λ-*intercambio*: se seleccionan dos rutas p y q, y dos subconjuntos de puntos de visita S_p y S_q, uno de cada ruta, en número menor que λ; se intercambian los puntos de visita de S_p con los de S_q siempre y cuando sea posible; el conjunto S_p o S_q puede ser vacío, la operación en este caso sería de cambio de puntos de visita de una ruta a otra. Dado que el número de combinaciones de pares de ruta y elecciones de S_p y S_q es en general elevado, este procedimientos se implementa con $\lambda = 1$ o 2 y en las versiones más eficientes del algoritmo, la búsqueda se detiene cuando se identifica un movimiento que mejora la solución (primer descenso); cuando esto no sucede, se debe explorar todo el vecindario. Otra versión del algoritmo consiste precisamente, en examinar todo el vecindario y elegir el mejor movimiento (mejor descenso). Se contrastaron los resultados de la implementación del algoritmo, utilizando la estrategia de primer descenso, para el VRP simétrico: en varias ocasiones se alcanzó el mejor valor conocido.

Van Breedam A. (1.995) ha probado y comparado Temple Simulado con diferentes estructuras de vecindarios. Los test se realizaron con 14 instancias de Christofides N. y otros (1.979). Los experimentos fueron útiles para ayudar a identificar la mejor estrategia de Temple Simulado, pero todos ellos confirmaron la superioridad de los heurísticos basados en Búsqueda Tabú.

Otros trabajos referenciados en la literatura son los siguientes:

- Hiquebran D. y otros (1.994);
- Janssens G. K. y Breedam A. V. (1.995);
- Teodorovich D. y Pavkovic G. (1.992) y
- Van Breedam A. (1.996).

1.4.2 Búsqueda Tabú.

Según Glover F. (1.996) la palabra Tabú se refiere a: "*..un tipo de inhibición por connotaciones culturales o históricas que puede ser superado en determinadas condiciones...*".

Búsqueda Tabú (*Tabú Search, TS*) dada a conocer por Glover F. (1.989) y (1.990-a), es un procedimiento metaheurístico utilizado con el fin de guiar un algoritmo heurístico de búsqueda local para explorar el espacio de soluciones más allá de la simple optimalidad local y obtener soluciones cercanas al óptimo. Se han publicado numerosos artículos y libros para difundir el conocimiento teórico del procedimiento; en Glover F. y Laguna M. (1.997) y (1.999) pueden encontrarse recientes y amplios tutoriales sobre Búsqueda Tabú que incluyen todo tipo de aplicaciones.

Los puntos que se desarrollan en este apartado son los siguientes:

1. Búsqueda Tabú. Fundamentos.
2. Memoria a Corto Plazo.
3. Memoria a Medio y Largo Plazo.
   3.1 Intensificación.
   3.2 Reencadenamiento de Trayectorias o Path Relinking.
   3.3 Diversificación.
   3.4 Oscilación Estratégica.
4. Aplicaciones de Búsqueda Tabú a Problemas de Rutas.

1.4.2.1 Búsqueda Tabú. Fundamentos.

Al igual que la búsqueda local, la búsqueda tabú en su diseño básico, constituye una forma agresiva de búsqueda del mejor de los movimientos posibles a cada paso; sin embargo también permite movimientos hacia soluciones del entorno aunque no sean tan buenas como la actual, de forma que se pueda escapar de óptimos locales y continuar la búsqueda de soluciones aún mejores (figura 1.10). Simultáneamente para evitar ciclos, los últimos movimientos realizados son declarados tabú durante un determinado número de iteraciones, utilizando las denominadas *restricciones tabú*. Dicha condición tabú puede ser ignorada bajo determinadas circunstancias dando lugar a los llamados *criterios de aspiración*. De esta forma se introduce cierta flexibilidad en la búsqueda.

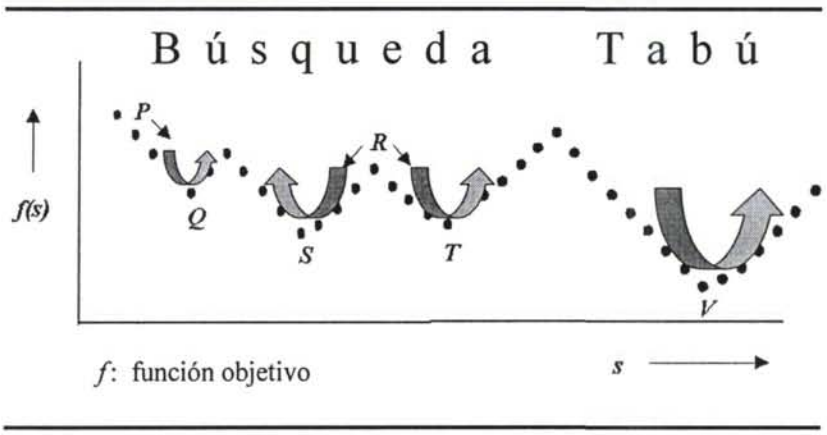


Fig. 1.10 Procedimiento de Búsqueda Tabú.

Búsqueda Tabú rebasa la búsqueda local empleando una estrategia de modificación del entorno $N(s)$ a medida que la búsqueda progresa, reemplazándolo por otro entorno $N^*(s) = N(s)-T$.

El procedimiento de búsqueda tabú básico puede ser formulado de la forma siguiente:

Procedimiento Búsqueda_Tabú_Básico

Leer una solución inicial s_0, $s^*=s_0$

Inicializar $T = \varnothing$

Repetir

Determinar $N^*(s_0) \subset N(s_0)$

Elegir $f(s') = min \{f(s) / s \in N^*(s_0)\}$

Si $f(s') < f(s^*)$ *entonces* $s^*=s'$

Hacer $s_0 = s'$

Actualizar T

Hasta alcanzar una condición de parada

T es el conjunto de *soluciones tabú* (últimas soluciones visitadas y según el tipo de implementación, también soluciones que tienen determinados atributos de esas últimas soluciones visitadas). $N^*(s)$ es el conjunto de soluciones vecinas que no son tabú o que satisfacen el criterio de aspiración. Habitualmente, una solución cumple el criterio de aspiración si es mejor que la mejor solución encontrada hasta ese momento (s^*). Las soluciones que son admitidas en $N^*(s)$ se determinan de varias formas; una de ellas, que da nombre a la búsqueda tabú, identifica soluciones visitadas en un pasado cercano (e implícitamente algunas soluciones identificadas con ellas) y les prohibe pertenecer a $N^*(s)$ clasificándolas como tabú.

TS es flexible y motiva a crear nuevos tipos de movimientos y criterios de evaluación según los diferentes problemas. En problemas de tamaño reducido consigue soluciones óptimas o dentro de una banda aceptable utilizando un tiempo de computación pequeño. Para problemas de mayor tamaño o más difíciles las soluciones compiten, y a veces superan a las mejores soluciones previamente encontradas. En otros casos TS ha demostrado ventajas facilitando la implementación o la capacidad de manejar consideraciones adicionales.

TS toma de la Inteligencia Artificial el concepto de memoria y lo implementa mediante estructuras simples, con el objetivo de dirigir la búsqueda teniendo en cuenta la historia de ésta. Es decir,

el procedimiento trata de extraer información de lo sucedido y actuar en consecuencia.

Las estructuras de memoria diseñadas, permiten utilizar criterios de evaluación e información de la búsqueda, que determinan cuáles son las soluciones tabú y cuales de ellas las que satisfacen el criterio de aspiración, y en consecuencia $N^*(s)$. TS incorpora estructuras de memoria con diferente plazo de tiempo, desde corto a largo plazo, para reforzar la búsqueda en entornos de buenas soluciones (*intensificación*) o explorar espacios de soluciones no visitados anteriormente (*diversificación*).

Se construyen estructuras de memoria teniendo en cuenta las cuatro principales dimensiones de la memoria: *recencia*, *frecuencia*, *calidad* e *influencia*. Las dos primeras se complementan entre sí y se emplean en estructuras de memoria a corto y largo plazo respectivamente. La calidad se refiere a la capacidad de diferenciar la bondad (en términos del valor de la función objetivo) de las soluciones visitadas durante la búsqueda. La memoria se puede usar para identificar los elementos que son comunes en las buenas soluciones o los caminos que permiten llegar a tales soluciones. La cuarta dimensión considera el impacto de las elecciones hechas durante la búsqueda, no sólo en lo que se refiere a la calidad de las soluciones sino también en lo que se refiere a su estructura. Puede ser muy útil el encontrar o diseñar otros evaluadores que guíen la búsqueda diferentes al de la

evaluación de la función objetivo. Su utilidad principal es la determinación de estructuras subyacentes en las soluciones. Esto permite que sean la base para procesos de Intensificación y Diversificación.

La búsqueda tabú se basa en la premisa de que para poder calificar de inteligente la estrategia de búsqueda de solución de un problema, debe incorporar *memoria adaptativa* y *exploración sensible*. El uso de memoria adaptativa contrasta con diseños desmemoriados, tales como los que se inspiran en metáforas de la física y la biología (Temple Simulado y otros enfoques aleatorizados) y con diseños de memoria rígida como los empleados en Branch-and-Bound y en Inteligencia Artificial. El uso de memoria puede ser explícito (se almacenan soluciones completas) o a través de atributos (atributos de soluciones que se modifican después de un movimiento). El énfasis en la exploración sensible se deriva de la suposición de que una mala elección estratégica puede producir más información que una buena elección al azar.

1.4.2.2 Memoria a Corto Plazo.

Los métodos basados en búsqueda local requieren de la exploración de un gran número de soluciones en poco tiempo; por ello es importante reducir al mínimo, el esfuerzo computacional de las operaciones que se realizan a menudo. En este sentido, la

memoria a corto plazo de TS está basada en atributos, en lugar de ser explícita; esto es, en lugar de almacenar las soluciones completas, como ocurre en los procedimientos enumerativos de búsqueda exhaustiva, se almacenan únicamente algunas características de éstas.

El uso de memoria a Corto Plazo en Búsqueda Tabú constituye un modo agresivo de exploración, que busca realizar el mejor movimiento posible, sujeto a las elecciones disponibles requeridas para satisfacer determinadas obligaciones, y que, como se ha comentado con anterioridad, vienen expresadas en las denominadas *restricciones tabú*. Su primer objetivo es permitir ir más allá de los óptimos locales, sin dejar de realizar movimientos de alta calidad en cada paso.

Las restricciones tabú se diseñan para preveer las repeticiones de ciertos *movimientos tabú*, por repetición de atributos seleccionados de esos movimientos tabú, evitando comportamientos cíclicos e induciendo a la búsqueda a seguir una nueva trayectoria. Los atributos seleccionados que se presentan en soluciones recientemente visitadas (memoria basada en recencia de eventos) son designados como *tabú-activos*, y las soluciones que contienen elementos tabú-activos o combinaciones particulares de estos atributos son las que se convierten en *soluciones tabú*, y son prohibidas durante un periodo de tiempo. Se considera que tras un cierto número de iteraciones, la búsqueda está en una región

distinta y puede liberarse del status tabú a las soluciones antiguas. Se denomina *tenencia tabú* (*tabu tenure*) a la duración que un atributo permanece tabú-activo. La tenencia tabú puede ser fija o variar para diferentes tipos o combinaciones de atributos, y para diferentes intervalos de tiempo o etapas de la búsqueda. Esta tenencia variable permite la creación de diferentes tipos de ajuste entre estrategias de corto y largo plazo. Esto produce también una forma dinámica y robusta de búsqueda.

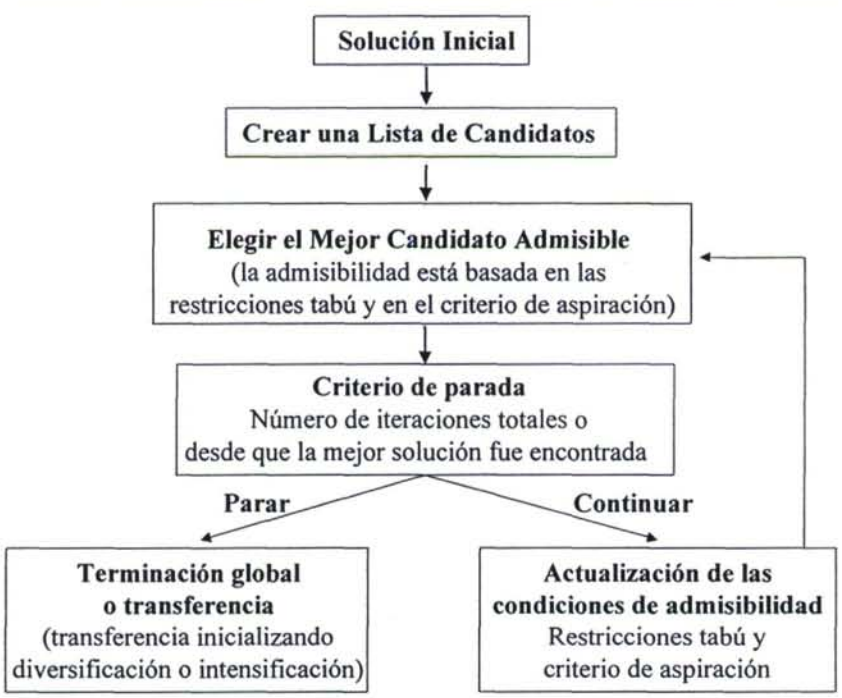


Fig 1.11 Componentes de memoria a corto plazo (tomado de Glover (1.990-b)).

Las restricciones tabú no operan de manera aislada ya que son contrapesadas por la aplicación del *criterio de aspiración*. Dicho criterio introduce un elemento importante de flexibilidad: el estatus tabú puede ser ignorado si se cumplen determinadas condiciones en la forma de *niveles de aspiración*; por ejemplo, cualquier solución que sea mejor que cualquiera de las encontradas anteriormente merece ser considerada admisible, incluso aunque para alcanzarla debamos utilizar un movimiento prohibido.

En ocasiones se utilizan estrategias de *listas de candidatos* para restringir el número de soluciones examinadas en una iteración dada o para mantener un carácter agresivo en la búsqueda. En la *elección del mejor candidato admisible*, se evalúa cada uno de los movimientos de la lista de candidatos; en algunos casos la evaluación de un movimiento puede basarse inicialmente en el cambio de valor producido en la función objetivo; en otros casos donde los movimientos son menos fáciles de evaluar, pueden utilizarse medidas de aproximación.

Dado que el número de movimientos clasificados tabú será generalmente pequeño en relación al número total de movimientos disponibles y asumiendo que el gasto de evaluación de movimientos no es grande, es normalmente preferible chequear primero si un movimiento dado tiene una evaluación más alta que sus predecesores admisibles antes que chequear su estatus tabú. Chequear el estatus tabú es el primer paso para investigar la

admisibilidad. Si el movimiento no es tabú es aceptado inmediatamente, en caso contrario el criterio de aspiración da una segunda oportunidad para calificar dicho movimiento como admisible. Es usual que la fase de memoria a corto plazo termine tras un número fijo de iteraciones sin mejora.

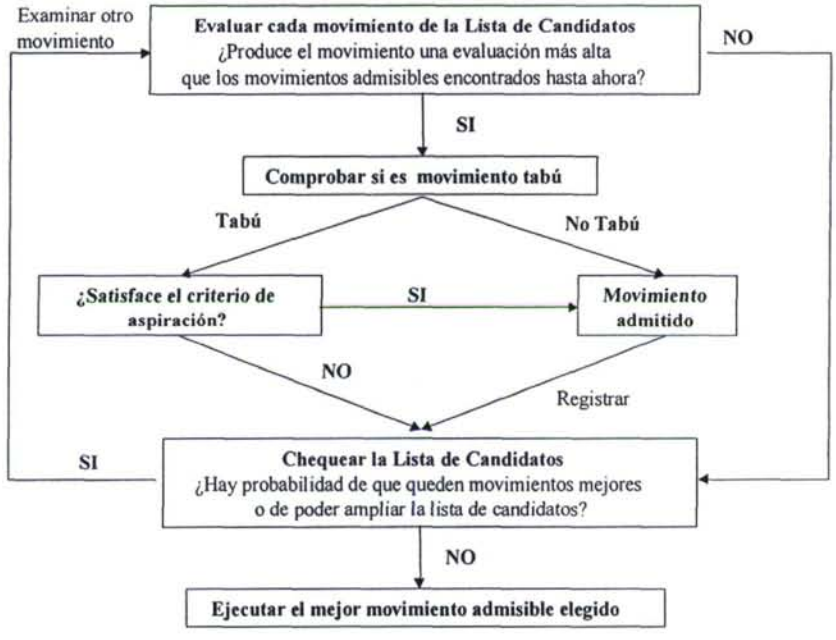


Fig. 1.12 Proceso Tabú de evaluación. Selección del mejor candidato admisible (tomado de Glover (1.990-b)).

Normalmente TS se basa en reglas sistemáticas en lugar de decisiones al azar. Sin embargo, en ocasiones se recomienda el aleatorizar algunos procesos (elecciones probabilísticas), para facilitar la elección de buenos candidatos, o cuando no está clara

la estrategia a seguir (quizá por tener criterios de selección enfrentados). La selección aleatoria puede ser uniforme o seguir una distribución de probabilidad construida empíricamente a partir de la evaluación asociada a cada movimiento.

Con los elementos descritos puede diseñarse un algoritmo básico de TS para un problema de optimización dado. Sin embargo, TS ofrece muchos más elementos para construir algoritmos realmente potentes y eficaces. A menudo, dichos elementos han sido ignorados en muchas aplicaciones y actualmente la introducción de estos en la comunidad científica constituye un reto para los investigadores del área.

1.4.2.3 Memoria a Medio y Largo plazo.

En muchos casos la memoria a corto plazo proporciona por si sola soluciones mejores que las encontradas por procedimientos alternativos, y el uso de memoria a largo plazo en dichos casos se ha evitado. Pero la memoria a largo plazo puede ser importante para obtener mejores resultados para problemas complejos. La idea es aplicar memoria a corto plazo y cuando se desee un mayor refinamiento introducir memoria a largo plazo.

La memoria a largo plazo almacena las frecuencias u ocurrencias de atributos en las soluciones visitadas, tratando de identificar o diferenciar regiones. Funciona principalmente como

base de estrategias para intensificar y diversificar la búsqueda, aunque los elementos fundamentales de estrategias de intensificación y diversificación ya se presentan en el componente de memoria a corto plazo de Búsqueda Tabú: las listas tabú de memoria a corto plazo tienen temporalmente un papel intensificador buscando determinados atributos atractivos y un papel diversificador obligando a nuevas elecciones para introducir (o excluir) atributos que no están entre aquellos recientemente descartados (o incorporados).

Para problemas sencillos hay variedad de fórmulas para escapar de las trampas que impiden el acceso a soluciones óptimas (por ejemplo empleando funciones de penalización lagrangianas o considerando vecindarios más amplios). Para problemas complejos, una respuesta instintiva podría ser volver a la aleatoriedad, intentando descubrir un movimiento efectivo por medio de una operación de "buena fortuna". Dicha aproximación, aunque posible, no es una aproximación basada en el uso de memoria, con lo que se estaría siguiendo una elección ciega para un objetivo de diversificación.

El uso de memoria a largo plazo no requiere necesariamente recorrer muchas soluciones antes de que sus beneficios se hagan visibles. A menudo sus mejoras comienzan a manifestarse en relativamente poco tiempo. Los métodos más rápidos para algunos tipos de problemas de rutas y programación de tareas por ejemplo,

se basan en la inclusión de memoria de búsqueda tabú a largo plazo. Por otro lado es también cierto que la oportunidad de encontrar buenas soluciones cuando el tiempo crece, en el caso de que la solución óptima no se haya encontrado aún, se mejora usando memoria a largo plazo además de la memoria a corto plazo.

1.4.2.3.1 Intensificación.

La intensificación consiste en regresar a regiones "atractivas" ya exploradas para estudiarlas más a fondo. Las estrategias de intensificación se basan en la modificación de reglas de elección, para incrementar las combinaciones de movimientos y las características de las soluciones, que históricamente han demostrado ser buenas. En la fase de intensificación se examinan las que hasta el momento son las mejores soluciones; se extrae de ellas la información relevante necesaria en cada caso, con el fin de realizar movimientos hacia soluciones con características similares, que den igualmente buenos resultados y así poder encontrar entre ellas la óptima. Un ejemplo sencillo de intensificación sería el siguiente:

Procedimiento Intensificación

Aplicar memoria a corto plazo

Aplicar una estrategia de selección de soluciones élite

Repetir
Elegir una de las soluciones élite
Reanudar memoria corto plazo desde las soluciones elegidas
Añadir nuevas soluciones a la lista de soluciones élite
cuando sea pertinente
Hasta que la lista de soluciones élite esté vacía

Dos variantes han demostrado suficiente éxito ofreciendo soluciones de alta calidad. La primera introduce una *medida de diversificación* para asegurar que las soluciones registradas difieran, una de cada una de las otras, en un grado deseado, y borra toda memoria de corto plazo antes de reanudar el proceso desde la mejor de las soluciones registradas (Voss S. (1.993)). La medida de diversificación puede por ejemplo relacionarse con el número de movimientos que son necesarios para transformar una solución en otra.

La otra variante mantiene una lista de soluciones de longitud limitada y añade al final una nueva solución, sólo si es mejor que cualquier otra previamente visitada (Nowicki E. y Smutnicki C. (1993)). El último miembro de la lista en cada momento es el escogido y suprimido para reanudar la búsqueda. La memoria a corto plazo que acompaña esta solución se salva; el movimiento previamente tomado de esta solución es inhibido iniciando así un nuevo camino de búsqueda. A este enfoque de recuperar soluciones élite seleccionadas se le denomina *desandar* (*backtracking*).

Una tercera variante, relacionada con la anterior, consiste en reanudar la búsqueda desde *entornos no visitados*, previamente generados (Glover F. (1.990-a)). Esta estrategia explora vecindarios de óptimos locales o vecindarios de soluciones visitadas en pasos inmediatamente antes de alcanzar tales óptimos locales. Este tipo de estrategia de vecindarios no visitados ha sido poco examinada.

Otro enfoque es el de *intensificación por descomposición*; en este caso se imponen restricciones a partes del problema o a la estructura de solución para generar una forma de descomposición que permita un enfoque más concentrado en otras partes de la estructura. Para ilustrar este tipo de intensificación podemos utilizar el TSP; los arcos que pertenecen a todas las rutas élite se pueden "fijar" en la solución, para concentrarse en la manipulación de otras partes de la ruta. El uso de intersecciones es un ejemplo extremo de una estrategia más general para explotar la información de frecuencia, mediante un proceso que busca identificar y restringir los valores de *variables fuertemente determinadas y consistentes.*

La intensificación por descomposición también abarca otros tipos de consideraciones estratégicas, que basan la descomposición no sólo en indicadores de fuerza y consistencia, sino también en oportunidades de determinados elementos para influirse mutuamente de forma productiva. Dentro de un contexto de

problemas de permutación como los de rutas o los de programación de tareas por ejemplo, donde las soluciones pueden representarse como selecciones de una o más secuencias de nodos en un grafo, una descomposición puede basarse en identificar subcadenas de soluciones élite, donde dos o más subcadenas pueden asignarse a un conjunto común si contiene nodos que son "fuertemente atraídos" para unirse con nodos de otras subcadenas del conjunto. Una colección de nodos inconexos de subcadenas, pueden tratarse mediante un proceso de intensificación que opera en paralelo en cada conjunto, sujeto a la restricción de que la identidad de los puntos extremos de las subcadenas no se altere. Como resultado de la descomposición, el mejor conjunto nuevo de subcadenas puede volver a unirse para crear nuevas soluciones. Los movimientos por influencia son aquellos que producen un cambio importante en la estructura de las soluciones. Usualmente, en un procedimiento de búsqueda local, la búsqueda es dirigida mediante la evaluación de la función objetivo. Sin embargo, puede ser muy útil el encontrar o diseñar otros evaluadores que guíen a ésta en determinadas ocasiones. Los movimientos de influencia proporcionan una evaluación alternativa de la bondad de los movimientos, al margen de la función objetivo. Su utilidad principal es la determinación de estructuras subyacentes en las soluciones. Esto permite que sean la base para procesos de Intensificación y Diversificación a largo plazo.

En el trabajo de Díaz J.A. y Fernández E. (1.998-b), los autores utilizan la memoria basada en frecuencia; en la fase de intensificación asignan trabajos a agentes, siempre que en las mejores soluciones la frecuencia de dicha asignación fuera al menos del 85%. Fijar algunas asignaciones es equivalente a reducir el tamaño del problema. Las asignaciones realizadas las consideran buenas ya que se han presentado en las mejores soluciones generadas hasta el momento, y si aparecen en la mayoría de las soluciones quiere decir también que han sido seleccionadas muchas veces o que después de seleccionadas no han sido cambiadas durante mucho tiempo. Posteriormente se restaura la fase de memoria a corto plazo para el problema resultante.

El proceso a seguir en cada caso para intensificar dependerá del problema concreto al que se aplique la búsqueda Tabú

1.4.2.3.2 Reencadenamiento de Trayectorias o Path Relinking.

El reencadenamiento de trayectorias es un tipo de intensificación que está tomando gran importancia y que merece un tratamiento en un apartado independiente. La utilización del *Paht Relinking* se basa en la idea de que entre dos soluciones élite pueden existir soluciones de calidad que deben ser estudiadas.

El proceso se inicia seleccionando dos soluciones s' y s'' de una colección de soluciones élite (óptimos locales de alta calidad producidos en fases de búsqueda previas). Se genera entonces un "camino" desde s' a s'', produciéndose una secuencia de soluciones $s' = s(1), s(2), ..., s(r) = s''$.

Se usa una estructura de vecindario estándar para definir los movimientos disponibles que permiten ir de una solución a la siguiente, desde $s(i)$ hasta $s(i+1)$, de forma que el número de movimientos restantes para llegar a s'' sea el menor posible. Se trata de buscar la ruta más directa entre ambas soluciones. Existe una gran variedad de caminos posibles que permiten alcanzar s'' desde s'.

Para seleccionar los movimientos no se considera la función objetivo o el criterio que se haya estado utilizando sino que se incorporan a s' los atributos de s'' hasta llegar a ésta. En algunas implementaciones se ha considerado el explorar el entorno de las soluciones intermedias para dar más posibilidad al descubrimiento de buenas soluciones.

Una variante de *Paht Relinking* comienza en los extremos s' y s'' de forma simultánea, generando dos secuencias $s' = s'(1), ..., s'(r)$ y $s'' = s''(1), ..., s''(t)$. Las elecciones se diseñan para producir $s'(r) = s''(t)$. Cuando esto no sucede, se selecciona $s'(r)$ para crear $s'(r+1)$ mediante el criterio de minimizar el numero de

movimientos sobrantes para alcanzar $s''(t)$ o bien se elige $s''(t)$ para crear $s''(t+1)$ mediante el criterio de minimizar el número de movimientos sobrantes para alcanzar $s'(r)$.

Paht Relinking puede ser mejorado mediante el uso de una estrategia denominada *tunneling*, que permite el uso de estructuras de vencindarios diferentes a las de la fase de búsqueda estandar. En particular, puede ser atractivo permitir de forma periódica, movimientos para *Paht Relinking* que normalmente se excluirían por crear infactibilidad. Existe la seguridad de no perderse en regiones infactibles, ya que hemos de alcanzar s'', por lo que en algún momento se recuperará la factibilidad. Esta estrategia nos permite visitar soluciones que de otro modo podrían no ser visitadas.

La elección de s' y s'' de un conjunto de soluciones élite agrupado estimula la intensificación; por el contrario si se eligen de grupos ampliamente separados se estará estimulando la diversificación.

El *Path Relinking* se extiende por supuesto más allá de la consideración de los puntos entre s' y s'', de la misma forma que la combinación lineal se extiende más allá de los puntos que se expresan como combinación convexa de dos puntos extremos. Para simplificar, supóngase que comenzamos con s' y buscamos un camino que continúe más allá de s''. La habilidad para ir más allá

de ese punto extremo resulta mediante un método de aproximación del criterio de elección de un movimiento que deja el menor número de movimientos restantes hasta alcanzar *s''*.

1.4.2.3.3 Diversificación.

Las estrategias de diversificación conducen la búsqueda hacia nuevas regiones aún no visitadas. Es usual modificar las reglas de elección para incorporar en la solución atributos que no hayan sido usados frecuentemente.

Un enfoque que parece tener mucho éxito para problemas de *scheduling* (programación de tareas) es el que mantiene la memoria basada en frecuencia sobre todas las soluciones previamente generadas. El esquema es el que se muestra en la figura 1.14.

Se han logrado mejoras significativas en la aplicación de memoria a corto plazo mediante este procedimiento, véase Glover F. y Laguna M. (1.993).

También ha sido utilizado este enfoque en el trabajo de Díaz J.A. y Fernández E. (1.998-b) con buenos resultados. Se utiliza la memoria basada en frecuencia pero de diferente modo que en la estrategia de intensificación. Cuanto mayor sea la frecuencia más se penaliza una asignación; de esta forma se producen soluciones con diferentes estructuras a las generadas anteriormente. Los

resultados obtenidos son de alta calidad y normalmente se obtienen tras un número reducido de iteraciones.

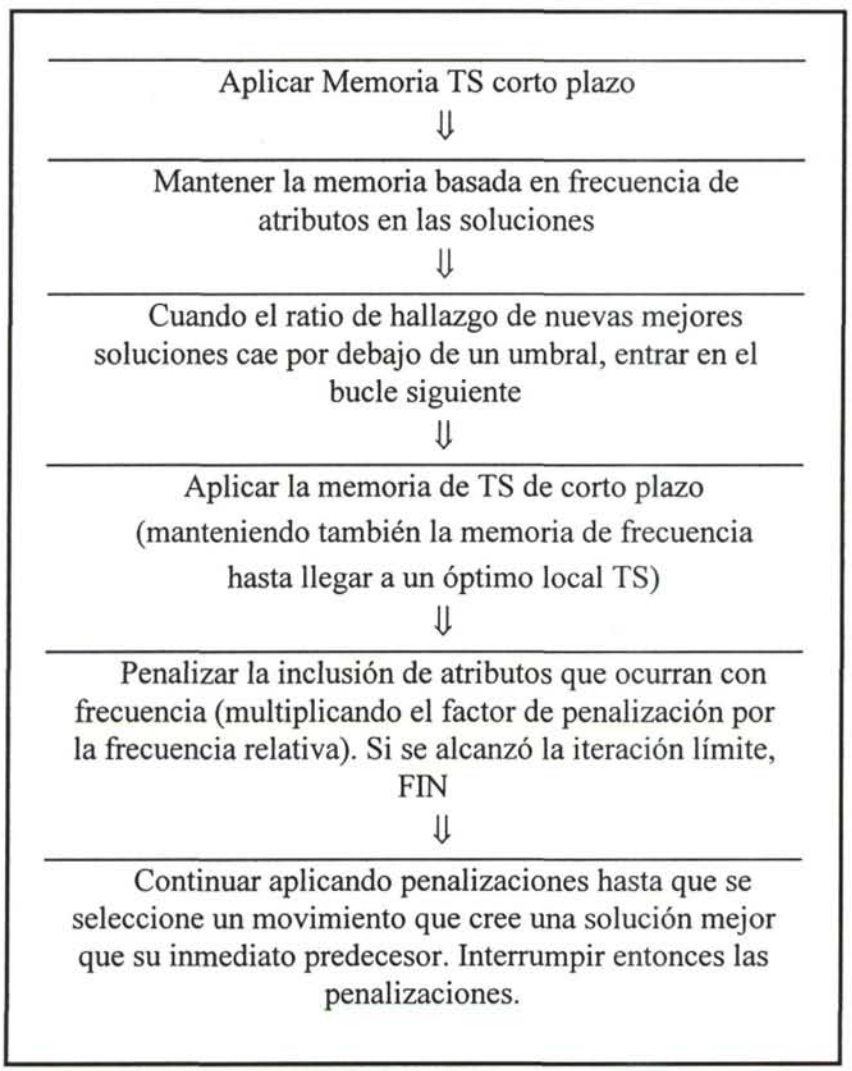


Fig. 1.14 Enfoque simple de diversificación en Búsqueda Tabú (tomado de Adenso Díaz y otros (1.996)).

En problemas de rutas es habitual para la fase de diversificación, utilizar la frecuencia con la que ha aparecido el

arco que une dichos puntos en las soluciones visitadas, como distancia entre dos puntos en la matriz de distancias. De esta forma, el proceso tiende a seleccionar arcos que anteriormente han aparecido poco y por lo tanto espacios de soluciones poco visitados. Otro método que está dando buenos resultados, consiste en usar un algoritmo constructivo que diseñe una solución inicial pero utilizando esta nueva matriz de distancias (*"re-start"*).

1.4.2.3.4 Oscilación Estratégica

En el marco de Tabu Search, la oscilación estratégica es un proceso usado para variar la dirección de la búsqueda y las regiones visitadas. La oscilación estratégica opera orientando los movimientos en relación a una cierta frontera en donde el método normalmente se detendría. Sin embargo no se detiene, ya que las reglas para la elección de los movimientos se modifican, para permitir que la región al otro lado de la frontera sea alcanzada. Posteriormente, se fuerza al procedimiento a regresar a la zona inicial. El proceso de aproximarse, traspasar, y volver sobre una determinada frontera, crea un patrón de oscilación que da nombre a esta técnica.

Una de sus aplicaciones más útiles es crear un proceso de búsqueda expandida, que puede admitir soluciones infactibles durante la búsqueda. Las soluciones infactibles se permiten, pero se penalizan por su grado de infactibilidad.

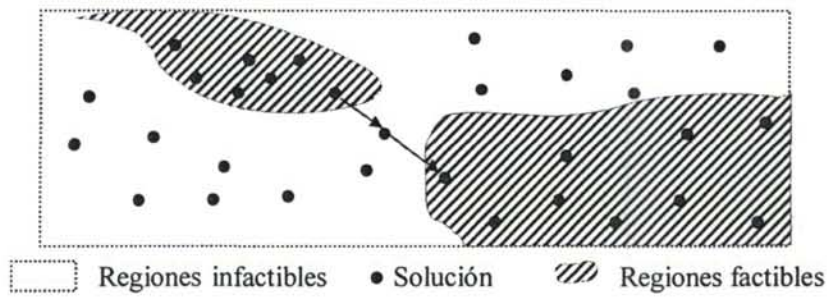


Figura 1.15 Oscilación estratégica

Para Glover F. y otros (1.995) existen al menos tres razones para considerar el uso de la oscilación estratégica cuando se resuelven problemas de optimización:

- La oscilación estratégica dota de mecanismos para cruzar regiones de infactibilidad en el curso de la búsqueda de las solución óptima. En algunos casos el intento para alcanzar la solución óptima puede requerir seguir un camino tortuoso y largo a través del espacio factible, mientras que si se permite entrar en regiones infactibles, entonces el óptimo puede encontrarse fácilmente.

- En algunos problemas el objetivo de obtener soluciones factibles es tan duro como el de obtener la solución óptima.

- Un tercer atributo atractivo de oscilación estratégica es un efecto indirecto. Cualquier búsqueda heurística que aspire a encontrar soluciones óptimas debe asegurar diversidad

suficiente en la búsqueda. La oscilación estratégica provee diversidad mejorando la robustez del método.

Implementaciones más complejas que la consideración de soluciones infactibles pueden crearse identificando determinadas estructuras de soluciones que no son visitadas por el algoritmo y considerando procesos de construcción/destrucción asociados a éstas. La oscilación estratégica proporciona un medio adicional para lograr una interacción muy efectiva entre intensificación y diversificación.

1.4.2.4 Aplicaciones de TS a Problemas de Rutas.

Dos de las primeras implementaciones de Búsqueda Tabú para el VRP son las de Willard J.A.G. (1.989) y Pureza V.M. y Franca P.M. (1.991). En el primer caso, la solución se considera como una única cadena de puntos (una única ruta gigante) y se define como soluciones factibles las que pueden ser alcanzadas desde la solución actual mediante intercambios 2-óptimos y 3-óptimos (Lin S. (1.965)). La siguiente solución viene dada por el mejor movimiento no tabú. En tres de los problemas de referencia de Christofides N. y otros (1.979), el algoritmo propuesto no parece ser competitivo con las mejores soluciones conocidas hasta ese momento.

El segundo trabajo define los vecindarios de una solución mediante movimientos de un vértice a una ruta diferente a la suya, o mediante intercambio de vértices entre dos rutas diferentes siempre preservando la factibilidad. Como en el trabajo anterior se selecciona el mejor movimiento factible no tabú en cada iteración. Aunque mejores que los aportadas por el algoritmo de Willard J.A.G. (1.989), esta implementación no consigue especialmente buenos resultados. Investigaciones posteriores demuestran que se necesitan mecanismos de búsqueda más elaborados.

El algoritmo de Osman I.H. (1.993) define los entornos mediante λ-intercambios con λ=2. Esto incluye una combinación de movimientos 2-óptimos, reasignación de vértices a diferentes rutas, e intercambios de vértices entre dos rutas. En una versión del algoritmo llamada BA (*best-admissible*, mejor-admissible), se explora todo el vecindario y se selecciona el mejor movimiento factible no tabú. En otra versión llamada FBA (*first-best-admissible*, primer mejor movimiento), se selecciona el primer movimiento admisible que produce una solución mejor que la actual, si existe uno; en caso contrario se selecciona el mejor movimiento admisible. Los resultados indican que las dos implementaciones producen excelentes resultados pero pueden ser aún mejoradas en muchos casos.

El "Taburoute" de Gendreau M. y otros (1.994-b) es preferentemente complejo y contiene varias características

innovadoras. La estructura de vecindario se define mediante todas las soluciones que se pueden alcanzar desde la solución actual eliminando un vértice de su ruta actual e insertándolo en otra ruta que contenga uno de sus *p* vecinos más cercanos usando GENI (*GENeralized Insertion*), un procedimiento de inserción generalizada desarrollado por Gendreau M. y otros (1.992) para el TSP. El resultado puede ser la eliminación de una ruta existente o la creación de una nueva. Una segunda importante característica de Taburoute es que el proceso de búsqueda examina soluciones que pueden ser infactibles con respecto a las restricciones de capacidad o de longitud máxima de la ruta. Más exactamente, la función objetivo contiene dos términos de penalización, uno que mide el exceso de capacidad, y el otro que mide el exceso de duración, cada uno equilibrado mediante parámetros de auto-ajuste: cada diez iteraciones, cada parámetro se divide por 2 si las 10 soluciones previas han sido factibles o se multiplica por 2 si han sido todas infactibles. Este modo de proceder, produce una mezcla de soluciones factibles e infactibles, y reduce la probabilidad de que el proceso de búsqueda quede atrapado en un mínimo local. En varios puntos durante el proceso de búsqueda, Taburoute reoptimiza la ruta en la que se ha insertado un vértice. Esto se consigue mediante el uso de rutinas de post-optimización para el TSP, US (*Unstringing Stringing*, desagrupamiento y agrupamiento), desarrolladas también por Gendreau M. y otros (1.992). Taboroute no usa de echo una lista tabú, sino órdenes tabú aleatorias. En el momento en que se mueve un vértice de una ruta

r a otra *s* en la iteración *t*, su reinserción en la ruta *r* se prohibe hasta la iteración $t+\theta$, donde θ es un valor entero generado aleatoriamente en el intervalo [5, 10]. Incluso otra característica de Taburoute se usa como estrategia de diversificación; consiste en penalizar vértices que se han movido con frecuencia para incrementar la probabilidad de considerar vértices con menos movimientos. La función objetivo se incrementa artificialmente añadiéndola un término proporcional a la frecuencia absoluta de movimientos del vértice *v* que se considera. Finalmente, Taburoute usa puntos de partida falsos (*false starts*). Inicialmente se generan varias soluciones y se lleva a cabo una búsqueda limitada en cada una de ellas. Se selecciona la mejor solución y se toma como punto de partida.

El algoritmo de Taillard E.D. (1.993) contiene algunas de las características de Taburoute: duración tabú aleatoria y diversificación. Define los vecindarios utilizando los λ-intercambios de Osman (1.993). Antes que ejecutar las inserciones con GENI, el algoritmo usa inserciones estándares, de ese modo permite que cada inserción sea llevada a cabo en menos tiempo, y la factibilidad sea siempre mantenida. Las rutas individuales se reoptimizan usando el algoritmo de optimización de Volgenant A. y Jonker R. (1.983). Una característica novedosa del algoritmo de Taillard es la descomposición del problema principal en subproblemas. En problemas relativos a planicies, esos subproblemas se obtiene mediante una partición inicial de los

vértices en sectores centrados en el depósito (punto 1-origen) y en regiones concéntricas dentro de cada sector. La combinación de estas estrategias produce excelentes resultados computacionales.

El algoritmo de Xu J. y Kelly J.P. (1.996) usa vecindarios más elaborados que los empleados en los dos trabajos anteriormente comentados. Considera intercambios de vértices entre dos rutas, una reposición global de algunos vértices en otras rutas y mejoras en rutas locales. Se han desarrollado aproximaciones para calcular los costes de expulsión e inserción, teniendo en cuenta la capacidad del vehículo. La reoptimización de las rutas se realiza mediante intercambios 3-óptimos (Lin S. (1.965)) y rutinas de mejora de Búsqueda Tabú. El algoritmo se controla mediante varios parámetros que son ajustados dinámicamente en el proceso de búsqueda. Se memoriza un conjunto de las mejores soluciones y periódicamente se usa para reinicializar la búsqueda con nuevos valores de los parámetros. En conjunto, este algoritmo ha conseguido varias mejores soluciones en instancias de referencia, pero es justo decir que no es tan efectivo como otras implementaciones de TS. Tiende a requerir sustanciales esfuerzos computacionales y puede resultar problemático el afinamiento adecuado de algunos de sus parámetros.

El algoritmo de Rego C. y Roucairol C. (1.996) tiene como característica principal el uso de *cadenas de expulsión* (*ejection chains*) para moverse desde una solución a otra. Una expulsión

consiste en mover un vértice desde la posición que ocupa al lado de otro vértice, de tal manera que se crea una cadena reacción de l niveles. Una cadena de expulsión de nivel l consiste en reemplazar el triplete $(v_{i-1}^k, v_i^k, v_{i+1}^k)$ $(k = 1, ...,l)$ por el triplete $(v_{i-1}^k, v_i^{k-1}, v_{i+1}^k)$ (k = 1, ..., l) y trasladar v_i^l. Se define una condición de autenticidad para garantizar que la solución resultante permanece factible, es decir, que cada arco no aparece más de una vez. En un paso común del algoritmo, se consideran un número de vértices como candidatos para una expulsión, junto con sus vecinos más cercanos que no producen una cadena de expulsión ilegítima. Se consideran soluciones intermedias infactibles, al igual que en Taburoute. Se desarrolló una implementación paralela de este procedimiento. De nuevo esta implementación de TS produce resultados de calidad, pero no alcanza a los mejores resultados conocidos. Una variante de este algoritmo, denominada *Subpath Ejection Method*, ha sido introducida por Rego C. (1.998). Deafortunadamente, no parece mejorar el algoritmo TS previo.

Barbarosogly G. y Ozgur D. (1.999) describen un algoritmo simple mejor de TS que contiene estrategias de no diversificación y en el que sólo se examinan soluciones factibles. Los entornos de las soluciones se definen mediante un esquema de λ-intercambio que favorece vértices relativamente distantes del centroide de su ruta actual y cerca del centroide de la nueva ruta. Las reoptimizaciones de las rutas se realizan aplicando un procedimiento 2-óptimo. El método fue aplicado a las 6 instancias

con restricciones de capacidad del conjunto de Christofides N. y otros (1.979) y se obtuvieron interesantes resultados.

Uno de los más interesantes desarrollos que se han dado en TS en los últimos años es el concepto de *memoria adaptativa*, desarrollada por Rochat Y. y Taillard E.D. (1.995). Se usa en la mayoría de los casos de TS, pero su aplicabilidad no está limitada a este tipo de metaheurísticos. Una memoria adaptativa es un conjunto de buenas soluciones que se actualizan dinámicamente a lo largo del proceso de búsqueda. Periódicamente algunos elementos de esas soluciones son extraídos del grupo combinados de diferentes maneras para producir nuevas buenas soluciones. En el VRP, las rutas de vehículos seleccionadas de varias soluciones serán usadas como puntos de comienzo. El proceso de extracción da un amplio peso a aquellas rutas que pertenecen a las mejores soluciones. Cuando se selecciona esas rutas, hay que tener cuidado y evitar incluir un punto de visita, dos veces en la misma solución. Esta restricción significa que el proceso de selección terminará a menudo con una solución parcial que tendrá que ser completada usando un heurístico de construcción. La memoria adaptativa puede mejorar la estrategia de búsqueda; en este trabajo se consiguen obtener dos mejores nuevas soluciones de las 14 instancias estándares del VRP.

Otro concepto prometedor es el *Granular Tabu Search* (GTS). introducido por Toth P. y Vigo D. (1.998);este método ha

conseguido excelentes resultados para el VRP. La idea principal que está detrás de GTS desciende de la observación de que las aristas más largas de un grafo sólo tiene una pequeña probabilidad si pertenecen a una solución óptima. Por consiguiente, eliminando todas las aristas cuya longitud excede un *umbral de granularidad*, se evitará considerar en el proceso de búsqueda varias soluciones no prometedoras.

1.4.3 GRASP.

GRASP son las iniciales en inglés de Procedimientos de Búsqueda basados en funciones Avidas, Aleatorias y Adaptativas (*Greedy Randomize Adaptive Search Procedures*); se dieron a conocer a finales de los ochenta en el trabajo de Feo T.A. y Resende M.G.C. (1.989), pero han tenido un desarrollo más reciente que los otros metaheurísticos. Una amplia descripción se puede encontrar en un trabajo posterior de los mismos autores Feo T.A. y Resende M.G.C. (1.995).

GRASP es una técnica simple aleatoria e iterativa, en la que cada iteración provee una solución al problema que se esté tratando. La mejor solución de todas las iteraciones GRASP se guarda como resultado final. Hay dos fases en cada iteración GRASP: la primera construye secuencial e inteligentemente una

solución inicial por medio de una función ávida, aleatoria y adaptativa; la segunda fase aplica un procedimiento de búsqueda local a la solución construida, con la esperanza de encontrar una mejora. En pseudocódigo puede escribirse de la forma siguiente:

Algoritmo GRASP

Repetir

Construir una solución Ávida Aleatoria

(Fase de Construcción)

Aplicar Búsqueda local a la solución obtenida

(Fase de mejora)

Hasta alcanzar una condición de parada

1.4.3.1 Fase de Construcción.

En la Fase de construcción se va añadiendo en cada paso un elemento, hasta obtener la solución completa. En cada iteración, la elección del próximo elemento para ser añadido a la solución parcial, viene determinada por una función *ávida* (*greedy*). Esta función mide el beneficio, según la función objetivo, de añadir cada uno de los elementos y elige la mejor opción. Esta medida es *miope,* en el sentido de que no tiene en cuenta qué ocurrirá en iteraciones sucesivas al realizar una elección, sino únicamente lo que pasa en esa iteración.

El heurístico es *adaptativo*, ya que los beneficios asociados con cada elemento se actualizan en cada iteración de la fase de construcción, para reflejar los cambios producidos por la selección de elementos previos.

GRASP es *aleatorizado* porque no selecciona el mejor candidato según la función ávida adaptada. A continuación se describe un sencillo ejemplo que ilustra esta idea: consideremos el problema de la mochila con los datos que muestra la figura 1.16.

El mejor elemento para empezar a construir la solución sería el segundo; después sólo podríamos añadir el tercero. La solución obtenida sería una mochila llena (peso=10) con un valor de 9+1=10 unidades. Si en lugar de tomar el mejor elemento, tomamos el segundo mejor la solución a la que se llega es una mochila llena (peso=10) con un valor de 11 unidades. La conclusión a la que se llega, es que no siempre es bueno tomar el mejor elemento disponible para la fase constructiva.

Elemento	Peso	Valor	Ratio valor por unidad de peso
1	10	11	1,1
2	6	9	1,5
3	4	1	0,25

Capacidad de la mochila : 10 unidades de peso.

Figura 1.16 Problema de la mochila

Con los mejores elementos a añadir se construye una lista denominada *Lista Restringida de Candidatos* (*RCL* en inglés), y se elige de forma aleatoria uno de los mejores candidatos de dicha lista, que no será necesariamente el mejor. La aleatoriedad sirve como mecanismo de diversificación en GRASP.

El tamaño de la lista de candidatos influye en la calidad de la solución; si la lista de candidatos se compone de un único elemento, sólo se podrá obtener una solución, y la variación de soluciones será por lo tanto cero; dada una función ávida, el valor de la solución en este caso sería bueno, pero probablemente subóptimo. Si se determina un número menos restrictivo, se pueden obtener diferentes soluciones de gran variedad. Experimentos realizados por Feo T.A. y Resende M.G.C. (1.995) demuestran que un incremento en el tamaño de la lista de candidatos se traduce en un mayor número de ejemplos con mejores soluciones.

La elección de cada elemento y por consiguiente la solución obtenida no es totalmente aleatoria ni totalmente determinística. Se espera obtener buenas soluciones iniciales de forma rápida que son mejoradas con procedimientos de búsqueda local.

Se puede mejorar el procedimiento por ejemplo, tomando como punto de partida ciertas subestructuras que se sabe forman parte de una solución óptima.

1.4.3.2 Fase de Mejora.

Como es el caso de muchos métodos determinísticos, no se garantiza que la solución generada por la fase de construcción de GRASP sea un óptimo local respecto a una definición simple de vecindario. Por ello se aplica búsqueda local para mejorar cada solución construida. La fase de búsqueda local finaliza cuando no se encuentra una solución mejor en el vecindario de la solución actual. GRASP se basa en realizar múltiples iteraciones y quedarse con la mejor, por lo que no es especialmente beneficioso para el método el detenerse demasiado en mejorar una solución dada.

El éxito de esta segunda fase viene determinado por la acertada elección de la estructura del vecindario, técnicas eficientes de búsqueda en vecindarios y la solución inicial.

El procedimiento de optimización local puede requerir un tiempo exponencial para un punto inicial arbitrario. Empíricamente su eficiencia mejora significativamente y se obtiene más éxito cuanto mejor es la solución obtenida en la fase de construcción. A menudo es posible generar muchas soluciones GRASP en el mismo tiempo que se requiere para que el procedimiento de optimización converja desde un punto inicial aleatorio simple. En base a las observaciones empíricas, en la mayoría de las aplicaciones se puede afirmar que la distribución de la muestra generalmente tiene un valor en promedio que es inferior al

obtenido por un procedimiento determinista, sin embargo, la mejor de esas soluciones GRASP es en general significativamente mejor que la solución simple obtenida a partir de un punto inicial aleatorio.

Una variante de este método que puede ayudar a disminuir los requerimientos computacionales consiste en construir un determinado número de soluciones factibles y aplicar la búsqueda local a la más prometedora de todas ellas. En el caso de Kontoravdis G. y Bard J.F. (1.995) la búsqueda local se aplica a la mejor solución encontrada tras cada cinco iteraciones de la primera fase.

Otras variaciones, también de Kontoravdis G. y Bard J.F. (1.995), sugieren las siguientes ideas para mejorar las soluciones:

- Introducir "azar" y aceptar, con cierta probabilidad, un movimiento que permita un deterioro en la solución actual para escapar de un óptimo local. Este es un principio básico de otros metaheurísticos como temple simulado o búsqueda tabú.

- Permitir, con cierta probabilidad, algunos intercambios no factibles con la esperanza de que la factibilidad pudiera ser recuperada en una etapa posterior. Este es el concepto de oscilación estratégica en búsqueda tabú.

Sugieren los autores, que estos procedimientos se pueden aplicar a la mejor solución encontrada por GRASP al finalizar la optimización, ya que en otro caso, el tiempo de computación sería excesivo.

1.4.3.3 Parámetros.

Tal y como señalan Feo T.A. y Resende M.G.C. (1.995) una de las características más relevantes de GRASP es su sencillez y facilidad de implementación: se requiere tan solo fijar el tamaño de la lista de candidatos y el número de iteraciones a realizar; el esfuerzo necesario para la elección de los valores de los parámetros es pequeño. El desarrollo puede enfocarse por lo tanto en la implementación eficiente de estructuras de datos que aseguren unas iteraciones rápidas.

El tamaño de la lista de candidatos influye en la calidad de la solución. Para fijar dicho tamaño es necesario realizar estudios computacionales. El aumento del valor de dicho parámetro aumenta el número de iteraciones de GRASP necesarias para obtener una solución de calidad.

El criterio de parada puede venir dado por el número de iteraciones o también por el hecho de haber encontrado la solución buscada.

1.4.3.4 Aplicaciones prácticas de GRASP al VRP.

Como ya se ha comentado, una característica especialmente atractiva de GRASP es la facilidad con la que se implementa ya que se necesitan manejar pocos parámetros. En las implementaciones en las que ha colaborado la autora para problemas de rutas, los experimentos realizados permitían fijar los siguientes valores (Delgado C. R. (1.998)):

- la condición de parada se produce cuando transcurren 100 iteraciones sin mejora en las soluciones encontradas,
- en la Lista Restringida de Candidatos se incluyen los elementos que se alejan menos del 40% del mejor candidato según la función voraz.

En el apartado 3.1 del capitulo 3 se expondrá con más detalle esta aplicación.

Un trabajo significativo es el de Kontoravdis G. y Bark J.F. (1.995). En la comparación de soluciones, dan preferencia a las de menor número de vehículos en primer lugar, y posteriormente a las de menor número de kilómetros. En la primera fase de elige un número r de rutas y en cada iteración se añade un cliente a alguna de las rutas. Cada cinco iteraciones se selecciona la mejor solución; dicha solución pasa a la segunda fase y se le aplica Búsqueda Local, tratando de reducir el número de rutas.

1.5.4 Concentración Heurística.

Este método ha sido propuesto en los trabajos de Rossing K.E (1.997), Rossing K.E. y Revelle C.S. (1.997) y Rosign K.E. y otros (1.998) aplicado al problema de las *p-medianas*.

Según Rossing K.E. y Revelle C.S. (1.997), "... *cada óptimo local puede ser considerado como una fuente de información acerca de la estructura de una parte de la solución óptima. Se espera que un conjunto de aquellos den información sobre todas las partes de esta*".

La idea de Concentración Heurística (*Heuristic Concentration*, HC) es el resultado de observaciones de diferentes pruebas aleatorias de un heurístico de intercambio para el problema de las *p-medianas*. Generalmente las soluciones aportadas eran muy similares en cuanto a nodos de demanda seleccionados; o lo que es lo mismo, la gran mayoría de los nodos nunca se seleccionaban. Esto permitió el desarrollo del *Conjunto de Concentración* (*Concentration Set*, CS) como la unión del conjunto de nodos seleccionados en las diferentes soluciones subóptimas.

También observaron que un número de nodos de demanda eran frecuentemente seleccionados como localizaciones en todas las diferentes soluciones subóptimas. Esto permitía particionar CS en

1.4.3.4 Aplicaciones prácticas de GRASP al VRP.

Como ya se ha comentado, una característica especialmente atractiva de GRASP es la facilidad con la que se implementa ya que se necesitan manejar pocos parámetros. En las implementaciones en las que ha colaborado la autora para problemas de rutas, los experimentos realizados permitían fijar los siguientes valores (Delgado C. R. (1.998)):

- la condición de parada se produce cuando transcurren 100 iteraciones sin mejora en las soluciones encontradas,
- en la Lista Restringida de Candidatos se incluyen los elementos que se alejan menos del 40% del mejor candidato según la función voraz.

En el apartado 3.1 del capitulo 3 se expondrá con más detalle esta aplicación.

Un trabajo significativo es el de Kontoravdis G. y Bark J.F. (1.995). En la comparación de soluciones, dan preferencia a las de menor número de vehículos en primer lugar, y posteriormente a las de menor número de kilómetros. En la primera fase de elige un número r de rutas y en cada iteración se añade un cliente a alguna de las rutas. Cada cinco iteraciones se selecciona la mejor solución; dicha solución pasa a la segunda fase y se le aplica Búsqueda Local, tratando de reducir el número de rutas.

1.4.4 Concentración Heurística.

Este método ha sido propuesto en los trabajos de Rossing K.E (1.997), Rossing K.E. y Revelle C.S. (1.997) y Rosign K.E. y otros (1.998) aplicado al problema de las *p-medianas*.

Según Rossing K.E. y Revelle C.S. (1.997), *"... cada óptimo local puede ser considerado como una fuente de información acerca de la estructura de una parte de la solución óptima. Se espera que un conjunto de aquellos den información sobre todas las partes de esta"*.

La idea de Concentración Heurística (*Heuristic Concentration*, HC) es el resultado de observaciones de diferentes pruebas aleatorias de un heurístico de intercambio para el problema de las *p-medianas*. Generalmente las soluciones aportadas eran muy similares en cuanto a nodos de demanda seleccionados; o lo que es lo mismo, la gran mayoría de los nodos nunca se seleccionaban. Esto permitió el desarrollo del *Conjunto de Concentración* (*Concentration Set*, CS) como la unión del conjunto de nodos seleccionados en las diferentes soluciones subóptimas.

También observaron que un número de nodos de demanda eran frecuentemente seleccionados como localizaciones en todas las diferentes soluciones subóptimas. Esto permitía particionar CS en

dos subconjuntos CS_0 (*open*, abierto) y CS_f (*free*, libre); CS_0 es el conjunto de elementos o nodos que aparecen en todos los óptimos locales seleccionados; se asume que son realmente elementos de la solución óptima y se resuelve el problema dejando dichos nodos fijos. El resto de nodos que aparecen en alguno de los óptimos locales pero no en todos, forman el CS_f y están disponibles para su posible elección.

1.4.4.1 Fases de Concentración Heurística.

El proceso de Concentración Heurística consta de dos fases:

- En la primera, se ejecuta durante un número de veces un heurístico básico; las mejores soluciones, según el valor de la función objetivo, forman lo que hemos denominamos el Conjunto de Concentración que se utilizará en la segunda fase.

- En la segunda fase se utiliza un método, heurístico o exacto, para desarrollar una nueva solución a partir de la información del Conjunto de Concentración; generalmente la solución que se obtiene es mejor que las obtenidas en la primera fase (Rosing K.E. (1.997)).

En la primera fase hay que fijar el número de ejecuciones del heurístico básico (algún método de intercambio heurístico). Evidentemente un mayor número de iteraciones supone un mayor

tiempo de computación y un mayor número de soluciones. Es necesario estudiar en que momento las soluciones aportadas por reiteradas iteraciones del algoritmo básico no proporcionan información adicional de interés.

Determinado número *m*, de las mejores soluciones según el valor de la función objetivo, formarán el CS, utilizado en la segunda fase para encontrar la solución "óptima".

La generación de soluciones en la primera fase puede ser aleatoria o determinística, pudiendo clasificar a Concentración Heurística como una técnica aleatoria o determinística.

El CS se espera que sea bastante más reducido que el conjunto de elementos original; por ello, la aplicación de un algoritmo heurístico de calidad e incluso de un exacto en el último paso, según el problema que se esté tratando, puede no requerir un tiempo de computación excesivo. En este sentido la variante propuesta por Rossing K.E. y Revelle C.S. (1.997) y Rosing K.E. y otros (1.998) de descomponer *CS* en dos subconjuntos CS_0 (abierto) y CS_f (libre), permite resolver el problema dejando fijos los elementos de CS_0 en la solución y seleccionando los elementos que quedan en CS_f, reduciendo aún más el tamaño del problema.

Una vez construido el Conjunto de Concentración se pasa a la segunda fase. En ella se va a diseñar un algoritmo que *concentre*

la búsqueda de elementos en el Conjunto de Concentración. Como se ha comentado, dado que con CS el tamaño del problema generalmente se reduce, se ha de estudiar en cada caso la posibilidad de aplicar un algoritmo exacto o un heurístico de alta calidad, ya que es factible que el tiempo de computación no sea excesivo.

La estrategia que se sigue en Concentración Heurística es la de intensificar la búsqueda en las soluciones que parecen ser buenas. Como se verá más adelante (capítulo 3) esta forma de intensificación puede ser incorporada en procesos de otros metaheurísticos, obteniendo resultados muy convincentes.

A continuación se presentan las dos fases del algoritmo de Concentración Heurística en pseudocódigo:

Algoritmo Concentración_Heurística

Fase I:

Repetir

Generar una solución y aplicar Búsqueda local

Hasta un determinado número de iteraciones.

Registrar las "m" mejores diferentes soluciones obtenidas (donde "m" es un parámetro preestablecido).

Fase II:

Definir CS (Conjunto de Concentración) como el conjunto de elementos que aparecen en alguna de dichas soluciones. Aplicar un método (exacto o heurístico) al problema original restringiendo (o concentrando) la selección de elementos a CS.

1.4.4.2 Aplicaciones Prácticas al VRP.

En Delgado C.R. (1.998) se propone un metaheurístico para el problema de rutas con ventanas de tiempo, carga y descarga simultánea y flota heterogénea (ver apéndice 1.3, 1.5, 1.6), basado en un proceso de tipo *Concentración Heurística,* que en su primera fase construye soluciones ávido-aleatorias como se hace en *GRASP*.

En Pacheco J. y Delgado C.R. (2.000-a) se diseña un metaheurístico para el problema de rutas con ventanas de tiempo, carga y descarga simultánea y flota heterogénea que utiliza de las ideas de *Concentración Heurística* en la fase de Intensificación de *Búsqueda Tabú.*

En el capítulo tres se desarrollan con más detalles ambas proposiciones y sus resultados.

1.5 Metaheurísticos Evolutivos o Basados en Población.

A lo largo de la historia diversas teorías han intentado explicar el funcionamiento de la evolución de los seres vivos; pero fue a mediados del siglo XIX cuando se postularon los principios básicos, que en gran medida han condicionado la visión actual de la Evolución. Estos principios se asientan sobre la *Teoría de la Selección Natural* de Darwin C. (1.859) y sobre *Herencia Genética* de Mendel G. (1.865). Cotta-Porras C. (1.998) hace el siguiente resumen de dichos principios:

" • *La evolución es un proceso que no opera directamente sobre organismos, sino sobre cromosomas. Estos son unos instrumentos orgánicos mediante los cuales se codifica la estructura de un ser vivo, i.e., un ser vivo se crea mediante la decodificación de un conjunto de cromosomas. Estos cromosomas (más concretamente, la información que contienen) pasan de una generación a otra mediante los procesos reproductivos.*

- *El proceso evolutivo propiamente dicho tiene lugar durante la etapa de reproducción. En la naturaleza existe una gran cantidad de mecanismos reproductivos. Los más básicos son los de mutación (que modelan la variabilidad del material genético de los individuos) y los de recombinación (que*

combinan los materiales genéticos de los padres para producir el de los descendientes).

- *La selección natural es el mecanismo que permite relacionar los cromosomas con la eficiencia de las entidades que representan, provocando que aquellos organismos eficaces y adaptados al medio se reproduzcan con mayor probabilidad que aquellos que no lo están, ocasionando de esta manera la extinción de los organismos inadaptados o poco eficientes."*

La sencillez de estos principios ha favorecido que muchos investigadores hayan intentado trasladarlos al campo de la algoritmia. Así los *Algoritmos Evolutivos* se inspiran en ellos, modificándolos para obtener sistemas eficientes para la resolución de diferentes problemas.

Existe un enfoque diferente que intenta reproducir fielmente los principios naturales, simula la naturaleza; esto ha originado el campo denominado *Vida Artificial* (Langton C.G. (1.989)), con el que los Algoritmos Evolutivos han tenido una gran interacción, gracias a la cual, se han producido avances notables en ambos campos.

Un *Algoritmo Evolutivo* es un proceso estocástico e iterativo que opera sobre un conjunto P de *individuos* que constituyen lo que se denomina *población*; cada individuo contiene uno o más

cromosomas que le permiten representar una posible solución al problema, la cual se obtiene gracias a un proceso de codificación/decodificación. La población inicial es generada aleatoriamente o con la ayuda de algún heurístico de construcción. Cada individuo es evaluado a través de una función de adecuación (*fitness*). Estas evaluaciones se usan para predisponer la selección de cromosomas de forma que los superiores (aquellos con mayor evaluación) se reproduzcan más a menudo que los inferiores.

El algoritmo se estructura en tres fases principales que se repiten de forma iterativa, lo que constituye el *ciclo reproductivo básico* o *generación*. Dichas fases son: selección, reproducción y reemplazo.

A finales de los 60 y principios de los 70 comenzaron a existir los algoritmos evolutivos tal y como hoy los conocemos. Distintos investigadores trataron de trasladar los principios de la Evolución al campo de la algoritmia originando diferentes modelos que pueden agruparse en tres grandes familias:

- PROGRAMACIÓN EVOLUTIVA (EVOLUTIONARY PROGRAMMING).

  Esta familia de algoritmos tiene su origen en el trabajo de Fogel L.J. y otros (1.966), y ponen un especial énfasis en la adaptación de los individuos más que en la evolución del material genético de éstos. Ello implica una visión mucho más abstracta del proceso, en la cual se modifica directamente el comportamiento

de los individuos en lugar de trabajar sobre sus genes. Dicho comportamiento se modela mediante estructuras de datos relativamente complejas como pueden ser autómatas finitos. Tradicionalmente, estas técnicas emplean mecanismos de reproducción asexual y técnicas de selección mediante competición directa entre individuos.

Las diferencias fundamentales con los algoritmos genéticos son dos: no se impone restricción sobre la representación y la operación de mutación simplemente modifica aspectos de la solución según una distribución estadística que pondera como muy probables variaciones poco importantes en la descendencia y como cada vez menos probables, variaciones considerables a media que se vaya aproximando al óptimo.

- ESTRATEGIAS DE EVOLUCIÓN (EVOLUTIONS STRATEGIE. EE).

Estas técnicas comenzaron a desarrollarse en Alemania. Su objetivo inicial era servir de herramienta para optimización de parámetros en problemas de ingeniería. Al igual que la programación evolutiva con la que se halla estrechamente emparentada, basa su funcionamiento en el empleo de un operador de reproducción asexual o de mutación, especialmente diseñado para trabajar con números reales. En la Universidad Técnica de Berlín, durante la búsqueda de las formas óptimas de los cuerpos en un flujo (que se reducía entonces a una experimentación laboriosa e intuitiva), se les ocurrió probar con cambios aleatorios en los parámetros que definían la forma,

siguiendo el ejemplo de las mutaciones naturales y nació así la EE. Se empezó a construir un experimentador automático que funcionara según las sencillas reglas de mutación y selección, y se probó la eficiencia de estos nuevos métodos. Recibieron el apoyo financiero de la Sociedad de Investigación Alemana (DFG) que posibilitó el trabajo que se concluyó de forma provisional en 1.974 con la tesis "Estrategia de evolución y la optimización numérica". De este modo se crearon EE para resolver problemas de optimización técnica y hasta hace muy poco éstas sólo se conocían entre la comunidad de la ingeniería civil, como una alternativa a las soluciones estándares.

- ALGORITMOS GENÉTICOS (GENETIC ALGORITHMS).

"*Técnicas de búsqueda basadas en la mecánica de la selección natural y la genética.*" (Goldberg D.E. (1.989)). Estas técnicas son probablemente el representante más conocido de los algoritmos evolutivos, y aquellas cuyo uso está más extendido. Fueron concebidas originalmente por John Holland y descritas en el ya clásico *Adaptation in Natura and Artificial Systems* (Holland J. (1.975)). Este texto ha tenido una gran trascendencia en el posterior desarrollo de estas técnicas ya que los mecanismos en él descritos han sido tomados durante largo tiempo como auténticos dogmas. Funcionan mediante la creación en una computadora, de una población de individuos representados por los cromosomas, que son en esencia un

conjunto de cadenas de caracteres análogos a los cromosomas de cuatro bases de nuestro ADN.

Posteriormente y debido a la interconexión entre estas tres familias han surgido nuevas variedades de algoritmos evolutivos. Se pueden destacar principalmente dos (Cotta-Porras C. (1.998)):

- PROGRAMAS DE EVOLUCIÓN (EVOLUTIONS PROGRAMS).

  Avalados, entre otros, por los trabajos de Michalewicz Z. (1.992) y (1.993), son técnicas de búsqueda que siguen los mismos principios de funcionamiento que los GAs pero hacen evolucionar estructuras complejas. Actualmente "algoritmo genético" es sinónimo de "programas de evolución", dejando a los anteriormente descritos la expresión "algoritmo genético tradicional".

- PROGRAMACIÓN GENÉTICA (GENETIC PROGRAMMING).

  Inspirada en el trabajo de Cramer M.L. (1.985) y popularizada por Koza J.R. (1.992), la programación genética es otra variante de los algoritmos genéticos en la que se hace evolucionar estructuras (típicamente árboles) que representan programas de ordenador. El objetivo final es el diseño automático de un programa que resuelva una tarea determinada, expresada a partir de una serie de casos de ejemplo.

Debido a que los algoritmos genéticos son probablemente las técnicas más extendidas y las más proclives a ser consideradas herramientas universales les dedicaremos el primer punto de este apartado.

1.5.1 Algoritmos Genéticos.

Las cuestiones que se desarrollan en este apartado son las siguientes:

1. Los Algoritmos Genéticos.
2. Tipos de Representación.
3. Población Inicial y Fitness.
4. Operadores:
   - 4.1 Operadores de Selección
   - 4.2 Operadores de Cruce.
   - 4.3 Operadores de Mutación.
   - 4.4 Otros Operadores.
5. Reemplazo y Condición de Parada.
6. Aplicaciones al VRP.

1.5.1.1 Los Algoritmos Genéticos.

Como se describió por Goldberg D.E. (1.989) y Davis L. (1.991) la evolución natural tiene algunas características, que

motivaron a John Holland a comenzar un intento de búsqueda en este área, dando lugar a lo que es ahora conocido como Algoritmos Genéticos. Estas características están brevemente detalladas por Davis L. (1.991) como sigue:

- La evolución es un proceso que opera en los cromosomas antes que en los seres vivos que codifican.

- Los procesos de selección natural provocan que aquellos cromosomas que codifican estructuras exitosas se reproduzcan con mayor probabilidad que aquellos que no lo son.

- Las mutaciones pueden causar que los cromosomas de hijos biológicos sean diferentes de sus padres biológicos y procesos de recombinación pueden crear cromosomas bastante diferentes en los hijos por la combinación de material de los cromosomas de los padres.

- La evolución biológica no tiene memoria.

Históricamente, el término evolutivo se ha asociado con algoritmos que usaban solamente selección y mutación, mientras que el término genético ha sido asociado a algoritmos que usan selección, mutación, recombinación y una variedad de otros mecanismos inspirados en la naturaleza. (Goldberg D.E. (1.994)). La principal característica de los GAs es el uso del operador de recombinación o cruce como mecanismo principal de búsqueda:

construye descendientes que poseen características de los cromosomas que se cruzan. Su utilidad viene dada por la suposición de que diferentes partes de la solución óptima pueden ser descubiertas independientemente y luego ser combinadas para formar mejores soluciones. Adicionalmente se emplea un operador de mutación cuyo uso se considera importante como responsable del mantenimiento de la diversidad.

La premisa que guía los GAs es que pueden resolverse problemas complejos simulando la evolución en un algoritmo programado por ordenador. La concepción de John Holland es que esto ocurre mediante algoritmos que manipulan strings binarios llamados *cromosomas*. Como en la evolución biológica, la evolución simulada tiene el objetivo de encontrar buenos cromosomas mediante una manipulación ciega de sus contenidos. El término ciego se refiere al hecho de que el proceso no tiene información sobre el problema que intenta resolver.

Los primeros diseños de Holland fueron simples, pero probaron ser efectivos para solucionar problemas considerados difíciles en aquel tiempo. El campo de los GAs se ha desarrollado desde entonces, principalmente como resultado de las innovaciones en 1.980 al incorporar más diseños elaborados con el propósito de resolver problemas en un amplio rango de escenarios prácticos.

Holland representa en forma binaria las soluciones individuales, problema sencillo e independiente de los operadores de cruce y de

las reglas de selección proporcionales. Los miembros de la población son cadenas o cromosomas, que como originalmente se concibieron, son representaciones binarias de vectores de solución. Los GAs seleccionan subconjuntos (normalmente pares) de soluciones de la población, llamados padres, para combinarlos y producir nuevas soluciones llamadas hijos (*offspring*) que posean características de ambos. En los algoritmos originales de Holland, uno de ellos era elegido de acuerdo a su valor de fitness, mientras que el otro padre era elegido aleatoriamente. La operación de sobrecruzamiento daba dos configuraciones binarias, los hijos, a los que se calculaba su fitness. Dichas soluciones reemplazaban dos soluciones de la población elegidas al azar.

Las reglas de combinación para producir hijos se basan en la noción genética de sobrecruzamiento (*crossover*), que consiste en un intercambio de valores de determinadas variables; ocasionalmente se realizan otras operaciones de reproducción como son cambios de valores aleatorios, llamados mutación, o inversión de partes del cromosoma. La mutación, como ya se ha comentado, es importante como responsable del mantenimiento de la diversidad en la población, y, aunque secundaria en relación con el operador de cruce, ocasionalmente introduce alteraciones beneficiosas. La inversión es un mecanismo que altera la ubicación de los genes en los cromosomas, permitiendo que algunos genes que se han coadaptado exitosamente se ubiquen más cerca en el cromosoma, lo que a su vez permite que se aumente la probabilidad de moverse en conjunto durante el sobrecruzamiento.

Los hijos producidos por la unión de los padres y que superan un test de supervivencia están entonces disponibles para ser elegidos padres de las siguientes generaciones. La elección de padres debe de ser pareja en cada generación y se basa en un plan tendente a pruebas aleatorias que en algunos casos (no estándares) se realiza de forma paralela sobre subpoblaciones separadas cuyos mejores miembros se intercambian o comparten periódicamente.

Los componentes que han de considerarse a la hora de implementar un GA son los siguientes:

- Una representación, en términos de "cromosomas", de las configuraciones de cada problema: método de codificación del espacio de soluciones en cromosomas.

- Una manera de crear las configuraciones de la población inicial.

- Una función de evaluación que permita ordenar los cromosomas de acuerdo con la función objetivo: medida de la bondad o función fitness.

- Operadores genéticos que permitan alterar la composición de los nuevos cromosomas generados por los padres durante la reproducción.

- Valores de los parámetros que el algoritmo genético usa (tamaño de la población, probabilidades asociadas con la aplicación de los operadores genéticos).

Un esquema básico de GAs en pseudocódigo podría ser el siguiente:

Procedimiento Algoritmo_Genético.
Generar una población inicial.
Repetir
Evaluar la bondad de cada cromosoma.
Seleccionar y Reproducir cromosomas para crear otros nuevos.
Sustituir cromosomas de la población por los hijos.
Hasta alcanzar un criterio de parada.

1.5.1.2 Tipos de representación.

Las representaciones binarias no son siempre efectivas por lo que algunos investigadores han empezado a explorar el uso de otras representaciones. El propósito de la representación es identificar las características similares del conjunto de soluciones, y diferentes codificaciones dan lugar a diferentes perspectivas y diferentes resoluciones.

Una *representación binaria* es la que viene determinada por una sucesión de unos y ceros; por ejemplo:

1	1	0	0	1	0	1	1	0	1

La *representación entera* se realiza como una secuencia de números enteros; por ejemplo:

5	3	2	4	6	1	8	9	7

Una ejemplo de *variación* de binaria y entera sería la siguiente:

1	0	1	-1	0	1	1	-1	0	-1

La *representación real* se realiza como una secuencia de números reales; por ejemplo:

0,5	3,1	2,4	1,7	9,8

1.5.1.3 Población Inicial y Fitness.

La *población inicial* de un GA puede ser creada de diferentes maneras:

- De forma aleatoria siendo cada cromosoma distinto e independiente de los demás.
- Dando un valor fijo fácil de calcular igual para todos los cromosomas, por ejemplo todos ceros.
- Mediante técnicas más directas como la perturbación de la salida de algún algoritmo goloso.

- Inicialización por perturbación de una solución generada por algún experto en el problema.

La *función de evaluación* de cada cromosoma depende del problema que se esté tratando y de la codificación elegida.

Dada una función objetivo *f(x)*, dicha función puede ser modificada con el objetivo de penalizar las violaciones de restricciones o de fomentar la diversidad:

- valor de la función objetivo + penalización por violación de restricciones:

  $f(x) + p(x)$ *o también:* $f(x)\, p(x)$

- valor de la función objetivo + fomento de diversidad:

  $f(x) + d(x)$ *o también:* $f(x)/\, d(x)$

1.5.1.4 Operadores.

1.5.1.4.1 Operadores de Selección.

El objetivo de este operador es transmitir y conservar la información que se considera valiosa a lo largo de las generaciones. La transmisión de la información se hace por medio de alguna estrategia de selección de los mejores individuos: el propósito es crear una población temporal P' de forma que los

individuos con mayor evaluación estén representados un mayor número de veces que los demás, incrementándose la probabilidad de reproducir miembros que tengan una alta evaluación (*principio de selección natural*); es necesario, sin embargo, permitir un factor de aleatoriedad que posibilite la elección de individuos que aunque inicialmente no parezcan muy adaptados, si puedan ser útiles en posteriores etapas. La selección se trata como un operador genético más, aunque por naturaleza es esencialmente distinto.

Tradicionalmente pueden considerarse tres mecanismos fundamentales de selección:

- **Ruleta o Selección Proporcional** a la adecuación (fitness-proportionate selection). Cada componente de la población seleccionada es escogido de la población actual para reproducirse de manera independiente. Para ello se realiza una selección aleatoria en la que el individuo *i-ésimo* cuya adecuación es f_i, tiene una probabilidad p_i de ser seleccionado. El símil que se establece es el de una *ruleta* en la que se reserva un trozo a cada individuo, proporcional a su valor de bondad. En cada selección se hace girar la ruleta y una flecha en una posición fija indica el elemento elegido. Una descripción detallada del algoritmo que implementa este método se tiene en Goldberg D.E. (1.989).

- **Selección por Ranking**. Ideado por Whitley L.D. (1.987), consiste en construir la población seleccionada atendiendo a

la ordenación de los valores de adecuación en lugar de a sus valores absolutos, esto es, el mejor individuo es seleccionado con probabilidad p_1, el siguiente con p_2, etc. Las formas de calcular las probabilidades son diversas; la más extendida se denomina ranking lineal (Baker J.E. (1.985)).

- **Selección por Torneo** (tournament selection). Mecanismo de selección con menor componente de aleatoriedad que el de la Ruleta, ya que los individuos con bajo grado de bondad pueden ser elegidos con menor probabilidad. Debido a su sencillez, el costo computacional es muy bajo: consiste en tomar un subgrupo de t individuos de la población actual, frecuentemente se toma $t = 2$ (torneo binario); se genera un número aleatorio entre 0 y 1 de tal forma que si el número es menor que un cierto límite K (típicamente 0,75) el individuo seleccionado es el de mayor valor de bondad, en caso contrario el de menor. El parámetro K mide el nivel de aleatoriedad. Este algoritmo de selección comparte las mismas propiedades que la selección mediante ranking; de hecho, pueden parametrizarse de manera que su comportamiento sea globalmente equivalente (Bickle T. y Thiele L. (1.995)).

1.5.1.4.2 Operadores de Cruce.

El cruce no permite dar un salto cualitativo pero asegura una exploración exhaustiva de la información contenida en las soluciones obtenidas hasta el momento. Se aplican diferentes operadores de cambio u operadores reproductivos con el objetivo de producir individuos con nuevas características mejores que las anteriores.

Distintos métodos simulan el proceso de reproducción; entre ellos podemos destacar:

- **Cruce de un Punto** (one-point crossover o single-point crossover): se seleccionan los padres y una posición interior de las cadenas, y se intercambian los segmentos de ambas cadenas a la izquierda de las mismas (figura 1.17). Al irse desarrollando el campo de los AGs se fueron haciendo notorias ciertas limitaciones del one-poin crossover. Una solución fue la utilización del two-point crossover. Booker L.B. (1.987) ha observado empíricamente que el incremento del número de cruces a dos mejora la ejecución de los GAs.

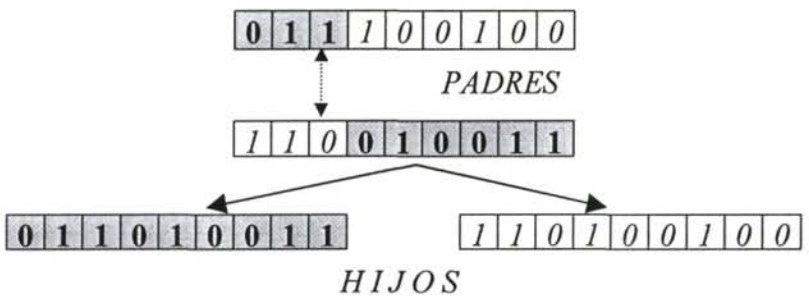


Fig. 1.17 Cruce de Un Punto

- **Cruce de *n* Puntos** o **Cruce Múltiple** (*n*-point crossover).. Constituye una generalización del anterior en el que se seleccionan *n* puntos en el interior de las cadenas y se intercambian los segmentos entre puntos de corte alternos. (ver Eshelman L.J. y otros (1.989) y Goldberg D.E. (1.989)).

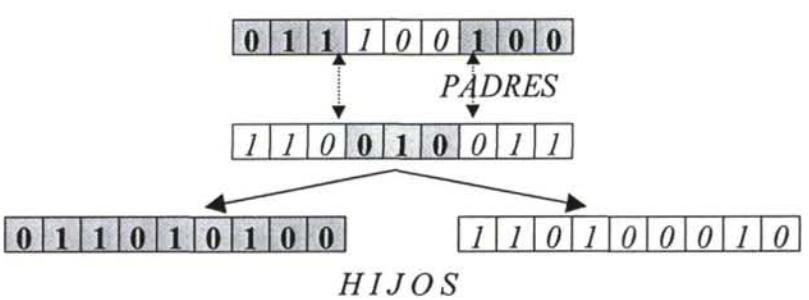


Fig. 1.18 Cruce Múltiple (en dos puntos).

El cruce más utilizado es el cruce en dos punto, n=2 (two-point crossover, figura 1.18), tal y como se describe en Davis L. (1.991).

- **Cruce Uniforme** (uniform crossover): este operador fue ideado por Syswerda G. (1.989), siendo su funcionamiento similar al del cruce de *n* puntos con $n=\lambda$. Para ser precisos, se realiza un test aleatorio para decidir de cuál de los antecesores se toma cada posición de la cadena (figura 1.19).

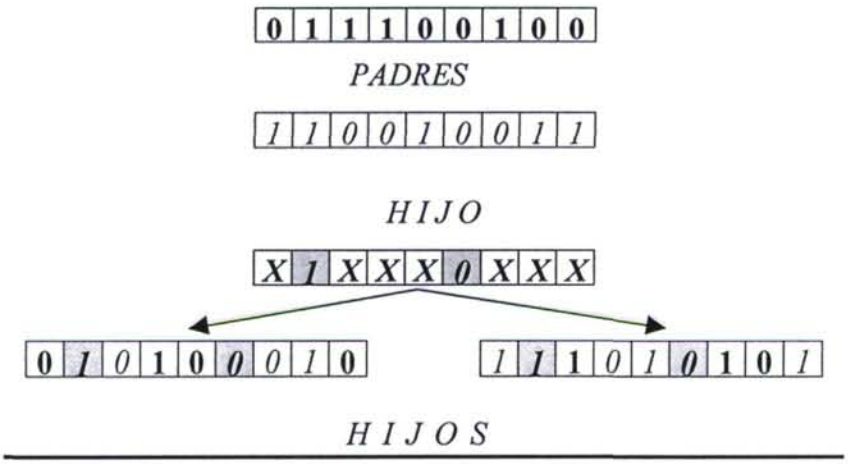


Fig. 1.19 Cruce Uniforme.

- **Cruce Aritmético**: Tipo de cruce descrito para codificación con números reales. Actúa a partir de los padres (x_1, x_2, ..., x_n) y (y_1, y_2, ..., y_n), generando dos nuevos individuos según la forma de la figura 1.20. El parámetro α puede ser constante, y se tiene entonces el *cruce aritmético uniforme*; o puede ser una variable dependiente del número de generación en que se aplique. Se tiene entonces *el cruce aritmético no uniforme.*

hijo 1	$(\alpha x_1 + (1-\alpha) y_1, ..., \alpha x_n + (1-\alpha) y_n)$
hijo 2	$((1-\alpha) x_1 + \alpha y_1, ..., (1-\alpha) x_n + \alpha y_n)$

Fig. 1.20 Cruce aritmético

1.5.1.4.3 Operadores de Mutación.

La misión del operador de mutación es producir nuevas soluciones a partir de la modificación de un cierto número de posiciones de una solución ya existente. El objetivo es por lo tanto introducir variabilidad dentro de la población. En muchos casos los individuos generados por mutación serán peores y se descartarán en el proceso de selección, pero en otros casos pueden ofrecer una mayor adaptación al medio y serán favorecidos en la selección.

Solución inicial	*1 0 1 0 1 1*	*2 3 4 1 6 5*
Nueva solución	*1 0 1 0* ***0*** *1*	***1*** *3 4* ***2*** *6 5*
	Cambio del valor de cada bit con una probabilidad del 10%	*Cambio de la posición de dos elementos con probabilidad del 10%*

Fig. 1.21 Ejemplos de mutación.

El operador de mutación está considerado como un operador menor; sin embargo, es necesario para conducir la búsqueda hacia regiones cualitativamente diferentes. Junto con la selección, pueden considerarse un conjunto de operadores canónicos, ya que

cualquier individuo puede ser alcanzado a partir de una secuencia de mutaciones más o menos larga. A pesar de que con la mutación puede resolverse cualquier tipo de problema, es necesario utilizar otros operadores que aceleren la convergencia, ya que la mutación es un operador de paso corto.

Existen distintas implementaciones de la mutación (Malumbres L. (1.994)):

- **Puntual**. Cada gen (bit) de un cromosoma es independiente del resto. La mutación de un gen es un proceso markoviano e independiente del proceso markoviano del resto de los genes. Este es el sistema más utilizado para problemas de aplicación general.

- **Dirigida.** Se imponen restricciones a una mutación puntual, como puede ser la de impedir más de una mutación por cromosoma, o permitir únicamente aquellas mutaciones que no separan mucho el cromosoma resultante del original.

- **Exponencial.** Se supone codificación binaria y se realiza una agrupación de bits previa a la aplicación del operador. Cada grupo de bits representa un valor. El objetivo de este tipo de mutación es limitar la amplitud del grado de variabilidad admitido. Para ello se favorecen las mutaciones en los bits de menor peso, a la vez que se dificultan las que pueden suceder en bits de mayor peso dentro de cada grupo. Se pretende evitar las variaciones bruscas de valores.

- **Decimal.** Se usa para trabajos con codificación en base diez. Consiste en alterar el dígito elegido por medio de la suma (o resta) de un incremento tal, que no incluya modificaciones bruscas pero permita variabilidad.

- **Real.** Los alfabetos finitos y reducidos facilitan los análisis teóricos y permiten la construcción de operadores genéticos elegantes. Pero en ocasiones puede resultar más eficiente la utilización de alfabetos de cardinal más alto, representando cada componente de un cromosoma como un número real por ejemplo.

1.5.1.4.4 Otros Operadores.

Otros operadores utilizados en general para casos más específicos son los siguientes (Malumbres L. (1.994)):

- **Recombinación**. Hace énfasis en la proximidad de los diferentes genes dentro de un cromosoma. Útil para resolver problemas donde es importante esta propiedad, tales como el del viajante. El procedimiento usado para implementar este operador sería el siguiente:

Procedimiento Recombinación.

Crear una tabla de adyacentistas. En ella para cada posible valor se sitúan todos los valores que están en algunos de los progenitores junto a él.

Seleccionar aleatoriamente un elemento de comienzo de cualquiera de los progenitores. Es el primero del nuevo individuo.
Seleccionar un elemento adyacente al elegido de la tabla.
Eliminar de la tabla este elemento. Se añade como nuevo gen del individuo que se crea.
Repetir hasta que se obtenga una secuencia válida.

- **Inversión**. Se utiliza cuando se cree que el algoritmo se ha estancado para reordenar los elementos de un cromosoma de otra forma. Esto se produce cuando la exploración del espacio de soluciones se puede dar por terminada, y la mejor solución obtenida no es la mejor posible; se ha caído en un mínimo local y existe otra área de soluciones donde se puede encontrar una mejor. El procedimiento sería el siguiente:

Procedimiento Inversión.
Seleccionar un progenitor.
Elegir dos puntos al azar del cromosoma; si son el primero y el último la inversión es total.
Invertir el orden de los genes comprendidos entre esos dos puntos.

---

padre 1 (1 **1 1 0 0 1 0** 0)

hijo 1 (1 **0 1 0 0 1 1** 0)

Fig. 1.22 Inversión.

1.5.1.5 Reemplazo y Condición de Parada.

REEMPLAZO.

En dicha fase se sustituyen individuos de la población original por otros nuevos *mejores (supervivencia de los más adaptados)*. Para "borrar" los cromosomas existen tres sistemas fundamentales:

- Cuando el número de individuos generados llega a un cierto límite prefijado, los individuos no elegidos en una selección anterior desaparecen.

- Cada vez que se genera un nuevo individuo, se elige otro de la población que se hace desaparecer. La elección del individuo a borrar se hará en función de la bondad de los individuos y en alguna forma será inversa al sistema de selección.

- Cada vez que se crea un nuevo cromosoma, se elige otro al azar que es eliminado, independientemente de su evaluación de bondad.

CONDICIÓN DE PARADA.

Si el óptimo es conocido, se parará cuando se alcance dicha solución o una muy cercana; en caso contrario, el criterio de parada

puede venir determinado por:

- tiempo de ejecución,
- número de generaciones,
- pérdida de diversidad,
- convergencia (no hay mejora sustancial en las últimas k iteraciones).

1.5.1.6 Aplicaciones a Problemas de Rutas.

La representación de las soluciones se realiza mediante una cadena de enteros donde cada entero denota un vértice o punto de visita determinado. La posición que ocupa cada vértice indica el orden en el que será visitado.

Se deben desarrollar operadores de cruce y mutación especiales para producir nuevas soluciones. Se han propuesto algunos operadores en la literatura, por ejemplo en el trabajo de Potvin J.Y. (1.996). Oliver I.M. y otros (1.987) trabajan con el operador de cruce denominado *order crossover* (OX): se seleccionan dos puntos de corte aleatoriamente y la cadena situada entre esos dos puntos se asigna a la nueva solución o solución hija. El resto de posiciones se rellenan de una en una, comenzando después del segundo punto de corte, considerando los vértices en el orden que se encuentran en la segunda solución (padre), evitando duplicaciones. Se puede crear una segunda nueva solución invirtiendo el papel de los padres. Con el OX, las nuevas

soluciones tienden a heredar el orden relativo de puntos de visita de las dos soluciones originales (padres). Un ejemplo ilustrativo se representa en la figura 1.23.

Padre 1	:	0	1	2	3	4	5
Padre 2	:	**0**	**4**	**3**	**2**	**5**	**1**
Hijo 1	:	-	-	-	3	4	-
Hijo 1	:	**0**	**2**	**5**	3	4	**1**

Fig. 1.23 Order Crossover.
Tomada de Gendreau M. y otros (1.999).

Otros operadores de cruce tienden a preservar la posición de los vértices (Goldberg D.E. y Lingle R. (1.985)), o los arcos de las soluciones padres (Whitley D. y otros(1.989)).

Con respecto a la mutación, se han diseñado sencillos operadores de intercambio o de *eliminación-reinserción* (RAR) que mueven uno o dos puntos de visita a otra posición dentro de la cadena. En la figura 1.24 se ilustra la mutación de una solución mediante el movimiento del vértice denotado con el número dos.

Hijo 1	:	0	1	2	3	4	5
Hijo 1	:	0	1	3	4	2	5

Fig. 1.24 Mutación por RAR.
Tomada de Gendreau y otros (1.999).

Van Breedam A. (1.996) compara un GA, con un SA y un TS desarrollados anteriormente, utilizando diferentes tipos de problemas de rutas de vehículos incluido el VRP con restricciones de capacidad. Una solución para el VRP está formada por múltiples rutas; la representación es larga y contiene varias veces el punto origen indicando el final de una ruta y el comienzo de otra. El autor aplica un operador de cruce y un operador de mutación, hasta conseguir un determinado número de soluciones factibles. Además, usa un operador de descenso local, basado en cuatro tipos diferentes de movimientos de intercambio y lo aplica sólo a la mejor solución de la población. Lo usa para su test de 15 problemas con 100 vértices, de los que los seis primeros son VRPs con restricciones de capacidad; demuestra que el uso del operador de descenso local tiene un impacto positivo en la ejecución de GA. Las soluciones encontradas son de calidad comparable en todos los casos: GA, SA y TS (desarrollados por el mismo autor). El tiempo de computación era superior en el caso de GA.

Otra aplicación para el VRP con restricciones de capacidad y de tiempo se desarrolla en los trabajos de Schmitt L.J. (1.994) y (1.995). Una interesante característica de estos trabajos es que utiliza el método de *ruta primero, cluster segundo* (se crea una ruta que incluye todos los puntos de visita del problema y luego se divide en rutas menores para conseguir una solución factible, Beasley J.E. (1.983)), lo que permite usar la representación clásica: secuencia única de puntos sin separación. La primera ruta de la solución comienza en el primer punto de la cadena de puntos. En

Según se comenta en Gendreau y otros (1.999), experiencias previas realizadas con GAs para resolver problemas de optimización combinatoria han demostrado que la estructura clásica de los algoritmos, con la mutación como segundo operador para perturbar ligeramente las soluciones, no ofrecía resultados competitivos. Para ser efectivos deben realizarse híbridos con métodos de búsqueda local, e incluso con Búsqueda Tabú.

Es escasa la literatura referente a desarrollos de GAs para el VRP. No sucede lo mismo con aplicaciones para el TSP (Potvin J.Y. (1.996)) o con variantes de VRP con ventanas de tiempo (VRPTW) y restricciones de precedencia. La presencia de restricciones complicadas, ha entorpecido últimamente el desarrollo de métodos efectivos de resolución de problemas, lo que ha favorecido claramente a los GAs dada su relativa robustez para proveer soluciones competitivas cuando intervienen restricciones complejas.

Entre las implementaciones más efectivas de GAs para el VRPTW se encuentran las realizadas por Potvin J.Y. y Bengio S. (1.996) y por Thangiah S.R. (1.993).

El trabajo hecho para el VRP con restricciones de capacidad por Blanton J.L. y Wainwright R.L. (1.993), que incluye las variantes de restricciones de tiempo o distancia, se realizó principalmente para evaluar el impacto que diferentes componentes o parámetros de GA tienen sobre la búsqueda.

Otros trabajos referenciados en la literatura son los siguientes:

- Bean J.C.(1.994),
- Benyahia I. y Potvin J.Y. (1.995),
- Hjorring C. (1.993),
- Leclerc F. y Potvin J.Y. (1.997),
- Thangiah S.R. (1.991),
- Thangiah S.R. y Nygard K.E. (1.992),
- Thangiah S.R., Nygard K.E. y Juell P.L., GIDEON (1.991)
- Wren A. y Wren D.O. (1.995).

1.5.2 Algoritmos Meméticos.

Los *Algoritmos Meméticos* (MAs) son procedimientos basados en población para la búsqueda heurística en problemas de optimización. Se han mostrado como técnicas más rápidas que los tradicionales Algoritmos Genéticos para algunos tipos de problemas. En 1.989 Pablo Moscato identificó varios autores que como él, habían introducido heurísticos para mejorar las soluciones antes de recombinarlas. Particularmente en el campo de los GAs varios autores fueron introduciendo conocimiento del problema de varios modos. En Moscato P. (1.989) se introdujo la denominación de *memetic algorithms* para este tipo de metaheurísticos. Según Minsky M. (1.994) "*... son modos de añadir conocimiento a la búsqueda basada en población.*"

el momento en el que las restricciones de capacidad o de duración son sobrepasadas, se inicia otra ruta. De esta forma, cada cadena de puntos puede decodificarse en una solución factible. En el trabajo de Schmitt L.J.(1.995) se utiliza un operador de cruce OX y un operador de mutación de permuta, en el que dos puntos de visita seleccionados aleatoriamente intercambian sus posiciones. Para mejorar el algoritmo, este operador de mutación se reemplaza por un método de búsqueda local 2-óptimo (Croes G. (1.958), Lin S. (1.965)), aplicándolo con una probabilidad del 0.15 a cada nueva solución. La comparación de los resultados obtenidos para los problemas de Christofides N. y otros (1.979) muestra mejores resultados que los aportados por Clarke G. y Wright J.W. (1.964) pero peores que los obtenidos por heurísticos de construcción sencillos combinados con procedimientos de mejora. El tiempo de computación en todos los casos es superior.

En base a los escasos resultados obtenidos en estos trabajos Gendreau M. y otros (1.999) afirman que los GAs no son aún competitivos para VRP, sobre todo teniendo en cuenta los desarrollos de TS. Los esfuerzos de los GAs se han centrado en el TSP o en variantes del VRP con ventanas de tiempo. No obstante el éxito obtenido en esta clase de problema, parece indicar que con un mayor trabajo en el VRP existe la posibilidad de llegar a realizar implementaciones competitivas.

Según comenta el propio Pablo Moscato en Corne D. y otros (1.999), una característica clave que presentan la mayoría de las implementaciones de MAs es el uso de una búsqueda basada en población que tiene la intención de usar todo el conocimiento disponible sobre el problema. Este conocimiento se incorpora en forma de heurísticos, algoritmos de aproximación, técnicas de búsqueda local, operadores especializados de recombinación, métodos exactos truncados, y algunas otras formas. Su éxito puede ser probablemente explicado como la consecuencia directa de la sinergia de diferentes enfoques de búsqueda incorporados.

Básicamente combinan heurísticos de búsqueda local con operadores de cruce, por lo que algunos investigadores les han visto como Algoritmos Genéticos Híbridos. La primera implementación de Pablo Moscato junto con Norman M.G. (Norman M.G. y Moscato P. (1.989)) fue un híbrido de GA y SA. Combinaciones con heurísticos constructivos o métodos exactos pueden también pertenecer a esta clase de metaheurísticos.

Han recibido también el nombre de *Algoritmos Genéticos Paralelos* (PGAs) por su mejor adaptabilidad a computadores paralelos MIND y sistemas de computación distribuidos como aquellos compuestos por redes de estaciones de trabajo. Los PGAs fueron desarrollados por Mühlenbein y permiten que los individuos en la población mejoren su fitness mediante mejoras iterativas (conocidas como *hillclimbing*).

El uso de operadores de búsqueda local también se encuentra en los trabajos de Montana D.J. y Davis L. (1.989), Huntley C.L. y Brown D.E. (1.991), Ulder N.L.J. y otros (1.991) quienes han denominado al método *Búsqueda Local Genética.*

El método está ganando amplia aceptación, en particular en los problemas de optimización combinatoria donde se han solventado óptimamente ejemplos de gran tamaño y donde otros metaheurísticos han fallado.

1.5.3 Búsqueda Dispersa (Scatter Search).

SS está basado en formulaciones originalmente propuestas en los 60 para reglas de decisión combinatoria y problemas con restricciones en el contexto de la programación entera. Ha demostrado obtener resultados prometedores para resolver problemas de optimización combinatorios y no-lineales. SS puede implementarse de múltiples formas y ofrece numerosas alternativas para explotar sus ideas fundamentales. Usa estrategias para combinar soluciones que han demostrado ser efectivas en diferentes tipos problemas.

1.5.3.1 Bases.

SS se puede clasificar dentro de los *algoritmos evolutivos* o también denominados *algoritmos poblacionales*, que construyen soluciones a partir de combinaciones de otras. Se caracteriza por el uso de un conjunto de soluciones, denominado *Conjunto de Referencia*, que es actualizado durante el proceso. El modo en el que combina soluciones y actualiza el conjunto de soluciones de referencia usadas para combinar conjuntos, aparta esta metodología de otros enfoques basados en población. Pablo Moscato incluye SS dentro de los MA (Algoritmos Meméticos) y comenta en el capítulo 14 de Corne D. y otros (1.999) que es un caso llamativo de MA que difícilmente se adecua a la noción intuitiva de MA.

Como se comenta en los trabajos de Glover F. (1.998) y Campos V. y otros (1.999) el enfoque de combinación de soluciones para crear nuevas soluciones se originó en los 60. La estrategia combinatoria se diseñó con la confianza de que la información sería explotada más efectivamente de forma integrada que tratándola aisladamente (Crowston W.B. y otros (1.963); Fisher H. y Thompson G.L. (1.963)).

En programación entera y no lineal, se desarrollaron procedimientos asociados para combinar restricciones. En este caso se introdujeron pesos no negativos para crear nuevas restricciones

de desigualdad llamadas *surrogate constraints* (restricciones "sucedaneas") (Glover F. (1.965), (1.968)).

La principal función de las *surrogate constraints* fue aportar modos para evaluar opciones o elecciones que se usarían para crear y modificar las soluciones prueba. Desde su creación se desarrollaron una variedad de procesos heurísticos que emplearon dichas *surrogate constraints*. Como una extensión natural, estos procesos llevaron a la estrategia de combinación de soluciones comentada. La combinación de soluciones, como se manifestó en scatter search, puede interpretarse como el equivalente primal a la estrategia dual de combinación de restricciones.

Scatter search opera en un conjunto de referencia (*reference set, RS*) combinando soluciones para crear unas nuevas. El conjunto de referencia puede evolucionar como se ilustra en la figura 1, cuando se crean nuevas soluciones de una combinación lineal de otras dos o más soluciones. Esta figura asume que el conjunto de referencia original de soluciones consta de los círculos etiquetados como A, B y C. Después una combinación no convexa de las soluciones de referencia A y B crea la solución 1. Mas precisamente se crean un número de soluciones en el segmento definido por A y B; sin embargo sólo la solución 1 se introduce en el conjunto de referencia. De modo similar, las combinaciones convexas y no convexas del conjunto de referencia original y la solución recién

creada, crea los puntos 2, 3 y 4. El conjunto de referencia completo mostrado en la figura 1.25 consta de 7 soluciones.

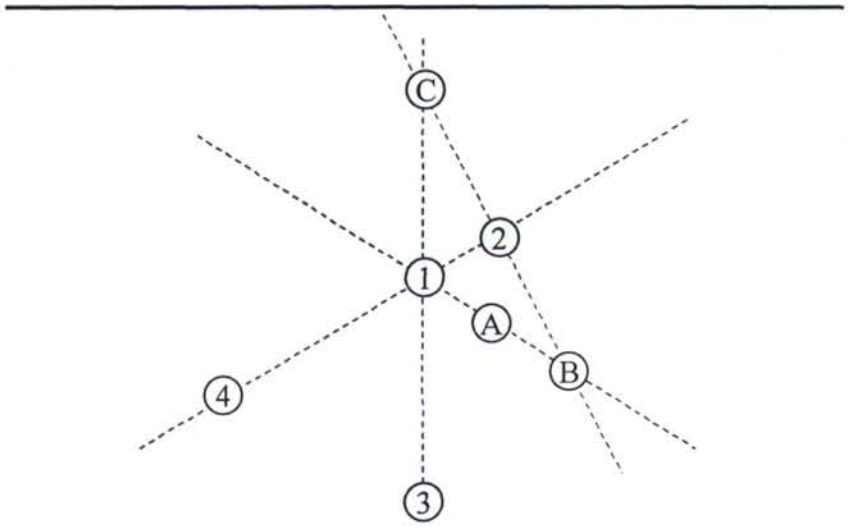


Fig.1.25 Conjunto de referencia; tomado de Laguna (1.999).

El conjunto de referencia de soluciones en búsqueda dispersa tiende a ser pequeño, a diferencia de la población de los algoritmos genéticos. En algoritmos genéticos se eligen aleatoriamente dos individuos de la población y se aplica un mecanismo de cruce o combinación para generar uno o más hijos. Un tamaño de población típico en algoritmos genéticos consta de 100 elementos, que son probados aleatoriamente para crear combinaciones. En contraste, SS elige dos o más elementos del conjunto de referencia de forma sistemática con el propósito de crear nuevas soluciones. Ya que el proceso de combinación considera al menos todos los pares de soluciones del conjunto de referencia en la práctica se

necesita trabajar con conjuntos de pocos elementos. Normalmente el conjunto de referencia en búsqueda dispersa tiene 20 soluciones o menos. En general, si el conjunto de referencia consta de *b* soluciones, el procedimiento examina aproximadamente *(3b-7) b/2* combinaciones de cuatro tipos diferentes (Glover F. (1.998)). El tipo básico consta de combinaciones de dos soluciones; el siguiente tipo combina tres soluciones y así seguimos con cuatro o más soluciones dependiendo del problema. La limitación del campo de búsqueda a un grupo selectivo de tipos de combinación puede usarse como un mecanismo de control del número de combinaciones posibles en un conjunto de referencia dado.

1.5.3.2 Esquema de Scatter Search.

El proceso de búsqueda dispersa, se organiza para: capturar información no contenida por separado en los vectores o soluciones originales, y aprovechar métodos de solución heurísticos auxiliares para evaluar las combinaciones producidas y generar nuevos vectores (Glover F. (1.998)).

El esquema descrito en Laguna M. (1.999) consta de las siguientes 4 fases:

1. Generar un conjunto inicial de soluciones que garanticen un nivel crítico de diversidad y aplicar procedimientos heurísticos diseñados para el problema considerado con el fin de mejorar

dichas soluciones. Seleccionar un subconjunto de las "mejores" soluciones que constituirán el conjunto de referencia. El concepto de mejores en este paso no está limitado a una evaluación dada exclusivamente por el valor de la función objetivo. Una solución puede ser añadida al conjunto de referencia si la diversidad del conjunto mejora incluso cuando el valor de la función objetivo de tal solución es inferior al de otras soluciones que compiten por la admisión en el conjunto de referencia.

2. Crear nuevas soluciones como combinaciones estructuradas de subconjuntos de soluciones de referencia actuales. Las combinaciones estructuradas son:

a) Elegidas para producir puntos tanto dentro como fuera de las regiones convexas abarcadas por las soluciones de referencia.

b) Modificadas para producir soluciones aceptables. Por ejemplo, si una solución se obtiene mediante una combinación lineal de dos o mas soluciones, puede aplicarse un proceso de acotamiento generalizado que produce valores enteros para componentes (vectores) enteros-restringidos. Hay que resaltar que una solución aceptable puede o no ser factible con respecto a otras restricciones del problema.

3. Aplicar el proceso heurístico usado en el paso 1 para mejorar las soluciones creadas en el paso 2. Esos procesos heurísticos

deben poder operar en soluciones infactibles y pueden o no producir soluciones factibles.

4. Extraer una colección de las mejores soluciones mejoradas del paso 3 y añadirlas al conjunto de referencia. El concepto de mejores es de nuevo amplio: el valor objetivo es por lo tanto, uno entre varios criterios para evaluar el mérito de los puntos nuevamente creados. Repetir los pasos 2, 3 y 4 hasta que el conjunto de referencia no varíe. Diversificar el conjunto de referencia mediante la reinicialización desde el paso 1. Parar cuando se alcance una iteración límite especificada.

La primera característica notable en búsqueda dispersa es que sus combinaciones convexas se diseñan con el objetivo de crear centros ponderados de subregiones seleccionadas. Además se añaden combinaciones no convexas que proyectan nuevos centros en regiones que son externas a las soluciones de referencia originales (por ejemplo, solución 3 en la figura 1.25). Los patrones de dispersión creados por tales centros y sus proyecciones externas se han encontrado de utilidad en varias áreas de aplicación.

Otra característica es el uso de mecanismos de mejora que puedan operar en soluciones infactibles, eliminando las restricciones que obligan a las soluciones a ser factibles para incluirlas en el conjunto de referencia.

Los siguientes principios resumen los fundamentos de la metodología de SS (Glover F. (1.998)):

- Existe información útil acerca de soluciones óptimas, contenida en una colección convenientemente diversa de soluciones élite.

- Cuando las soluciones se combinan para explotar tal información, es importante proveer mecanismos capaces de construir combinaciones que extrapolen más allá de las regiones abarcadas por las soluciones consideradas. De igual forma es también importante incorporar heurísticos para mejorar estas soluciones combinadas. El propósito de esos mecanismos de combinación es incorporar diversidad y calidad.

- Teniendo en cuenta múltiples soluciones simultáneamente, como base para crear combinaciones, mejora la oportunidad de explotar información contenida en la unión de soluciones élite.

Según las observaciones y principios comentados, los métodos necesarios para implementar SS serían los siguientes:

1) Un **Método de Generación de Diversidad** para generar una colección de soluciones prueba diversas, usando una solución prueba arbitraria (solución semilla) como input.

2) Un **Método de Mejora** que transforme una solución en una o más soluciones mejores. No se requiere que las soluciones input

ni output sean factibles, aunque se espera que la solución output si lo sea.

3) Un Método de **Actualización del Conjunto de Referencia** que construya y mantenga el conjunto de referencia con las *b* mejores soluciones encontradas (donde *b* es normalmente pequeño, no más de 20). El conjunto de referencia se divide en dos partes, subconjunto de calidad y subconjunto de diversidad.

4) Un **Método de Generación de Subconjuntos** del conjunto de referencia. Los elementos del subconjunto se combinan para generar nuevas soluciones.

5) Un Método de **Combinación de Soluciones** para producir nuevas soluciones.

1.5.3.3 Aplicaciones de SS a Problemas de Rutas.

Corberán A. y otros (2.000), aplican SS al transporte escolar; los resultados obtenidos serán comentados en el capítulo cuatro de este trabajo.

1.6 Otros Metaheurísticos.

Los métodos de optimización encuadrados en las dos amplias clasificaciones vistas en los puntos anteriores (metaheurísticos basados en búsqueda local y metaheurísticos basados en búsqueda poblacional), tienen una idea común: un método denominado *hillclimbing*, ilustrado en la figura 1.26.

1. **Empezar**: Generar y Evaluar una solución "actual" inicial *s*.
2. **Ejecutar**: Cambiar *s* produciendo *s'*, y evaluar *s'*.
3. **Renovar**: Si *s'* es mejor que *s*, entonces reemplazar *s* por *s'*.
4. **Repetir**: Si no se cumple el criterio de parada, volver a 2.

Fig. 1.26 Algoritmo Hillclimbing sencillo. Tomado de Corne D. y otros (1.999).

En el caso de los métodos basados en población, la noción de solución única se reemplaza por un conjunto de diferentes soluciones y en función de ello se adaptan los pasos 1, 2 y 3.

En algunos *nuevos metaheurísticos* es difícil o inapropiado implementar las ideas o pasos de la figura anterior; esto sucede por ejemplo con Colonia de Hormigas.

1.6.1 Colonias de Hormigas.

Este método propuesto por Dorigo M. y otros (1.996) es un ejemplo, como el Temple Simulado, Redes Neuronales y otros, del afortunado uso de metáforas naturales para diseñar un algoritmo de optimización. En este caso se aprecia con claridad como las soluciones generadas previamente afectan a las soluciones que se generan en el futuro.

Las hormigas reales son capaces de encontrar el camino más corto desde una fuente de comida al hormiguero sin usar señales visuales. También son capaces de adaptarse a cambios del entorno, por ejemplo, encontrando un nuevo camino más corto cuando el anterior ya no es factible debido a un obstáculo interpuesto.

Las hormigas se mueven en línea recta que conecta la fuente de comida con su hormiguero. El medio básico que tienen para formar y mantener la línea es un reguero de pheromone: las hormigas depositan determinada cantidad de esta sustancia mientras caminan, y cada hormiga prefiere (probabilísticamente) seguir una dirección rica en pheromone. Cuando aparece de forma imprevista un obstáculo no esperado que interrumpe el camino inicial, aquellas hormigas que están justo en frente del obstáculo no pueden continuar siguiendo el reguero de pheromone: tienen que elegir entre girar a la izquierda o a la derecha. En esta situación podemos esperar que la mitad de las hormigas elijan una cosa y la

otra mitad otra. Una situación muy similar puede encontrarse en el otro lado del obstáculo. Es interesante destacar que aquellas hormigas que eligen por casualidad el camino corto, reconstruyen más rápidamente el reguero de pheromone interrumpido, en comparación con aquellas que eligen el camino largo. Así que el camino corto recibirá mas cantidad de pheromone por unidad de tiempo y en cada turno un mayor número de hormigas elegirá el camino corto. Debido a este proceso retroalimentado positivo, todas las hormigas elegirán rápidamente el camino más corto. El aspecto más interesante de este proceso autocatalítico es que encontrar el camino más corto alrededor del obstáculo parece ser una propiedad emergente de la interacción entre la forma del obstáculo y la conducta distributiva de las hormigas: aunque todas las hormigas se mueven aproximadamente a la misma velocidad y depositan aproximadamente la misma cantidad de pheromone, es un hecho que lleva más tiempo recorrer el lado largo del obstáculo que el corto; la pheromone se acumula más rápidamente en el lado corto y la preferencia de las hormigas por el camino con más alto nivel de pheromone permite una acumulación más rápida en el camino corto.

A continuación se describen algunas ideas de una aplicación al TSP propuesta por Dorigo M. y Gambardella L.M. (1.997-a).

Una hormiga artificial es un agente que se mueve de una ciudad a otra en un grafo de TSP. Elige la ciudad a la que moverse (o arco que añade a su ruta) con una probabilidad proporcional al reguero

acumulado y la distancia del arco que se añade. Las hormigas artificiales prefieren probabilísticamente ciudades que están conectadas por arcos con mucho reguero de pheromone y que están próximas. Inicialmente, *m* hormigas artificiales se colocan en ciudades seleccionadas aleatoriamente. En cada iteración se mueven a nuevas ciudades y modifican el reguero de pheromone de los arcos usados - esto se *denomina actualización de reguero local*. Cuando todas las hormigas han completado una ruta, la hormiga que hace la ruta más corta modifica los arcos pertenecientes a su ruta - se *denomina actualización de reguero global*- añadiendo una cantidad de reguero de pheromone que es inversamente proporcional a la longitud de la ruta.

Hay tres ideas de la conducta de las hormigas que se transfieren a la colonia de hormigas artificiales:

1. la preferencia por caminos con alto nivel de pheromone,
2. el alto ratio de crecimiento de la cantidad de pheromone en los caminos cortos, y
3. el reguero mediador de comunicación entre las hormigas.

Se ha dado a las hormigas artificiales algunas capacidades que no tienen sus colegas naturales pero que se ha observado que son aptas para su aplicación al TSP: las hormigas artificiales pueden determinar la distancia a la que están las ciudades y están dotadas con una memoria de trabajo M_k usada para memorizar las ciudades ya visitadas (la memoria de trabajo está vacía al comienzo de cada

nueva ruta y se actualiza después de cada paso de tiempo añadiendo las nuevas ciudades visitadas).

La actualización local del reguero tiene como fin evitar que un arco muy atractivo sea elegido por todas las hormigas: Cada vez que un arco es elegido por una hormiga su cantidad de pheromone se cambia aplicando la fórmula de actualización local. La evaporación del reguero de pheromone en el mundo de las hormigas reales se traslada a la colonia de hormigas artificiales en forma de actualización local del reguero.

Se pueden encontrar aplicaciones problemas de rutas referenciadas en la página web que mantiene Marco Dorigo, relativa a Colonia de Hormigas (http://iridia.ulb.ac.be/~mdorigo/ACO/ACO.html#ACO):

- Bullnheimer B., Hartl R.F. y Strauss C. (1.998)
- Bullnheimer B., Hartl R.F. y Strauss C. (1.999) y
- la tesis doctoral de Bullnheimer B. (1.999).

Información más detallada sobre este metaheurístico y sus aplicaciones a diversos campos, entre ellos el VRP está disponible en Corne D. y otros (1.999), páginas 9-76.

CAPÍTULO II

EXPERIENCIAS CON BÚSQUEDA LOCAL Y ENTORNOS: RESULTADOS EMPÍRICOS

2.1 Construcción de entornos:
  2.1.1 Variante de los intercambios r-óptimos (vecindarios tipo Or).
  2.1.2 Intercambio entre dos rutas limitado (vecindarios tipo Gendreu-Clark).
  2.1.3 Intercambio entre dos rutas generalizado (vecindaros tipo Taillard).

2.2 Diseño de los algoritmos.

2.3 Comparación de Búsqueda Local con diferentes entornos e instancias simuladas.

2.4 Comparación de criterios de elección de nuevas soluciones (primer descenso, mejor descenso).

2.5 Ahorro de tiempo de computación mediante el uso de la Búsqueda Local Rápida.
  2.5.1 Planteamiento inicial y resultados computacionales.
  2.5.2 Modificación en la Búsqueda Local Rápida.

2.6 Análisis de los resultados.

2.1 Construcción de Entornos.

En los problemas de optimización combinatoria siempre han tenido gran importancia los Algoritmos de Búsqueda Local dentro de las técnicas heurísticas; esto ha sido así especialmente a partir de la aparición de muchas de las recientes técnicas Metaheurísticas que, como se ha indicado anteriormente, en buena parte están basadas en el uso de movimientos vecinales o por entornos. En este apartado se analizarán experiencias relacionadas con diferentes tipos de entornos que se aplicarán al VRPTW Mixto[1] (carga y descarga). El primero de estos entornos, propuesto por Or I. (1.976), es una variante de los *intercambios r-óptimos*. En segundo lugar se analizarán tres tipos de entornos a los que hemos denominado *Vecindarios de Intercambio entre dos rutas limitado*; aparecen en los trabajos de Gendreu M. y otros (1.991) y Clarke G. y Wright J.W. (1.964). Por último se estudian entornos, también de intercambio entre dos rutas de la solución, pero más amplios que los anteriores, aumentando consecuentemente el tiempo de computación del algoritmo. Este tipo de entorno fue propuesto en el trabajo de Taillard E.D. y otros (1.997).

En este trabajo las soluciones para el VRP se representan según el tipo de entorno que se aplique, de dos formas diferentes:

---

[1] VRPTW Mixto: Descrito en los apéndices 1.3 y 1.5.

- para el primer tipo, como secuencias de puntos de visita en el que los 1 indican la finalización de una ruta y el comienzo de la siguiente (obviamente el primer y último elemento siempre serán 1); por ejemplo:

  1 -2 -5 -9 -6 -1 -3 -4 -7 -8 -10 -1 -11 -12 -13 -1

- para los dos siguientes tipos, como un conjunto formado por diferentes rutas; el ejemplo anterior se representaría de la siguiente manera:

  - primera ruta: 1 - 2 - 5 - 9 - 6 - 1
  - segunda ruta: 1 - 3 - 4 - 7 - 8 - 10 - 1
  - tercera ruta: 1 - 11 - 12 - 13 - 1

2.1.1 Variante de los intercambios r-óptimos (vecindario Tipo Or).

Se van a definir como soluciones vecinas las obtenidas por el método de intercambio propuesto por Or I. (1.976) que sean factibles (Or-intercambios o vecindarios tipo Or). Como ya se ha comentado, el método de *intercambio de Or* es una variante de los conocidos intercambios *r-óptimos* desarrollados por Lin S. (1.965) y Lin S. y Kernighan B.W. (1.973) para el TSP simétrico, aunque este método se puede utilizar también en problemas asimétricos.

Su eficacia para el TSP ha sido contrastada en trabajos como el de Nurmi K. (1.991).

La representación de la solución, tal y como se ha comentado anteriormente, se realiza considerando una única secuencia de puntos. Or I. (1.976) 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.

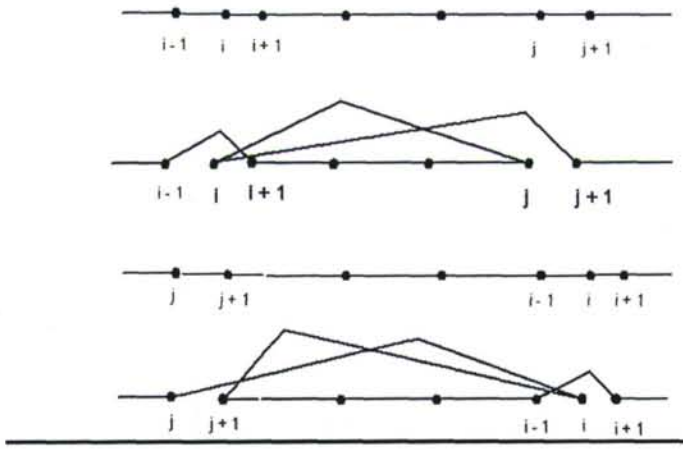


Fig. 2.1 Posible recolocación del elemento i hacia adelante y hacia atrás entre j y j+1

En el cálculo del valor de la función objetivo de la nueva solución, se ha de tener en cuenta la variación en la distancia

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

recorrida (diferencia entre distancia de los arcos eliminados y los que se incorporan), y el posible cambio en el tipo de vehículo requerido en las rutas implicadas.

En este caso se sigue la misma idea, pero sólo consideraremos recolocaciones hacia adelante y no pondremos límite al tamaño de la cadena a recolocar (salvo el determinado por el tamaño del problema). Las recolocaciones hacia atrás no son consideradas principalmente por dos motivos: suponen un mayor tiempo de computación y no aportan mejoras sustanciales, más aún si tenemos en cuenta que cualquier solución puede alcanzarse sólo con recolocaciones hacia adelante. Se debe chequear la factibilidad de cada posible recolocación respecto a las restricciones del problema. Para cada $s \in S$ se denotará $N^k_1(s)$ al conjunto de soluciones factibles generadas mediante dicho movimiento, siendo k el tamaño de la cadena a recolocar. En la figura 2.2 se ilustra la recolocación de una cadena de k elementos, comenzando en i entre j y j+1.

Sea m el número de rutas que componen una solución $s \in S$, el número de puntos que contiene dicha solución, considerada como una única cadena, es de n+m. Como cada intercambio viene determinado por la posición de los puntos i y j y por el tamaño k de la cadena a recolocar, se tiene que para k finito, el número de intercambios a considerar para obtener $N^k_1(s)$ es de $O(n^2)$, y para $N^\infty_1(s)$ es de $O(n^3)$.

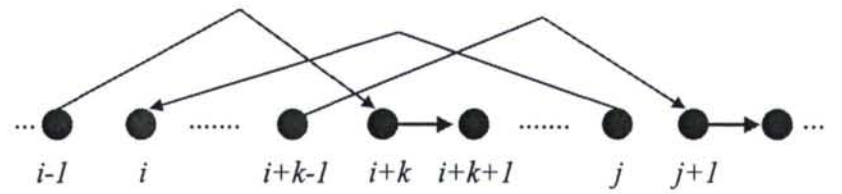


Fig. 2.2 Recolocación hacia adelante de una cadena de k elementos.

En este trabajo se aplicarán los *intercambios de Or* tanto a soluciones completas considerándolas como una única cadena de puntos, como a rutas particulares dentro de una solución.

2.1.2 Entornos de intercambio entre dos rutas limitado (vecindario tipo Gendreau-Clark).

El segundo tipo de entornos considera 3 formas de intercambio entre 2 rutas diferentes de una misma solución.

Tipo I: Intercambio del punto i de la ruta r con el punto i' de la ruta r':

- Eliminación de los arcos $(i\text{-}1, i)$, $(i, i\text{+}1)$, $(i'\text{-}1, i')$, $(i', i'\text{+}1)$.
- Incorporación de los arcos $(i\text{-}1, i')$, $(i', i\text{+}1)$, $(i'\text{-}1, i)$, $(i, i'\text{+}1)$.

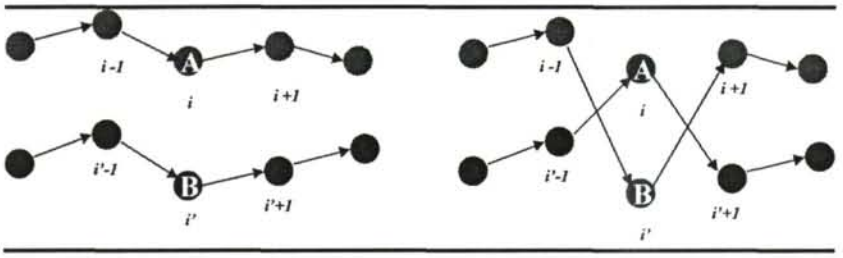


Fig. 2.3 Intercambio de un punto de una ruta por otro de otra.

Tipo II: Inserción del punto i de la ruta r entre los puntos i' e $i'+1$ de la ruta r':

- Eliminación de los arcos $(i\text{-}1, i)$, $(i, i+1)$, $(i', i'+1)$
- Incorporación de los arcos (i', i), $(i, i'+1)$, $(i\text{-}1, i+1)$

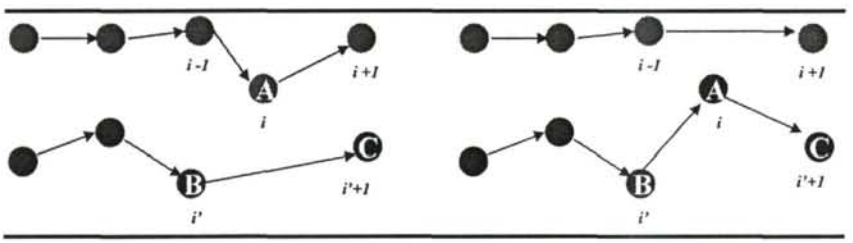


Fig. 2.4 Inserción de un punto de una ruta entre otros dos de otra.

Tipo III: Cruce de las rutas r y r' por los elementos i e i' según la figura 2.5:

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

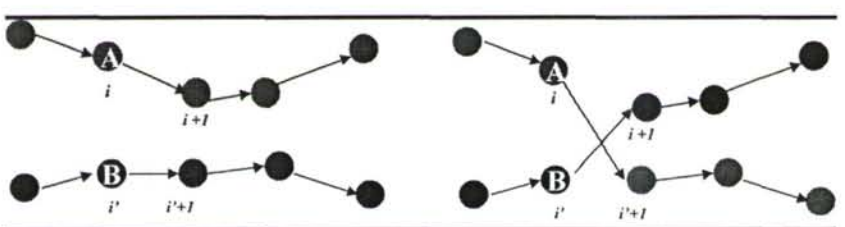


Fig. 2.5 Cruce de dos rutas por dos de sus elementos.

Los dos primeros tipos de entornos aparecen en el trabajo de Gendreu M. y otros (1.991) y Osman I.H. (1.993); el tercero aparece en el trabajo de Kilby P. y otros (1.997), en el que se utilizan también los dos primeros, y se basa en algunas de las ideas propuestas por Clarke G. and Wright J.W. (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 denota a los conjuntos de soluciones factibles generadas por cada uno de estos 3 tipos de intercambios por $N^1_2(s)$, $N^2_2(s)$ y $N^3_2(s)$ respectivamente; así mismo se denota por $N_2(s)$ a la unión de estos 3 conjuntos.

2.1.3 Entornos de intercambio entre dos rutas generalizado (vecindarios Tipo Taillard).

El tercer tipo de entornos, propuesto como ya se ha dicho por Taillard E.D. y otros (1.997), considera intercambios de cadenas de puntos de tamaños k y k' entre dos rutas diferentes (CROSS intercambios) según muestra la figura 2.6:

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

- Incorporación de *(i, i'+1), (i+k, i'+k'+1), (i'+k',i+k+1), (i', i+1).*

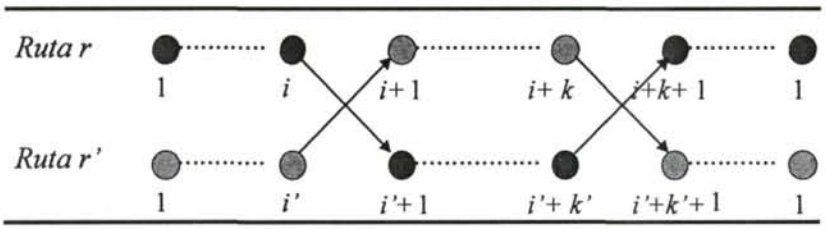


Fig. 2.6 Solución que resulta del intercambio de cadenas de puntos entre dos rutas diferentes.

Obsérvese que estos intercambios generalizan los descritos en el apartado anterior:

- si $k = k' = 1$, se tiene un intercambio tipo I;
- si $k = 1$ y $k' = 0$ intercambio tipo II;
- si $i+k+1$ e $i'+k'+1$ coinciden con el final de la ruta (punto 1) intercambio tipo III.

Para cada $s \in S$ se denota por $N^k_3(s)$ a los conjuntos de soluciones factibles generadas por estos CROSS intercambios de cadenas de a lo sumo k elementos; análogamente se denota por $N^\infty_3(s)$ el conjunto de soluciones factibles generadas por todos estos CROSS intercambios. En nuestro caso se utilizarán vecindarios $N^\infty_3(s)$, de tal forma que k se referirá al tamaño de la cadena en los *intercambios de Or*.

El chequeo de la factibilidad y la valoración de cada intercambio en cualquiera de los vecindarios considerados puede

suponer un tiempo de computación excesivo. En los trabajos de Pacheco J. y Delgado C.R. (1.996) y (1.997-b) 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.

2.2 Diseño de los Algoritmos a Usar.

A continuación se muestran en seudocódigo los procedimientos de movimiento vecinal y búsqueda local atendiendo a los diferentes vecindarios o entornos estudiados. En todos ellos se denota como s_0 la solución actual.

Procedimiento Movimiento_Vecinal_Or(k, s_0)

Determinar $s' \in N^k_1(s_0)$ *verificando* $f(s') = min\{f(s) / s \in N^k_1(s_0)\}$

Si $f(s') < f(s_0)$ *entonces hacer* $s_0 = s'$

Procedimiento Búsqueda_Local_Or(k, s_0)

Repetir

Ejecutar Movimiento_Vecinal_Or(k, s_0)

Hasta que $f(s) \geq f(s_0)$, $\forall s \in N^k_1(s_0)$.

Procedimiento Movimiento_Vecinal_Ge(k, s_0)

Determinar $s' \in N_2(s_0)$ *verificando* $f(s') = \min\{f(s) / s \in N_2(s_0)\}$

Si $f(s') < f(s_0)$: *hacer* $s_0 = s'$

Ejecutar Búsqueda_Local_Or(k, r') y

Búsqueda_Local_Or(k, r'')

(*donde r' y r'' son las dos rutas de* s_0 *modificadas para dar lugar a s'*)

Procedimiento Búsqueda_Local_Ge(k, s_0)

Repetir

Ejecutar Movimiento_Vecinal_Ge(k, s_0)

Hasta que $f(s) \geq f(s_0)$, $\forall s \in N^k{}_2(s_0)$

Procedimiento Movimiento_Vecinal_Ta(k, s_0)

Determinar $s' \in N^{\infty}{}_3$ *(s) verificando* $f(s') =$

$\min\{f(s) / s \in N^{\infty}{}_3(s_0)\}$

Si $f(s') < f(s_0)$: *hacer* $s_0 = s'$

Ejecutar Búsqueda_Local_Or(k, r') y

Búsqueda_Local_Or(k, r''')

(*donde r' y r'' son las dos rutas de* s_0 *modificadas para dar lugar a s'*)

Procedimiento Búsqueda_Local_Ta(k, s_0)

Repetir

Ejecutar Movimiento_Vecinal_Ta(k, s_0)

Hasta que $f(s) \geq f(s_0)$, $\forall s \in N^{\infty}{}_3(s_0)$

Obsérvese como en *Búsqueda_Local_Ge(k, s_0)* y *Búsqueda _Local_Ta(k, s_0)* cada movimiento vecinal se compone de un intercambio de cadenas entre dos rutas diferentes y de la mejora de cada una esas dos rutas (véase figura 2.7), siguiendo la idea de Osman I.H. (1.993).

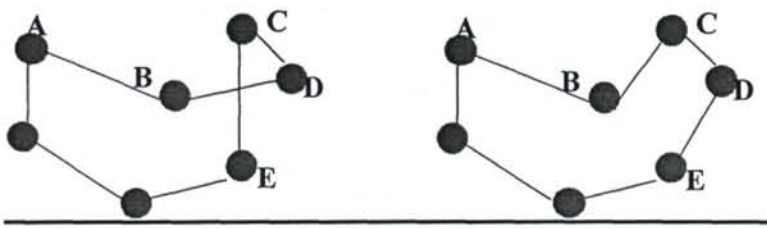


Fig. 2.7 Tras la recolocación de la cadena formada por los elementos C – D entre los elementos B y E, la ruta resultante puede ser mejorada posteriormente.

2.3 Comparación de Búsqueda Local con Diferentes Entornos e Instancias Simuladas.

Pretendemos averiguar que procedimiento de Búsqueda Local proporciona mejores soluciones para el VRPTW y VRPTW Mixto (ver apéndice 1.5 y 1.3); así mismo analizaremos el tiempo de computación utilizado en cada caso.

Se van a comparar los procedimientos *Búsqueda_Local_Or* con $k = 3$, *Búsqueda_Local_Or* con $k = \infty$, *Búsqueda_Local_Ge* y *Búsqueda_Local_Ta* con $k = 3$. Para ello se han considerado dos tipos de instancias; el primer tipo considera instancias con descarga ($n2 = 0$; instancias puras), y el segundo con igual número de puntos de carga que de descarga ($n1 = n2$; instancias mixtas). Para cada tipo se consideran 10 tamaños diferentes: 10 puntos, 20, ... hasta 100 puntos. Para cada tipo y tamaño se generan 10 instancias. Los datos de cada instancia se definen de la forma siguiente[3]:

- Se asigna a cada punto del problema dos coordenadas x e y, cuyos valores son generados aleatoriamente con distribución uniforme entre 0 y 100. La distancia entre cada par de puntos se define como la distancia euclídea correspondiente. Los tiempos (en minutos) de trayecto se toman igual a la distancia (en kilómetros.); es decir consideramos una velocidad de 60 kms/hora. El coste por kilómetro es de 1 u.m.

- Se asigna respectivamente a e_i y l_i para $i = 2,..., n1+n2$, dos valores enteros aleatorios generados uniformemente entre 600 y 720 (minutos) en el primer caso y entre 960 y 1.080 en el segundo. Se hace e_1 igual a 480 y l_1 igual a 1.200.

- A cada $q(i)$, $i = 2, ..., n1+n2$, se le asigna un valor entero generado uniformemente de forma aleatoria entre 1 y 12.

---

[3] La nomenclatura utilizada se explica en el apéndice 2.

- En todos los casos se supone dos tipos de vehículos, con capacidades 10 y 20, y con costes 1.000 y 1.200 u.m. respectivamente.

Los cuatro procedimientos de Búsqueda Local utilizan la misma solución inicial, obtenida por una adaptación a este modelo de un algoritmo de *inserción* que se describe en el apéndice 3. En las tablas siguientes se muestran los resultados medios de los costes y los tiempos de computación, según el tipo de instancia y tamaño, para la solución inicial y para cada uno de los cuatro procedimientos mencionados.

Tipo I (n2 = 0)

Tamaño	Solución Inicial	B.L._ Or(k=3)	B.L._ Or(k=∞)	B.L._ Ge(k=3)	B.L._ Ta(k=3)
10	40.240,4 0,0091	4.076,4 0,0020	4.058,3 0,0070	4.029,0 0,0150	**4.021,4** 0,0130
20	7.786,6 0,0221	7.629,8 0,0180	7.498,9 0,0750	7.486,1 0,0621	**7.459,5** 0,0992
30	12.126,4 0,0602	11.815,5 0,0390	11.768,1 0,3476	11.707,1 0,2653	**11.673,9** 0,2653
40	16.098,6 0,1222	15.594,8 0,1341	15.292,9 1,1598	**15.264,1** 0,3678	15.355,7 0,7179
50	19.631,0 0,2143	19.215,7 0,1523	18.903,3 2,1961	18.796,8 0,6009	**18.659,8** 1,2498
60	23.887,1 0,3504	23.317,3 0,3115	22.775,3 4,8688	22.577,7 1,0233	**22.550,4** 2,1552
70	27.314,1 0,5378	26.762,0 0,4026	26.112,1 9,5955	**25.937,7** 1,5423	25.951,7 3,7375

Tamaño	Solución Inicial	B.L._ Or(k=3)	B.L._ Or(k=∞)	B.L._ Ge(k=3)	B.L._ Ta(k=3)
80	31.556,0 0,7650	31.047,1 0,4387	30.234,2 18,1501	29.945,0 2,2281	**29.830,8** 5,2667
90	34.568,7 1,0727	33.994,5 0,8281	33.114,2 27,2734	32.984,1 3,6935	**32.937,7** 8,0858
100	39.481,6 1,4089	39.053,6 0,8914	37.989,3 50,5523	37.770,0 4,2954	**37.681,8** 10,6062

Fig. 2.8 Tabla.

Tipo II (n1 = n2)

Tamaño	Solución Inicial	B.L._ Or(k=3)	B.L._ Or(k=∞)	B.L._ Ge(k=3)	B.L._ Ta(k=3)
10	2.860,5 0,0090	2.793,9 0,0070	2.787,1 0,0050	2.779,4 0,0110	**2.778,2** 0,0120
20	4.890,8 0,0330	4.585,9 0,0210	4.579,2 0,0511	4.544,0 0,0472	**4.523,9** 0,0902
30	6.865,7 0,0911	6.670,7 0,0411	**6.295,2** 0,2354	6.377,0 0,1043	6.360,7 0,2783
40	9.417,5 0,1927	9.292,0 0,0900	9.041,8 0,7401	9.054,0 0,2124	**8.869,9** 0,6569
50	11.488,1 0,3674	11.088,4 0,1711	10.755,7 1,4770	10.914,5 0,4226	**10.720,1** 1,5293
60	14.183,7 0,5887	13.515,6 0,4167	**13.187,0** 4,0559	13.222,0 0,8834	13.220,7 3,2409
70	15.727,7 0,9465	15.360,0 0,4897	14.737,0 6,5393	**14.603,4** 1,3748	14.725,4 5,2313
80	17.424,8 1,5972	17.043,1 0,6250	16.647,6 11,3062	16.611,3 1,9637	**16.369,4** 8,0787
90	20.075,4 1,9460	19.388,7 0,9323	18.703,9 17,0927	18.696,2 2,8691	**18.598,5** 11,2082
100	21.790,9 2,6690	21.092,3 1,2338	**20.641,5** 23,6641	20.850,7 3,4097	20.661,7 14,4868

Fig. 2.9 Tabla.

Las conclusiones que podemos extraer en este apartado son las siguientes:

- Los procedimientos basados en *intercambios de Or* con $k = 3$ dan lugar a peores soluciones que los basados en *intercambios tipo Gendreu-Clark* y su generalización, es decir, los *intercambios tipo Taillard* (*CROSS intercambios*).

- Tomando $k = \infty$, y con instancias del segundo tipo ($n1 = n2$), los *intercambios de Or* consiguen soluciones cercanas e incluso algunas mejores que los *intercambios tipo Gendreu* y los *CROSS intercambios*, pero siempre a costa de utilizar un tiempo de computación considerablemente mayor.

- Los *CROSS intercambios* de Taillard consiguen, por lo general, mejores soluciones que los *intercambios tipo Gendreu*, y en definitiva las mejores soluciones de todos. En cuanto al tiempo de computación los CROSS intercambios emplean más que los de Gendreu, pero sustancialmente menos que los de Or, para $k = \infty$.

En vista de los resultados podemos decir que las mejores soluciones, en general, las aportan los *CROSS intercambios* de Taillard. Sin embargo, en aquellos casos en los que se pretenda aportar buenas soluciones empleando el menor tiempo de computación posible, los *intercambios tipo Gendreu* serían la mejor opción de las analizadas ya que:

- las soluciones son tan sólo un 0,182% peores para problemas puros y un 0,705% para problemas con carga y descarga, y
- el tiempo de computación se reduce en un 56,226% para problemas puros y en un 74,787% para problemas con carga y descarga.

2.4 Comparación de Diferentes Criterios de Elección de Nuevas Soluciones.

En la sección anterior, en todos los casos, los procedimientos de búsqueda local se movían en cada paso hacia la mejor solución vecina; este criterio de elección de soluciones vecinas es el denominado *mayor descenso*. Sin embargo, como ya se ha comentado anteriormente, no es el único criterio existente. Otras opciones son por ejemplo:

- *descenso aleatorio* que consiste en elegir aleatoriamente una solución que mejore la actual, o
- *primer descenso*, criterio que elige la primera solución que mejora la actual en la exploración del vecindario.

Algunas experiencias, como en Laguna M. y otros (1.994) y (1.998), o Hansen P. y Mladenovic N. (1.999) indican que el

criterio de *mayor descenso*, no necesariamente lleva a mejores soluciones finales ya que en muchos casos puede llevar al proceso a "encajonarse" prematuramente en mínimos locales muy cercanos a la solución inicial; la estrategia de primer descenso, ahorra tiempo de computación al no explorar todo el vecindario, y puede llevar al proceso a mejores soluciones.

Para analizar este punto, a continuación se van a comparar los dos criterios, mayor descenso y primer descenso, en procedimientos de Búsqueda Local con vecindarios tipo Taillard, dado que es el que hemos visto que aporta mejores soluciones. El criterio de mayor descenso fue usado en la sección 2.2; para obtener el procedimiento de Búsqueda Local con primer descenso, vale sustituir en *Movimiento_Vecinal_Ta*(k, s_0):

Determinar $s' \in N^{\infty}_3(s)$ *verificando* $f(s') = min\{f(s) / s \in N^{\infty}_3(s_0)\}$

por:

Tomar s' *el primer elemento de* $N^{\infty}_3(s)$ *verificando* $f(s') < f(s_0)$.

Para comparar ambos criterios se han simulado el mismo tipo de instancias, tamaños, número, y forma de definir los datos que en la sección anterior. La solución inicial también se ha generado como en la sección anterior, mediante una adaptación a este modelo de un algoritmo de *inserción.*

La estrategia de *descenso aleatorio* supone explorar todo el vecindario, con lo que el tiempo de computación será similar o

mayor al de *mayor descenso*. Las soluciones, según como se implementen *primer descenso* y *descenso aleatorio* pueden llegar a ser similares. No obstante adaptaciones de descenso aleatorio serán objeto, al menos, de análisis en futuros trabajos.

A continuación se muestran los resultados medios en cuanto a costes y tiempo de computación de las estrategias de mayor descenso y primer descenso por tipo de instancias y tamaño.

Tipo I (n2 = 0)

Tamaño	B. Local_Ta (k=3) Mayor Descenso		B. Local_Ta (k=3) Primer Descenso	
	Coste	Tiempo Comput.	Coste	Tiempo Comput.
10	**3.713,2**	0,0220	3.723,1	0,0231
20	**7.526,9**	0,0881	7.538,5	0,0781
30	11.516,3	0,2965	**11.428,0**	0,2483
40	**15.554,0**	0,5607	15.554,1	0,4917
50	**19.555,5**	1,2518	19.614,9	0,8771
60	**22.327,6**	2,0931	22.490,9	1,3169
70	26.548,2	3,6424	**26.453,4**	2,2282
80	31.048,6	5,7562	**31.016,5**	3,3250
90	**32.708,5**	7,6140	32.844,7	4,4865
100	**36.613,2**	11,3324	36.738,3	6,2239

Fig. 2.10 Tabla.

Tipo II (n2 = n1)

Tamaño	B. Local_Ta (k=3) Mayor Descenso		B. Local_Ta (k=3) Primer Descenso	
	Coste	Tiempo Comput.	Coste	Tiempo Comput.
10	**2.286,2**	0,0150	2.289,0	0,0110
20	4.669,4	0,0901	**4.562,1**	0,0882
30	6.594,4	0,2652	**6.592,8**	0,2383
40	**8.593,5**	0,6408	8.795,3	0,4897
50	**10.380,3**	1,6875	10.421,7	1,0366
60	**12.714,4**	2,5838	12.789,4	1,5663
70	**14.226,7**	4,6617	14.343,5	2,8724
80	**16.877,7**	7,6078	17.015,6	4,2925
90	18.554,3	10,3379	**18.536,8**	6,0424
100	20.839,4	16,5227	**20.782,4**	10,0444

Fig. 2.11 Tabla.

Los mejores resultados vienen en negrita y sombreados. Se observa que el criterio de *mayor descenso* lleva, en el 65% de los casos, a soluciones ligeramente mejores (el coste disminuye en un 0,238% respecto a primer descenso); si analizamos los resultados para *primer descenso* hemos de destacar que:

- en cuanto a costes, cuando *primer descenso* es peor, lo es tan sólo en un 0,5980% del coste resultante con *mayor descenso*

(0,311% para problemas puros y 0,884% para problemas con carga y descarga) ;

- en cuanto a tiempo de computación, *primer descenso* emplea un tiempo de computación inferior en 19 de los 20 tipos de instancias simulados; en media la reducción con respecto a *mayor descenso* es del 40,413% para el total de los 20 problemas.

En definitiva, *mayor descenso* es mejor, pero no se puede descartar el uso de *primer descenso*, sobre todo cuando el tiempo de computación es importante.

2.5 Ahorro de Tiempo de Computación Mediante Uso de la Búsqueda Local Rápida.

En esta sección adaptamos al VRPTW Mixto una idea propuesta por Bentley J.L. (1.992) para el TSP, y usada por Voudouris C. y Tsang E. (1.995) también para el TSP, que puede ahorrar mucho tiempo de computación en procedimientos de Búsqueda Local, sin perjuicio de la solución obtenida.

2.5.1 Planteamiento Inicial y Resultados Computacionales.

La idea básicamente consiste en dividir el vecindario de la solución actual en subvecindarios más pequeños; se asocia a cada uno de estos subvecindarios, una variable binaria que indique si están activos o inactivos, de forma que en cada paso solamente se exploran los subvecindarios activos. Inicialmente todos están activos; en cada iteración, si al explorar un subvecendario vemos que no contiene ninguna solución que mejore la actual, pasa a ser inactivo, en caso contrario permanece activo. Por otra parte si como resultado de un movimiento un subvecindario cambia su estructura, entonces pasa a ser activo.

De esta forma a medida que el proceso avanza y la solución mejora, hay menor número de subvecindarios activos y por tanto se gasta menos tiempo de computación en exploraciones innecesarias sin que quede afectada la solución final. Esta modificación en la forma de exploración del entorno da lugar a lo que se denomina *Búsqueda Local Rápida (BLR)*.

En este sentido cuanto mayor sea el número de subvecindarios que se consideren, más tiempo de computación se consigue ahorrar como lo ilustra la figura 2.12: se representa con el círculo grande de la izquierda, un subvecindario grande de la solución actual sin subdividir, y con el círculo grande de la derecha el mismo

subvecindario dividido en subvecindarios más pequeños. Los círculos pequeños en blanco representan soluciones que no mejoran la actual y el círculo pequeño en negro una que si mejora la actual. Esta solución obliga en el primer caso, a explorar todo el subvecindario grande en los siguientes pasos; en el segundo caso, sólo es necesario explorar el subvecindario pequeño que contiene la solución que mejora la actual, ya que los demás estarán inactivos.

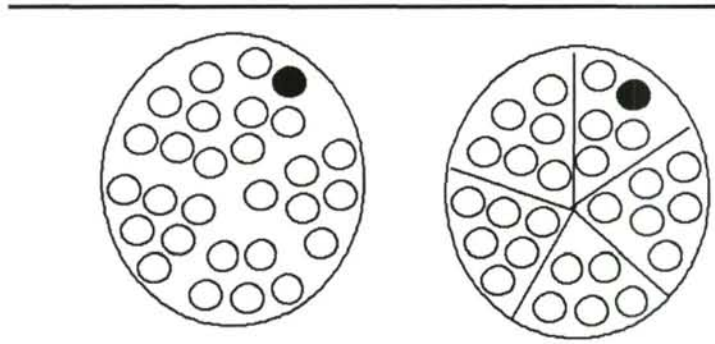


Fig. 2.12 Subvecindario sin dividir: hay que explorarlo todo. Subvecindario dividido: no hay que explorar las partes que no contengan mejora.

En nuestro caso se usa la misma idea básica aplicada a *Búsqueda_Local_Ta*. El conjunto de intercambios tipo Taillard lo vamos a dividir en subconjuntos de la siguiente forma: para cada solución s y para cada par de puntos i e i_1 pertenecientes a rutas diferentes (incluimos los *1* iniciales) se define $N_3(i, i_1)$ como: conjunto de intercambios entre cadenas, que tienen su origen en $i+1$, e i_1+1 respectivamente. Cada cadena sólo puede contener elementos de una misma ruta y no puede contener al 1. El tamaño

de alguna de las cadenas puede ser 0; (como se indicó en la sección 2.1.3).

Así se tiene que $N^{\infty}{}_3(s)$ se puede escribir como unión disjunta de los subconjuntos $N_3(i, i_1)$, para todos los pares de elementos i e i_1 pertenecientes a rutas diferentes. Fácilmente se puede guardar el carácter activo o inactivo de todos los subconjuntos en una matriz booleana, con gasto de memoria pequeño.

De esta forma cuando un subconjunto $N_3(i, i_1)$ se ha explorado sin encontrarse mejora se debe desactivar; así mismo cuando se realiza un intercambio se deben hacer activos los subconjuntos donde o bien i o bien i_1 sea alguno de los elementos que aparecen en alguna de las rutas afectadas por el intercambio.

Se va a analizar el efecto de incorporar esta idea al procedimiento *Búsqueda_Local_Ta*, tanto con el uso criterio de *primer descenso* como con el de *mayor descenso*. Para ello se han simulado el mismo tipo de instancias, tamaños, número y forma de definir los datos que en la sección anterior. La solución inicial también se ha generado como en el apartado anterior.

En las tablas 2.13 y 2.14 se muestran los tiempos medios de computación para cada tipo de problema y según el tamaño. En cursiva y sombreado se muestra el mejor resultado para cada tipo de instancia en cada uno de los dos métodos de elección de solución considerados (mayor descenso y primer descenso); en

negrita se resalta el mejor de todos los tiempos para cada tipo de instancia.

Tipo I (n2 = 0)

Tamaño	B. Local_Ta (k=3) Mayor Descenso		B. Local_Ta (k=3) Primer Descenso	
	Sin BLR	Con BLR	Sin BLR	Con BLR
10	0,0231	***0,0190***	*0,0200*	0,0221
20	0,0910	***0,0783***	0,0960	*0,0913*
30	0,3026	*0,1973*	0,2393	***0,1894***
40	0,6087	*0,3257*	0,4527	***0,3112***
50	1,3278	*0,5547*	1,0245	***0,5339***
60	2,3944	*0,8873*	1,4821	***0,7571***
70	3,7572	*1,1877*	2,2452	***1,0085***
80	5,3517	*1,4973*	3,3076	***1,3201***
90	7,7070	*1,9498*	3,9675	***1,5321***
100	10,9106	*2,4966*	6,3632	***2,2200***

Fig. 2.13 Tabla.

Tipo II (n2 = n1)

Tamaño	B. Local_Ta (k=3) Mayor Descenso		B. Local_Ta (k=3) Primer Descenso	
	Sin BLR	Con BLR	Sin BLR	Con BLR
10	0,0170	***0,0090***	0,0140	*0,0110*
20	0,0842	*0,0792*	***0,0701***	0,0721
30	0,3624	*0,2966*	0,2571	***0,2365***
40	0,7872	*0,5088*	0,5669	***0,4306***
50	1,5615	*0,8572*	1,1798	***0,7513***
60	2,9594	*1,4069*	1,8558	***1,1328***
70	4,0508	*1,6563*	2,4434	***1,3200***
80	6,7858	*2,5828*	3,8936	***1,9608***
90	9,8130	*3,2554*	5,6792	***2,7121***
100	15,4442	*4,7399*	6,7888	***2,9441***

Fig. 2.14 Tabla.

A la vista de los resultados es claro el efecto positivo del uso de BLR: el tiempo de computación se reduce sustancialmente, y más a medida que el tamaño del problema aumenta. Para instancias de tamaño 100 y en algunos de los problemas puros, el tiempo de computación llega a ser 5 veces menor. La reducción es

ligeramente inferior, aunque sigue siendo muy significativa con el criterio de *Primer Descenso*; la explicación es clara: al tomar la primera solución que mejora la actual no se exploraran todos los intercambios, pudiendo quedar sin explorar subconjuntos activos, que de esta forma no se desactivan aunque no ofrezcan mejoras.

Indicar también que los tiempos de computación de ambos criterios (mayor descenso y primer descenso), reducen sus diferencias con el uso de Búsqueda Local Rápida. A continuación se muestra una gráfica (figura 2.15) de la evolución de los tiempos de computación para las instancias del primer tipo:

- 1st - Mayor Descenso / Sin BLR;
- 2st - Mayor Descenso / Con BLR;
- 3st - Primer Descenso / Sin BLR;
- 4st - Primer Descenso / Con BLR;

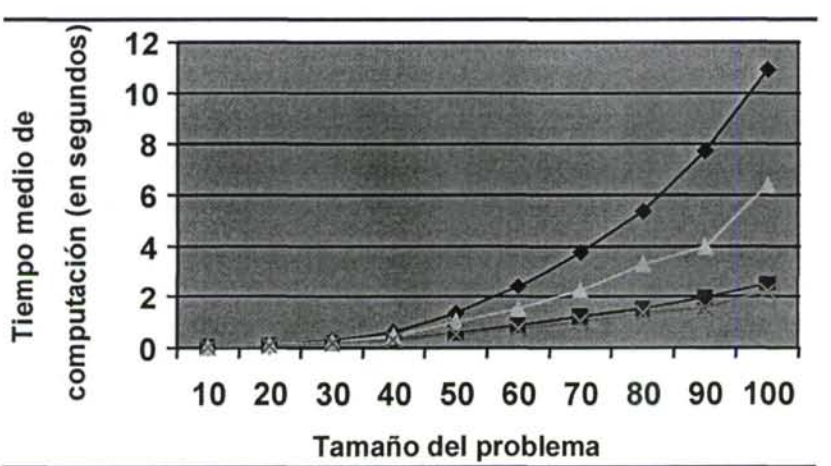


Fig. 2.15 Evolución de los tiempos medios de computación según el tamaño (instancias tipo I).

2.5.2 Modificación en Búsqueda Local Rápida.

Por los resultados del apartado anterior es evidente el efecto positivo de la incorporación de la Búsqueda Local Rápida: división del vecindario en subvecindarios e incorporación del carácter activo o inactivo a cada subvecindario; de esta forma sólo se exploran los subvecindarios modificados en la iteración anterior o que contengan mejoras en la solución actual.

Pero, ¿para que repetir la exploración de un vecindario que no se ha modificado aunque tenga mejoras?; ¿no sería mejor usar la información encontrada en pasos anteriores sobre ese subvecindario en vez de repetir la exploración?

Para no repetir exploraciones innecesarias, proponemos asociar a cada subvecindario una variable binaria que indique si se ha modificado o no en la última iteración y otra variable que contenga la información necesaria sobre la mejor solución de ese subvecindario. La forma de proceder es la siguiente: inicialmente se exploran todos los subvecindarios; en las siguientes iteraciones sólo se exploran los subvecindarios modificados en la iteración anterior; cuando un subvecindario se explora se guarda la información sobre su mejor solución; de los subvecindarios no modificados sólo se considera su mejor solución sin que haga falta explorar el resto de soluciones. Así, es posible esperar muy ligeras

mejoras en los tiempos de computación al no tener que explorar subvecindarios no modificados en pasos anteriores.

Se asocia a cada subconjunto $N_3(i, i_1)$ una variable booleana que indique si se ha modificado o no, y un registro que guarde el tamaño k y k_1 de las cadenas que intervienen en el mejor intercambio así como el valor de la ganancia o perdida de dicho intercambio. Obsérvese que en este caso se usa mucha más memoria que en la propuesta anterior de Búsqueda Local Rápida, ya que además de una matriz booleana, se usa una matriz de una variable compuesta por dos valores enteros (tamaños de las cadenas) y uno real (ganancia/pérdida).

En las siguientes tablas (figuras 2.16 y 2.17) se analiza el efecto de estas modificaciones utilizando *Búsqueda_Local_Ta* como en la sección anterior. Se compara la evolución de los tiempos de computación *sin* usar *Búsqueda Local Rápida*, con *Búsqueda Local Rápida*, y *Búsqueda Local Rápida Modificada*; en todos los casos se usa el criterio de *mayor descenso*. Para ello se han simulado los mismos tipos de instancias, forma de definir los datos y solución inicial que en los casos anteriores.

El tamaño en este caso llega hasta 200 puntos y se han generado 10 instancias para cada tamaño. Se muestran los tiempos medios de computación en segundos, por tamaño y tipo. Los mejores resultados para cada tamaño se escriben en negrita y sombreado.

Tipo I (n2 = 0)

Tamaño	Búsqueda Local_Ta (k=3) Mayor Descenso		
	Sin Búsqueda Local Rápida	Búsqueda Local Rápida	B. L. Rápida Modificada
10	0,0230	**0,0210**	0,0211
20	0,0790	**0,0682**	0,0712
30	0,2814	**0,1785**	0,1793
40	0,7181	**0,3866**	0,3897
50	1,2726	**0,5597**	0,5657
60	2,3384	0,8553	**0,8502**
70	3,8106	1,2278	**1,2147**
80	5,3839	1,5009	**1,4691**
90	8,4261	2,1079	**2,0629**
100	12,0320	2,7220	**2,6280**
110	16,5340	3,9037	**3,7835**
120	20,1770	4,3652	**4,2289**
130	28,0233	5,6142	**5,3788**
140	35,3428	6,5306	**6,2449**
150	44,2526	7,7021	**7,3106**
160	52,2871	8,3791	**7,9394**

Tamaño	Búsqueda Local_Ta (k=3) Mayor Descenso		
	Sin Búsqueda Local Rápida	Búsqueda Local Rápida	B. L. Rápida Modificada
170	63,7826	9,8651	**9,2364**
180	77,1929	11,3262	**10,6244**
190	89,5038	12,6432	**11,8860**
200	108,5680	14,4858	**13,5004**

Fig. 2.16 Tabla.

Tipo II (n1 = n2)

Tamaño	Búsqueda Local_Ta (k=3) Mayor Descenso		
	Sin Búsqueda Local Rápida	Búsqueda Local Rápida	B. L. Rápida Modificada
10	0,0340	**0,0321**	0,0330
20	**0,1091**	0,1103	0,1182
30	0,4346	**0,3636**	0,3735
40	0,7752	**0,5497**	0,5610
50	1,6252	**0,9656**	0,9845
60	3,6713	**1,8326**	1,8595
70	5,1062	**2,2893**	2,3182
80	8,6466	**3,3387**	3,3756

Tamaño	Búsqueda Local_Ta (k=3) Mayor Descenso		
	Sin Búsqueda Local Rápida	Búsqueda Local Rápida	B. L. Rápida Modificada
90	13,4894	**4,6957**	4,7430
100	15,3832	**4,9472**	4,9851
110	22,6018	**6,7879**	6,8611
120	29,4053	**7,9507**	7,9868
130	42,8997	**10,4511**	10,5020
140	48,6099	**11,2151**	11,2302
150	67,6903	14,6719	**14,5929**
160	82,8242	16,9565	**16,8753**
170	103,9865	20,3321	**20,2672**
180	132,2781	24,0506	**23,8544**
190	139,0670	24,8016	**24,7305**
200	167,3755	27,9641	**27,8081**

Fig. 2.17 Tabla.

A la vista de los resultados, se aprecia que el efecto de la modificación propuesta en la Búsqueda Local Rápida es positivo, aunque no muy significativo si se le compara con el efecto producido por el uso de la propia Búsqueda Local Rápida. Con instancias de menor tamaño la gestión de memoria de las propias

variables introducidas hace que el tiempo de computación aumente (aunque insignificantemente). Sin embargo a medida que aumenta el tamaño, el ahorro en el tiempo de computación se va haciendo mayor (tanto de forma relativa como absoluta), especialmente en el primer tipo de instancias, llegando hasta el 7% aproximadamente. Es previsible que para problemas mayores el ahorro aumente aún más.

En la figura 2.18 se muestra la evolución de los tiempos medios de computación según el tamaño para las instancias del tipo I:

- NBLR = Sin Búsqueda Local Rápida;
- BLR = Búsqueda Local Rápida;
- BLRM = Búsqueda Local Rápida Modificada.

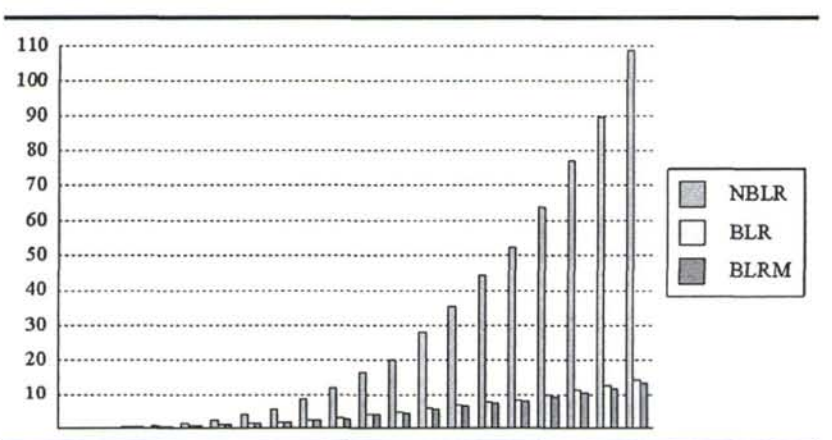


Fig. 2.18 Evolución de los tiempos medios de computación según el tamaño (instancias tipo I).

2.6 Análisis de los Resultados.

En este capítulo se ha pretendido analizar algunos aspectos relacionados con el uso de la Búsqueda Local en problemas de rutas. El primero de ellos es la discusión sobre el tipo de entorno que se construye. Se ha realizado una comparativa de diferentes tipos de entornos usados en la literatura. En base a los resultados obtenidos, parece claro que, tanto en tiempo de computación como en calidad de los resultados, es mejor en general considerar intercambios de elementos entre pares de rutas, que considerar la solución como una única cadena compacta. En este sentido las mejores soluciones las aporta el *CROSS intercambio de Taillard*, aunque para problemas de gran tamaño, quizá fuera conveniente limitar el tamaño de las cadenas a recolocar, o incluso si lo que se precisa es la utilización de un menor tiempo de computación, emplear los *intercambios tipo Gendreu* cuyas soluciones no se alejan mucho de las aportadas por los CROSS intercambios.

En segundo lugar, también se ha puesto de manifiesto que puede ser conveniente, sobre todo para problemas de gran tamaño, no tomar la mejor solución vecina (*mejor descenso*), sino la primera que mejore (*primer descenso*), puesto que se ahorra casi la mitad del tiempo de computación y la calidad de la solución es parecida. Para problemas de rutas como los analizados aquí, la estrategia de *primer descenso* es aconsejable para todas aquellas situaciones en las que el tiempo de computación es importante, ya

que se ha visto que las soluciones aunque peores, son muy similares a las proporcionadas por mejor descenso.

En tercer lugar, es totalmente recomendable el uso de la *Búsqueda Local Rápida*, o posibles variantes como la analizada en el punto anterior; la incorporación de esta estrategia nos ha permitido conseguir espectaculares reducciones en el tiempo de computación, sin afectar a la solución final. Esto posibilita abordar problemas de mayor tamaño que, teniendo en cuenta los tiempos de computación, de otra forma serían imposibles de abordar. Entendemos que esta idea puede ser adaptada fácilmente a otros modelos: se trata de gastar algo más de memoria (no tiene que ser mucha) donde guardar información obtenida en pasos anteriores, para no repetir cálculos y ahorrar un considerable tiempo de computación.

Indice

CAPÍTULO III

DISEÑO DE UN ALGORITMO: USO DE UN CONJUNTO DE CONCENTRACIÓN EN BÚSQUEDA TABÚ.

3.1 Diseños de Algoritmos Basados en GRASP y Concentración Heurística.

En este apartado se propone un algoritmo Metaheurístico para el problema de rutas con ventanas de tiempo, carga y descarga simultánea y flota heterogénea (ver apéndice 1.3, 1.5, 1.6), basado en un proceso de tipo *Concentración Heurística*. La finalidad es comprobar como esta estrategia dada a conocer, como ya se indicó en el apartado 1.5.4.2, por Rosing K.E. (1.997) y Rosing K.E. y ReVelle C.S. (1.997) mejora en muchos casos las soluciones obtenidas por otros metaheurísticos para el citado modelo.

La función de costes a minimizar f se descompone en dos partes:

- una parte proporcional a la distancia total recorrida;
- el coste fijo por cada vehículo (según su tipo).

La idea básica recordemos que es sencilla: obtener un conjunto de buenas soluciones; formar un conjunto, denominado *Conjunto de Concentración (CS)*, con los elementos que componen dichas soluciones; y finalmente ejecutar un algoritmo, exacto o heurístico, que busque solamente en las soluciones que contengan (en su

totalidad o en la mayor parte, según el diseño que se haga) elementos de este CS. En este caso se opta por diseñar un algoritmo heurístico para la segunda fase que intensifique la búsqueda en soluciones que contengan elementos de CS, en vez de usar un exacto.

Las soluciones generadas para formar el CS no se construirán de forma aleatoria sino de forma ávido-aleatoria, que por lo general, son mejores. De esta forma, en realidad, la primera fase de Concentración Heurística es un GRASP. La forma de generar estas soluciones ávido-aleatorias consiste básicamente en considerar cada problema del VRPTW Mixto como un Problema de Asignación Lineal Generalizada como se hace en Fisher M.L. y Jaikumar R. (1.981); posteriormente se resuelve cada problema de Asignación Lineal Generalizada usando el método de aproximación MTHG propuesto en el texto de Martello y Toth (1.990).

3.1.1 Diseño de un GRASP.

Fisher M.L. y Jaikumar R. (1.981), diseñan un algoritmo para el VRP considerado por varios autores (Haouari M. y otros (1.990); Laporte G. (1.992),...) como uno de los métodos constructivos de mayor eficacia. La idea es plantear el VRP como un problema de Asignación No lineal Generalizada, de la siguiente forma:

Sean: m - el número de rutas totales;

1 - el punto origen,

$\{2, ..., n\}$ - el conjunto de puntos de visita,

$q(i), i = 2, ..., n$ - la carga a llevar a cada punto; y

Q - la capacidad de cada vehículo;

Sean las variables:

x_{ij} = 1, si se asigna el envío i a la ruta j; 0 en caso contrario; $i = 2, ..., n$ y $j = 1, ..., m$;

z_j = $\{i / x_{ij} = 1, i = 2, ..., n\}$, i.e. el conjunto de envíos asignados a la ruta j; $j = 1, ..., m$;

El problema se puede formular como:

$$min \sum_{j=1}^{m} g(z_j) \qquad (1)$$

sujeto a:

$$\sum_{i=2}^{n} x_{ij} q(i) \leq Q \qquad j = 1, ..., m \qquad (2)$$

(las carga de los puntos asignados a cada ruta no superan la capacidad de los vehículos)

$$\sum_{j=1}^{m} x_{ij} = 1 \qquad i = 2, .., n \qquad (3)$$

(cada envío se debe realizar por una ruta)

$$x_{ij} \in \{0, 1\} \qquad i = 2, ..., n \qquad j = 1, ..., m; \qquad (4)$$

$g(z_j)$ se define como el coste de la ruta óptima que desde el origen recorre las delegaciones correspondientes a z_j y vuelve. Obviamente $g(z_j)$ no es función lineal de los x_{ij}; sin embargo Fisher M.L. y Jaikumar R. (1.981) proponen definir unos puntos ficticios o *puntos-semilla* e_j, para $j = 1, ..., m$ y sustituir la función objetivo anterior, por otra lineal:

$$min \sum_{i=2}^{n} \sum_{j=1}^{m} c_{ij} x_{ij} \qquad (1')$$

donde c_{ij} es el coste de insertar el punto i en la ruta que va del origen 1 al punto semilla e_j; si el coste coincide con la distancia recorrida entonces $c_{ij} = d_{1,i} + d_{i,ej} - d_{1,ej}$, siendo:

- $d_{1,i}$ distancia del origen al punto de visita i;
- $d_{i,ej}$ distancia de i al punto-semilla e_j; y
- $d_{1,ej}$ distancia del origen a e_j).

Así planteado, (1'), (2), (3) y (4), se tiene un problema de Asignación Lineal Generalizada, (GAP); una vez resuelto este, en una segunda fase se resuelve el TSP para cada conjunto de puntos encontrado y el origen.

Obviamente, esta es una aproximación al problema original y, por tanto, el óptimo del GAP, no asegura el óptimo del VRP. Por otra parte el GAP es a su vez NP-Hard, por consiguiente, es claro que se debe optar por un método heurístico, al menos en este caso, para su resolución, como el propuesto por Martello S. y Toth P.

(1.990), que por lo general da buenos resultados. Básicamente actúa de la forma siguiente:

Algoritmo Martello_y_Toth_(MTHG)

Hacer $Q(j) = Q, j = 1,..,m;\ U = \{2,..,n\}$ *y* $x_{ij} = 0$.

Mientras $U \neq \varnothing$ *hacer:*

$\forall i \in U$: *determinar* $c_{ij'(i)} = min\ \{c_{ij} / q(i) \leq Q(j), j = 1,...,m\}$

determinar $c_{ij''} = min\ \{c_{ij} / q(i) \leq Q(j), j=1,...,m; j \neq j'(i)\}$

hacer $b_i = c_{ij''} - c_{ij'(i)}$

determinar $b_{i'} = max\ \{b_i / i \in U\}$

Hacer $x_{i'j'(i')} = 1,\ U = U - \{i'\},\ Q(j'(i')) = Q(j'(i')) - q(i')$.

Una cuestión importante es donde situar los *puntos-semilla*. En problemas donde hay varios puntos con mucha carga demandada (más de la mitad de la capacidad del vehículo) los *puntos-semilla* se pueden situar precisamente en los clientes de mayor demanda. La misma experiencia puede indicar cuales pueden ser esos puntos significativos. En cualquier caso Fisher M.L. y JaikumarR. (1.981), incorporan en su trabajo un método para su cálculo de forma general (ver figura 3.1):

- determinar las semirectas que unen el origen con cada punto $i \in \{2,..., n\}$;

- hallar las bisectrices de los ángulos que forman cada par de semirectas consecutivas. Cada ángulo *alfa*(i) entre dos

bisectrices consecutivas corresponde a un punto *i*, y lleva asociado un peso *q(i)* correspondiente a la carga del punto;

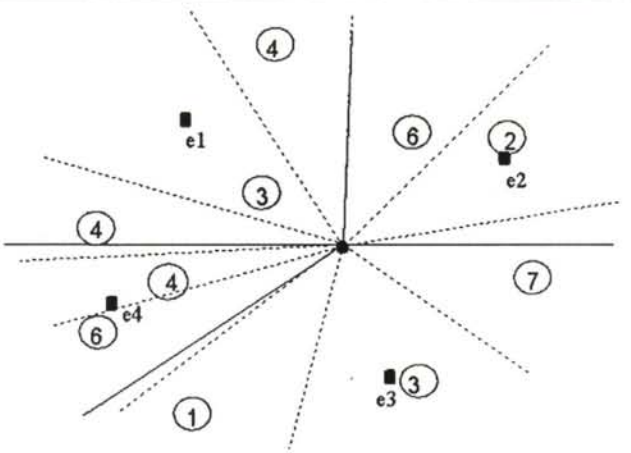


Fig. 3.1 Ejemplo de obtención de los puntos semilla (cuadros en negro), con m=4 y las cantidades de cada punto en circulo.

- comenzando por un punto cualquiera y "barriendo" en el mismo sentido ir formando *m* ángulos mayores *beta(j)*, con los ángulos *alfa(i)* o con partes proporcionales de estos, de forma que el peso total de estos nuevos ángulos sea $K = \sum_{i=2,..,n} q(i) / m$;

- cada punto semilla e_j se situará en la bisectriz del ángulo *beta(j)* de forma que el la suma de las cargas de los clientes en *beta(j)* que queden por debajo del arco que pasa por e_j trazado desde el origen más una proporción de la carga del cliente más cercano a este arco por fuera sea 0'75K. Esta proporción es A/(A+B) donde A y B son las distancias a los

puntos mas cercanos al arco por dentro y por fuera respectivamente.

En nuestro caso, se van a adaptar las ideas anteriores al GRASP que proponemos en este apartado, pero con las siguientes consideraciones:

- Durante la ejecución de MTHG a medida que se va añadiendo cada punto i a la ruta $j(i')$ se va a determinar también en que posición dentro de esa ruta va a ser insertado. En otras palabras no sólo se determinan los puntos que forman cada ruta sino que además se diseña el orden de dicha ruta;

- Las rutas j donde un punto i puede ser asignado deben cumplir no solamente que $q(i) \leq Q(j)$; además existen restricciones temporales (ventanas de tiempo), puntos de carga y puntos de descarga. Estos aspectos hacen que la factibilidad de la asignación dependa de la posición donde sea insertado; el siguiente ejemplo ilustra esta afirmación: supóngase, el origen O, un punto de descarga A con $q = 10$ y un punto de carga B con $q = 5$; la ruta O - A - B - O requiere un vehículo de 10 unidades de capacidad, sin embargo la ruta O - B - A - O requiere un vehículo de 15 unidades de capacidad. Por tanto se ha de determinar para cada punto i que queda por asignar y para cada ruta j los posibles posiciones donde ese punto puede ser insertado en esa ruta. Las rutas donde i se puede asignar serán aquellas con al menos una posición factible.

- De entre las posiciones factibles de cada ruta, será elegida aquella que menos incremente del coste total de la ruta (distancia y tipo de vehículo usado).

- Por tanto para cada punto i por asignar (a lo sumo n), y cada punto ya asignado i^* (a lo sumo $2n$ incluyendo los orígenes de cada ruta) se ha de examinar la factibilidad y evaluar el coste de la inserción de i tras i^*. Estas dos pasos pueden hacer muy expansivo el número de operaciones. En Pacheco J. y Delgado C.R. (1.996) y (1.997-b) proponemos un método, basado en el uso de variables globales, para realizar estos pasos en un número de operaciones constante e independiente del tamaño del problema.

- El orden de visita de cada una de las rutas formadas a la finalización de este proceso de asignación, obviamente no tienen por que ser el óptimo para ese conjunto de puntos (aunque se espera sea aceptable). Se pueden mejorar cada una de las rutas individualmente aplicando métodos de intercambio de Or.

Se va a usar la función b_i como función ávida para formar la *Lista Restringida de Candidatos* o LRC. De esta forma la variante propuesta para resolver el problema de asignación es de la forma siguiente:

Procedimiento Variante_MTHG

Hacer $U = \{2,..., n\}$, $bmax = -\infty$ *y factible* = *TRUE*

Mientras $(U \neq \varnothing)$ *y factible hacer:*

$i \in U$: Hacer $min = \infty$, $min1 = \infty$;

Para $j=1, ..., m$, si $c_{ij} < min1$:

Determinar y evaluar posiciones factibles (de la ruta j para la inserción de i) y guardar en k el valor de la mejor posición factible (si hubiera).

Si existen posiciones factibles entonces:

Si $c_{ij} < min$, hacer: $min1 = min$;

$min = c_{ij}$;

$j'(i) = j$;

$kpos(i) = k$

Si $c_{ij} \geq min$, hacer: $min1 = c_{ij}$

Si $min = inf$ entonces: Hacer factible = FALSE

En caso contrario: Hacer $b_i = min1 - min$;

Si $b_i > bmax$ hacer $bmax = b_i$

Formar el conjunto $LCR = \{i / i \in U, (b_i / bmax) > 1 - \alpha\}$

Elegir aleatoriamente un elemento $i' \in LCR$

Asignar i' en la ruta $j'(i')$ insertándolo en la posición $kpos(i')$

Hacer $U = U-\{i'\}$

Fin.

Obsérvese que se añade la variable auxiliar lógica factible que controla el proceso y lo detiene en el momento en que algún punto no puede asignarse a ninguna ruta. Cuando esto ocurre, se repite el proceso desde el principio añadiendo una ruta más. De esta forma la fase constructiva queda de la siguiente forma:

Procedimiento Fase_Constructiva_GRASP

Hacer $m = m1$

Repetir

Elegir aleatoriamente un elemento $i_1 \in \{2, ..., n\}$

Comenzando por el ángulo alfa(i_1*) construir los ángulos beta(j)*

Determinar los puntos-semilla

Calcular los coeficientes de asignación c_{ij}

Resolver el Problema de Asignación

Si (no factible) entonces $m = m+1$

Hasta factible

Mejorar cada ruta formada por Búsqueda local

Antes de iniciar las iteraciones que constituyen el algoritmo GRASP básico se han de determinar previamente los ángulos *alfa*(*i*) y un número mínimo de rutas iniciales $m1$. Este valor $m1$ será el máximo de $m2$ y $m3$ que son respectivamente los óptimos (o cotas inferiores) del Bin-Packing correspondientes a los puntos de carga y de descarga considerando la capacidad del bin o mochila la del tipo de vehículo de mayor capacidad. Por otra parte la formación de los ángulos *beta* se realizará sin distinguir entre puntos de carga o descarga. Finalmente el aleatorizar el ángulo *alfa* inicial para formar los beta se explica porque se ha comprobado que el elegir diferentes ángulos de partida puede variar los coeficientes de asignación y cambiar ligeramente el resultado final de esta fase, incluso aunque la Resolución del Problema de

Asignación sea determinística (elegir en cada paso el *i* correspondiente a *bmax*).

3.1.2 Diseño de la Fase I de Concentración Heurística.

Se va a seguir la idea básica descrita del algoritmo pero con algunas modificaciones que se especifican a continuación.

En primer lugar y remitiéndonos al capítulo 2, hay que definir cuáles son los elementos que componen una solución; recuérdese que en nuestro caso, cada solución viene representada por una secuencia de puntos, por ejemplo:

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

representa a una solución con dos rutas: 1 - 3 - 5 - 1 y 1 - 4 - 2 - 1. Tales secuencias de puntos pueden considerarse cadenas de arcos de la forma $(1, i_1)$, (i_1, i_2), (i_2, i_3), ..., $(i_s, 1)$; consideramos que los elementos que componen una solución son los arcos que aparecen en la cadena correspondiente.

Obsérvese que en este caso, las dos estrategias propuestas por Rosing, -considerar un único conjunto de elementos *CS* o dividirlo

en dos subconjuntos CS_0 (abierto) y CS_f (libre)-, coinciden: de todos los arcos que pueden aparecer en cada solución (n·(n-1) en total), debe aparecer uno y solamente uno que salga de cada punto de visita {2, 3,... ,n} y uno y solamente uno que llegue a él; si un arco aparece en todas las soluciones, en ambas estrategias aparecerá en la nueva solución; si hay varias opciones de llegadas/salidas a/de un punto, en ambas opciones serán arcos disponibles para la nueva solución. Supongamos que un arco aparece en todas las soluciones del conjunto de concentración, por ejemplo el arco (2, 3); utilizando la estrategia de división del conjunto de concentración dicho arco pertenece al conjunto abierto, y se toma como arco fijo para la nueva solución; si no se hace dicha distinción, no varía la solución final, pues como no hay otro que salga de 2, ni otro que llegue a 3, dicho arco es el único posible y aparece de igual modo en la nueva solución.

En segundo lugar, el número de óptimos locales m, seleccionados para construir el conjunto de concentración no se va a fijar previamente, sino que va a venir dado por un número de arcos diferentes prefijado, que en adelante denotaremos *num_arcos*. Concretamente, se ordenan los óptimos locales obtenidos en la primera fase según el valor de la función objetivo; comenzando por el mejor, se van añadiendo los elementos de cada uno de ellos al conjunto de concentración hasta que el cardinal del conjunto de concentración sea mayor o igual que *num_arcos*. Téngase en cuenta que si m soluciones son muy parecidas producen un conjunto de concentración pequeño; por el contrario

si las *m* soluciones son muy diferentes dan lugar a un conjunto de concentración muy grande. Por esta razón para evitar excesos en ambos sentidos se opta por fijar un número de elementos diferentes *num_arcos*.

En tercer lugar, la solución inicial de cada iteración no se genera de forma aleatoria sino que es de tipo ávido-aleatoria, ya que se producen por lo general mejores soluciones. De esta forma, en realidad, esta primera fase es un GRASP. La forma de generar estas soluciones es la descrita en el apartado anterior.

Con respecto al procedimiento de Búsqueda Local, se consideran soluciones vecinas, al conjunto de soluciones factibles obtenidas por intercambios de Or I. (1.976), pero sólo aquellas obtenidas mediante recolocaciones hacia adelante de cadenas de a lo sumo 3 elementos, considerando cada solución como una única secuencia de puntos (procedimiento descrito en el capítulo 2).

La primera fase puede escribirse en seudocódigo de la siguiente forma:

Procedimiento Concentración_Heurística-Fase_I

Repetir

Generar una solución Avida Aleatoria.

Aplicar Búsqueda_Local_Or(3, s_0).

Hasta un determinado número de iteraciones.
Registrar las soluciones obtenidas en una lista ordenada según el valor de la función objetivo 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(SC) ≥ num_arcos.

3.1.3 Diseño de la Fase II de Concentración Heurística.

Según se ha comentado anteriormente, se va a diseñar un algoritmo heurístico que *concentre* la búsqueda de elementos en el *CS*. Con el fin de contrastar la eficacia de esta búsqueda dirigida al *CS*, el algoritmo diseñado va a ser similar al de la primera fase. Inicialmente se define la 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 $max\_d = \max\{d(i,j) / i,j = 1, ..., n\}$; además se define $f1(s)$ como el valor de la función objetivo considerando el coste de los

arcos dado por $d1$. Se propone el siguiente procedimiento de búsqueda que tiene en cuenta ambas matrices d y $d1$:

Procedimiento Búsqueda_Local_Hacia_CS

Leer solución inicial s_0

Hacer $s^* = s_0$

Repetir

Hacer $coste\_anterior = f(s^*)$

Repetir

Buscar $s' \in N_1^3(s_0) / f1(s') = \min\{f1(s) / s \in N_1^3(s_0)\}$

Buscar $s'' \in N_1^3(s_0) / f(s'') = \min\{f(s) / s \in N_1^3(s_0)\}$

Si $f1(s') < f1(s_0)$ *entonces* $s_0 = s'$

Si $f(s'') < f(s^*)$ *entonces* $s^* = s''$

Hasta $f1(s') \geq f1(s_0)$

Hacer $s_0 = s^*$

Hasta $f(s^*) = coste\_anterior$

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 del *CS*; cuando no hay mejora en $f1$, se sustituye s_0 por s^*, la mejor solución según f observada en los vecindarios explorados y se reinicia la búsqueda local. El proceso acaba cuando no hay mejora en $f(s^*)$.

Es un procedimiento de búsqueda "guiado" por $f1$, es decir, por $d1$, hacia soluciones que contengan el mayor número posible de elementos del *CS*. Obsérvese que $d1$ penaliza a los arcos no pertenecientes al *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.

El procedimiento de *Búsqueda_Local_Hacia_CS* se inserta en la segunda fase de Concentración Heurística, que en seudocódigo puede ser escrita de la siguiente forma:

```
Procedimiento Concentración_Heurística-Fase_II
  Hacer iter_mejor = 0, niter = 0, coste_min = inf.;
  Repetir
    niter = niter+1;
    Generar una solución Avida-Aleatoria (considerando como
    función objetivo f1);
    Ejecutar Búsqueda_Local_Hacia_CS;
    Si f(s0) < coste_min hacer: coste_min = f(s0), s_mejor = s0,
    iter_mejor = niter
  Hasta niter - itermejor > 100
```

Esta fase es similar a la primera salvo las siguientes consideraciones:

- se toma $f1$ en vez de f al generar las soluciones iniciales, y
- se ejecuta el procedimiento de *Búsqueda_Local_Hacia_SC* dirigido por $f1$, en vez de la búsqueda local habitual; en definitiva, se "concentra" la búsqueda de soluciones en los elementos del *CS*.

3.1.4 Resultados Computacionales.

Se quiere comprobar si la segunda fase, dirigiendo la elección de elementos al *CS*, puede mejorar o no los resultados de la primera. También se pretende determinar con que "níveles de concentración", es decir, con que valores del parámetro *num_arcos* se consiguen mejores resultados. Por último, se quiere comprobar si las listas más grandes en la primera fase llevan a mejores soluciones en la segunda.

Para ello se han generado 50 instancias: 25 con 39 puntos de descarga y el origen, y otras 25 con 19 puntos de carga, 20 de descarga y el origen. Los datos de cada problema se definen de la forma siguiente (ver apéndice 2 para conocer la nomenclatura de las variables que intervienen):

- Se asigna a cada punto del problema dos coordenadas x e y, cuyos valores son generados aleatoriamente con distribución uniforme entre 0 y 100. La distancia entre cada par de puntos se define como la distancia euclídea correspondiente. Los tiempos (en minutos) de trayecto se toman igual a la distancia (en kilómetros.); es decir, consideramos una velocidad de 60 kilómetros/hora. El coste por kilómetro es de 1 u.m.

- Se asigna a e_i y l_i, para $i = 2, ..., n1+n2$, valores enteros aleatorios generados uniformemente entre 600 y 720 minutos para e_i, y entre 960 y 1.080 para l_i. Se hace e_1 igual 480 y l_1 igual a 1.200.

- Se asigna a cada $q(i)$, $i = 2, ..., n1+n2$, un valor entero generado uniformemente de forma aleatoria entre 1 y 12.

- En todos los casos se supone dos tipos de vehículos, con capacidades 10 y 20, y con costes 1.000 y 1.200 u.m. respectivamente.

De igual forma, en la fase I se han realizado 10.000 iteraciones obteniéndose 4 listas ordenadas de óptimos locales según el valor de f correspondientes a las 1.000, 2.000, 5.000 y 10.000 primeras iteraciones respectivamente. Además se ha generado una quinta lista ordenada de la siguiente forma: se vuelve a aplicar el procedimiento de Búsqueda Local usado en la primera fase a cada una de las 500 mejores soluciones de las generadas anteriormente,

pero tomando los vecindarios $N_1(s)= N_1^{nt-3}(s)$ en vez de $N_1(s) = N_1^3(s)$ (donde *nt* es el número de puntos que componen una solución *s*), es decir, se consideran todo el conjunto de recolocaciones ($N_1^{\infty}(s)$).

Procedimiento Obtención_Lista_5

Desde i = 1 hasta 500 hacer:

Tomar la i-ésima mejor solución de las 10.000 de la Fase I

Ejecutar Búsqueda_Local_Or(∞, s_0).

Guardar nuevas soluciones en una lista ordenada (Lista _5)

Para la segunda fase se han establecido 4 niveles de concentración, correspondientes a fijar *num_arcos* igual a 260 (16'6% de los arcos), 195 (12'5%), 130 (8'3 %) y 78 (5%), formándose para cada nivel y para cada lista el conjunto *CS* correspondiente (5 listas · 4 niveles = 20 conjuntos en cada instancia).

Las tablas correspondientes a las figuras 3.2 y 3.3 muestran el resumen de los resultados obtenidos para las instancias de tipo I y de tipo II respectivamente.

Los resultados observados son interesantes, especialmente para el primer tipo de instancias (sólo puntos de descarga): la estrategia utilizada en ambas fases de Concentración Heurística análoga, sin

embargo, las soluciones encontradas en la segunda fase superan al mejor óptimo local de la primera fase en casi todas las listas (en valores medios y en la mayoría de las instancias analizadas).

	Lista 1	Lista 2	Lista 3	Lista 4	Lista 5
Fase I	14.971,00	14.956,44	14.946,00	14.935,04	14.916,32
Nivel 1 (16'66%)	14.940,52	14.929,32	14.932,40	14.925,32	14.923,40
Nivel 2 (12'5%)	14.940,48	14.931,08	14.926,32	14.923,32	14.914,68
Nivel 3 (8'33%)	**14.937,40**	**14.926,24**	**14.921,20**	14.915,20	**14.900,48**
Nivel 4 (5%)	14.945,24	14.926,64	14.925,28	**14.914,48**	14.903,84

Fig 3.2 Resultados medios de las mejores soluciones obtenidas en cada fase para cada lista (tipo de instancias I).

En las dos últimas listas la mejora es menos notable; se puede suponer que en el caso de la lista cuatro esto se debe al gran número de iteraciones que se realizan (10.000) en comparación con las otras listas (5.000, 2.000 y 1.000). En el caso de la lista cinco hay que tener en cuenta dos hechos: en primer lugar que se toman las 500 mejores iteraciones; y en segundo lugar que en la fase I se utiliza el vecindario de Or con $k=\infty$, es decir, se exploran vecindarios muy amplios y por lo tanto es de suponer que se encontrarán soluciones muy buenas. Tanto es así, que la solución para la fase I de la lista cinco, es mejor que las mejores soluciones

encontradas en las listas uno, dos, tres y cuatros. La desventaja de la lista cinco frente a las demás, es el desmesurado tiempo de computación.

	Lista 1	Lista 2	Lista 3	Lista 4	Lista 5
Fase I	8.730,68	8.722,36	8.717,56	**8.710,88**	**8.680,68**
Nivel 1 (16'66%)	8.737,36	8.730,68	8.716,92	8.725,92	8.711,88
Nivel 2 (12'5%)	8.731,52	8.729,48	8.717,40	8.718,96	8.712,84
Nivel 3 (8'33%)	8.724,80	**8.718,88**	8.713,56	8.720,16	8.698,16
Nivel 4 (5%)	**8.719,80**	8.721,08	**8.705,88**	8.717,12	8.696,16

Fig. 3.3 Resultados medios de las mejores soluciones obtenidos en cada fase y en cada lista (tipo de instancias II).

En el segundo tipo de instancias (figura 3.3) y para las tres primeras listas, también se observa que las soluciones encontradas en la segunda fase superan al mejor óptimo local de la primera fase de la lista correspondiente, aunque de forma menos clara que en el primer tipo de instancias.

Llama la atención, que para las dos últimas listas (4 y 5) no se mejoran los resultados de la primera fase. En el capítulo anterior apartado 2.3, se observaba la bondad de las soluciones utilizando

los vecindarios de Or con k=∞ para problemas de carga y descarga; este hecho puede explicar que en la lista cinco, la mejor solución se de precisamente en la fase I que utiliza dicho vecindario. Con respecto a la lista cuatro, y a la vista de los resultados obtenidos, podemos afirmar que en problemas de carga y descarga, cuando el número de iteraciones de la fase I es muy elevado, el uso del conjunto de concentración no aporta mejoras en la solución obtenida.

En cualquier caso parece claro, aunque de forma general, la capacidad de mejorar buenas soluciones, que tiene el hecho "concentrar" la búsqueda en un conjunto de arcos pertenecientes a esas mismas soluciones.

Se confirma la idea de que considerando una lista de soluciones más grande se obtiene mejores resultados en la primera fase y también en la segunda; o mas concretamente: a mejores soluciones en la primera fase, mejores en la segunda.

En cuanto a los niveles de concentración parece que a menor tamaño de *CS* mejores soluciones, siendo los niveles 3 (8,33%) y 4 (5%) los mejores.

Finalmente se han generado otras 10 instancias de cada tipo para comparar GRASP y Concentración Heurística con los metaheurísticos diseñados en Pacheco J. y Delgado C.R., (1.998): *Temple Simulado* y *Búsqueda Tabú* (sólo fase básica).

	Temple Simulado	Búsqueda Tabú B.	GRASP-1.000 it (CH-Fase I)	Concentr. Heurística Fase II
N. 1	14.670	14.666	14.530	14.514
N. 2	14.742	14.585	14.501	14.471
N. 3	15.739	15.628	15.586	15.570
N. 4	15.241	15.114	15.174	15.123
N. 5	15.726	14.806	14.864	14.836
N. 6	14.582	14.534	14.524	14.468
N. 7	15.708	15.741	15.662	15.649
N. 8	14.466	14.466	14.490	14.450
N. 9	15.890	15.883	15.878	15.795
N. 10	13.649	13.428	13.384	13.394
Coste Medio	**15.041,30**	**14.885,10**	**14.859,30**	**14.827,00**

Fig. 3.4 Comparación con otros procedimientos (tipo de instancias I).

Se considera como criterio de parada en el algoritmo GRASP ejecutar 1.000 iteraciones; de esta forma sirve simultáneamente como Fase I en Concentración Heurística. Para la Fase II se establece un nivel de concentración del 10%. Los resultados se muestran en las tablas correspondientes a las figuras 3.4 (instancias del tipo I) y 3.5 (instancias del tipo II).

	Temple Simulado	Búsqueda Tabú B.	GRASP-1.000 it (CH-Fase I)	Concentr. Heurística Fase II
N. 1	6.774	6.921	6.622	6.581
N. 2	8.004	7.885	7.921	7.883
N. 3	9.297	9.319	9.242	9.227
N. 4	8.313	8.355	8.331	8.314
N. 5	9.334	9.473	9.436	9.365
N. 6	9.267	9.289	8.333	8.343
N. 7	8.291	8.352	8.131	8.133
N. 8	9.305	9.339	9.316	9.301
N. 9	8.284	8.306	8.242	8.215
N. 10	9.540	8.738	8.533	8.514
Coste Medio	**8.640,90**	**8.597,70**	**8.410,70**	**8.387,60**

Fig. 3.5 Comparación con otros procedimientos (tipo de instancias II).

Obsérvese como las soluciones obtenidas en la Fase II del algoritmo *Concentración Heurística* mejoran en la mayoría de los casos a las obtenidas en la Fase I (GRASP); además *Concentración Heurística* es el mejor de los 4 metaheurísticos propuestos tanto para la mayoría de las instancias como en resultados medios.

Búsqueda Tabú es quizá el metaheurístico más prometedor para el VRP y el que se implementa con mayor frecuencia. ¿Qué

resultado daría el uso de un Conjunto de Concentración, como se hace en este trabajo, en la fase de Intensificación de Búsqueda Tabú?.

En este capítulo se propone un Metaheurístico para el problema de rutas con ventanas de tiempo, carga y descarga simultánea y flota heterogénea, basado en procesos de *Búsqueda Tabú* y *Concentración Heurística*. El procedimiento básico es Búsqueda Tabú; los fundamentos de Concentración Heurística se utilizan en la fase de intensificación de Búsqueda Tabú, en la que se define un *Conjunto de Concentración*. Para chequear la eficacia de este procedimiento de Intensificación, así como de todo el algoritmo, se usan diferentes librerías disponibles en la red.

Los vecindarios utilizados son los que hemos denominado tipo Or y tipo Gendreau-Clark, descritos en el capítulo 2. Se ha preferido no usar los Cross-Intercambios para evitar tiempos de computación excesivos.

El chequeo de la factibilidad y la valoración de cada intercambio puede suponer un tiempo de computación desmesurado. En los trabajos de Pacheco J. y Delgado C.R. (1.996) y (1.997-b) 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.

3.2 Algoritmo Básico de Búsqueda Tabú.

Sea $k \in N$ un número natural, el algoritmo que se propone básicamente actúa de la forma siguiente:

Procedimiento Búsqueda_Tabú_Básico(k).

Leer solución inicial s_0 *y hacer* $s^* = s_0$; (3.3)

Hacer $T = \varnothing$, *niter* = 0, *kiter* = 0 (3.4)

Repetir

niter:=niter+1; (3.1)

Seleccionar $s \in N_2(s_0) / s \notin T$ *o* $f(s) < f(s^*)$ *(criterio de aspiración) con* $f(s)$ *mínimo*

Hacer $s_0 = s$

Ejecutar Búsqueda_Local_Or *(k, r') y* Búsqueda_Local_Or *(k, r'')*

donde r' y r'' son las dos rutas de s_0 *modificadas* (3.5)

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

Actualizar T (3.2)

Hasta niter - kiter $\geq$ *maxiter*

Se denota por s^* a la mejor solución encontrada, y a s_0 como la solución actual en cada momento. T, como ya se sabe, es el conjunto de movimientos tabús y se obtiene determinando que conjunto de soluciones tienen ciertos *atributos tabús activos*; se

han de definir estos atributos tabús y durante cuantas iteraciones van a permanecer activos (y por tanto las soluciones que les contienen). Los contadores *niter* y *kiter* indican respectivamente el número de iteraciones ejecutadas hasta el momento y el número de la iteración en la que se encontró la mejor solución obtenida hasta el momento.

Cualquier movimiento vecinal, supone la incorporación de un conjunto de arcos y la eliminación de otros. 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*). De esta forma se evita que en las iteraciones siguientes se vuelva a soluciones ya visitadas. La definición de atributo tabú activo, sólo se aplica a los arcos eliminados en los cambios de elementos entre rutas, pero no se aplica a los arcos eliminados en las mejoras de cada una de las dos rutas implicadas (ejecución *Busqueda_Local_Or(k, s_0)* para r' y r'').

Para identificar que atributos tabús (arcos) van a estar activos se define *arco_tabu* como una matriz $n \cdot n$ de la siguiente forma:

arco_tabu(r, l) = n^o de la última iteración en la que fue eliminado el arco (r, l).

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

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

siendo *maxiter_tabu* el número de iteraciones que permanece activo como atributo tabú desde que es eliminado de la ruta actual. Se va a dar al parámetro *maxiter_tabu* un valor igual a $2 \cdot n^{1/2}$.

Inicialmente se define:

- $arco\_tabu(r, l) = 0$ para cada arco (r, l) en la solución inicial,
- $arco\_tabu(r, l) = - maxiter\_tabu$ (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 $T = \varnothing$.

Para el criterio de parada se toma $maxiter = 10 \bullet n$.

El algoritmo de Búsqueda Tabú básico puede ser complementado y ampliado, como ya se vio en el capítulo uno, 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*).

3.3 Fase de Intensificación: Uso de un Conjunto de Concentración.

Como ya se sabe, en esta fase se *intensifica* la exploración de las regiones y los vecindarios donde se hallan las mejores soluciones encontradas en la fase inicial (*Algoritmo Básico*) con la esperanza de lograr aún ligeras mejoras. Esta fase puede ser diseñada de diferentes formas. En este caso nos inspiramos en los trabajos de Rossing K.E. (1.997), Rossing K.E. y Revelle C.S. (1.997) y Rosing K.E. y otros (1.998) sobre *Concentración Heurística.*

Con los elementos (arcos) de las mejores soluciones obtenidas en el Algoritmo Básico se construye un Conjunto de Concentración; para ello se utiliza el siguiente procedimiento:

Procedimiento Construcción_Conjunto_de_Concentracion.

Ordenar las soluciones obtenidas hasta el momento en una lista según el valor de f (comenzando con la mejor)

Hacer $i = 0$ *y* $CS = \varnothing$

Repetir

Hacer $i = i+1$

Añadir los elementos de la i-ésima solución de la lista a CS
Hasta Cardinal(CS) ≥ num_arcos

El procedimiento *Búsqueda_Local_Hacia_CS* es similar al diseñado en el apartado anterior, punto 3.1.2, con la salvedad de que en este caso se utilizan los denominados *vecindarios tipo Gendreu-Clark* (N_2).

Procedimiento Búsqueda_Local_Hacia_CS(k, s_0).
Hacer $s^* = s_0$.
Repetir
Hacer coste_anterior $= f(s^*)$.
Repetir
Buscar $s' \in N_2(s_0)/f1(s') = \{\min f1(s) / s \in N_2(s_0)\}$.
Buscar $s'' \in N_2(s_0)/f(s'') = \{\min f(s) / s \in N_2(s_0)\}$.
Ejecutar en s' Búsqueda_Local_Or(k, r') *y* Búsqueda_Local_Or (k, r'') *donde r' y r'' son las dos rutas modificadas de* s_0.
Si $f1(s') < f1(s_0)$ *entonces* $s_0 = s'$.
Ejecutar en s'' Búsqueda_Local_Or(k, r') *y* Búsqueda_Local_Or (k, r'') *donde r' y r'' son las dos rutas modificadas de* s_0.
Si $f(s'') < f(s^*)$ *entonces* $s^* = s''$.
Hasta $f1(s') \geq f1(s_0)$.
Hacer $s_0 = s^*$.
Hasta $f(s^*) =$ *coste_anterior.*

Este procedimiento se inserta en la fase de Intensificación que queda de la siguiente forma:

Procedimiento Intensificación(k).

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 la ultima ejecución de* Búsqueda_Tabú_Básico(k).

Ejecutar Búsqueda_Local_Ge(k, s_i).

Para cada ruta r de s_i *ejecutar* Busqueda_Local_Or (∞, f, r)

{obviamente si $k=\infty$ *este paso no es necesario}.*

Ejecutar Búsqueda_Local_Hacia_CS(k, s_i).

Si $f(s_i) < f(s^*)$ *hacer* $s^* = s_i$.

Para este trabajo se ha tomado *num_soluciones* = 50. En cuanto al valor del parámetro *num_arcos* se toma el 10% del total de arcos (es decir, $num\_arcos = 0{,}1 \cdot n \cdot (n\text{-}1)$).

3.4 Fase de Diversificación.

Una vez finalizada la fase de Intensificación se ejecuta la fase de Diversificación, cuyo objetivo es dirigir el proceso de búsqueda a regiones no exploradas hasta este momento. Esto se hace

básicamente "penalizando" los elementos más visitados durante el proceso. Se van a definir dos matrices de frecuencias:

- *freq_corto (i, j)* = número de veces que ha aparecido el elemento (i, j) en la última ejecución *de Búsqueda_Tabú_Básico* e *Intensificación.*

- *freq_largo (i, j)* = número de veces que ha aparecido el elemento (i, j) hasta el momento en todo el proceso.

Durante una serie de iteraciones se van a realizar movimientos vecinales considerando estas matrices de frecuencia como matrices de distancias, y teniendo en cuenta, como en el procedimiento básico, la calificación de atributos tabú para evitar ciclos. Más concretamente, se define:

- *freq (i, j)* = *freq_corto (i, j)* si en la ultima ejecución de *Búsqueda_Tabú_Básico* o de *Intensificación* ha habido mejora en $f(s^*)$;

- *freq (i, j)* = *freq_largo (i, j)* en caso contrario.

y a continuación:

- $d2\ (i,j) = freq\ (i,j)$ si $(i,j) \notin T$

- $d2\ (i,j) = \infty$ si $(i,j) \in T$

se define la función g como la función de costes f, pero considerando el coste de los arcos el dado por $d2$ en vez de por d. De esta forma la fase de diversificación queda como sigue:

Procedimiento Diversificación(k).

Hacer s_0 como la última solución obtenida en el procedimiento básico; $T = \varnothing$,

Ejecutar maxiter_div veces las iteraciones del procedimiento Búsqueda_Tabú_Básico(k) *(línea 3.1 a línea 3.2) considerando la función g en vez de la función f.*

En este caso se ha tomado $maxiter\_div = 5 \cdot n$; s^* denota la mejor solución encontrada durante todo el proceso. Al añadir las fases de Intensificación y Diversificación el Algoritmo Búsqueda Tabú queda de la siguiente forma:

Algoritmo Búsqueda_Tabú_Principal(k).

Leer solución inicial s_0 y hacer $s^* = s_0$. (3.6)

Hacer: iter_global = 0, mejor_iter = 0.

Repetir

Hacer iter_global = iter_global+1; costeanterior = $f(s^*)$.

Ejecutar Búsqueda_Tabú_Básico(k).

Ejecutar Intensificación(k).

Ejecutar Diversificación (k).

Si $f(s^*) <$ costeanterior entonces hacer

mejor_iter = iter_global
Hasta iter_global - mejor_iter ≥ maxiter_global

La introducción de la línea (3.6) en este procedimiento hace innecesaria la línea (3.3) en el procedimiento Búsqueda Tabú Básico.

El criterio de parada se alcanza cuando transcurren una serie de iteraciones globales (*Búsqueda_Tabú _Básico(k)* + *Intensificación(k)* + *Diversificación(k)*) sin mejora en el valor de $f(s^*)$.

3.5 Oscilación Estratégica.

Opcionalmente se puede permitir la visita de soluciones no factibles, de forma controlada, para dar más flexibilidad al proceso y no dejar que se "encajone" en una determinada región en torno a un mínimo local, facilitando el acceso a otras regiones.

En el algoritmo propuesto, esta opción se da en la fase Básica, pero no en las otras dos. Para ello se redefine la función f de la siguiente forma:

$f(s) = f(s) + coef\_carga \cdot \mathrm{QS}(s) + coef\_tiempo \cdot \mathrm{TS}(s)$, para $s \in S^*$

donde :

- S^* = conjunto de soluciones factibles y no factibles (con respecto a la capacidad de los vehículos y las ventanas de tiempo) del problema, obviamente $S \subset S^*$;
- QS(s) = suma, en el conjunto de las rutas de s, de la cantidad de mercancía en que cada ruta de s sobrepasa la capacidad del vehículo (de máxima capacidad);
- TS(s) = suma, en el conjunto de las rutas de s, del máximo retraso de tiempo que se da en cada ruta de s;
- *coef_carga* y *coef_tiempo* = coeficientes de penalización para las violaciones de las restricciones de carga y tiempo respectivamente;

Así mismo se deben considerar vecindarios de soluciones tanto factibles como infactibles y para ello es necesario hacer los siguientes cambios:

- se debe sustituir N_2 por N^*_2, y
- en la ejecución de *Busqueda_Local_Or(k, s_0)* , N^k_1 por N^{*k}_1

Para controlar la visita a soluciones no factibles se van a seguir dos estrategias o criterios:

1. Si la solución actual s_0 es infactible, sólo se permite el cambio a una solución que reduzca su infactibilidad.

2. Variar el valor de *coef_carga* y *coef_tiempo* cada 10 iteraciones de la siguiente forma:

   - definir *ncarga* y *ntiempo* respectivamente como número de iteraciones en que $QS(s_0) > 0$ y $TS(s_0) > 0$ en esas 10 últimas;
   - si *ncarga* > 0 hacer *coef_carga* = *coef_carga**2;
   - si *ncarga* = 0 hacer *coef_carga* = *coef_carga*/2;
   - de igual forma si *ntiempo* > 0 hacer *coef_tiempo* = *coef_tiempo**2;
   - si *ntiempo* = 0 hacer *coef_tiempo* = *coef_tiempo*/2;
   - inicialmente *coef_carga* y *coef_tiempo* toman un valor igual a 1.

Esta idea de variar los coeficientes de penalización ha sido tomada del trabajo de Gendreu M. y otros (1.991). El objetivo de estas estrategias es mantener el proceso "cercano" a la región factible *S*. Cuando el proceso sale de *S* se le obliga a volver disminuyendo la infactibilidad. Si esta disminución es lenta, se la acelera aumentando los coeficientes de penalización. Por otra parte si el proceso lleva varias iteraciones en *S* se favorece que pueda visitar regiones infactibles disminuyendo estos coeficientes.

Una característica de este método es la preferencia por soluciones factibles o por aquellas que disminuyan la infactibilidad a la hora de explorar vecindarios de soluciones infactibles. En este sentido hay que indicar que también se ha probado con otras

estrategias, en las que no se da ninguna preferencia a las soluciones factibles ni al grado de infactibilidad, excepto la que lleva implícita en la función penalizada; un interesante trabajo al respecto es el de Díaz y Fernández (1.998) para el Problema de Asignación Lineal Generalizada. El problema que se ha encontrado al aplicar estas estrategias al VRPTW es que además de restricciones de carga, hay restricciones de tiempo, y la proporción de soluciones factibles visitadas en el conjunto de iteraciones era bajo. Por eso se ha optado por "forzar" aún más a que el proceso busque soluciones en el conjunto de soluciones factibles *S*.

3.6 Resultados Computacionales.

Para chequear la eficacia del algoritmo propuesto se han utilizado dos conocidas librerías de instancias: TSPLIB de Reinelt G. (1.994) para el CVRP, y Solomon Instances para el VRPTW (página WEB de *APES Group* de la Universidad de StrathClyde: *http:// www.cs.strath.ac.uk /~apes*.).

Cada una de estas instancias se ha resuelto utilizando 4 variantes del algoritmo propuesto. Estas variantes son el resultado de combinar la utilización o no, por una parte de la fase de

Intensificación y por otra parte de la Oscilación Estratégica, dando lugar a:

- variante sin intensificación y sin oscilación;
- variante sin intensificación y con oscilación;
- variante con intensificación y sin oscilación;
- variante con intensificación y con oscilación.

En todos los casos, se ha utilizado como número de iteraciones sin mejora *maxiter_global* = 4, y el valor del parámetro $k = \infty$. Para obtener la solución inicial se usa un algoritmo de *inserción de máxima diferencia* descrito en el apéndice 3. Los algoritmos se han programado en PASCAL, utilizando los compiladores BORLAND PASCAL 7.0 y BORLAND DELPHI 3.0.

3.6.1 TSPLIB/CVRP.

Esta librería se puede encontrar en una página mantenida por G. Reinelt (1.994), *http:// www.iwr.uni_heildeberg.de / iwr / comopt / soft / TSPLIB95 / CVRP*. Se muestran los resultados de las 4 variantes, en cuanto a número de vehículos, distancia y tiempo de computación, y en la última columna la mejor solución registrada hasta la fecha, teniendo en cuenta el número de vehículos utilizado y la distancia recorrida. Según la descripción dada en la página, se trata de minimizar la distancia independientemente del número de

vehículos. Se usan distancias euclídeas redondeadas a enteros, excepto para *eil7*, *eil13* y *eil31* que viene dada explícitamente. El ordenador usado en este caso es un PC Pentium - MMX 200 Mhz.

En la tabla de la figura 3.6 se resumen los resultados obtenidos en cada una de las 4 variantes en cuanto a número de vehículos, distancia y tiempos de computación; figura también la mejor solución registrada a fecha de realización de las pruebas (finales de 1.999).

Se ha adoptado la siguiente nomenclatura:

- (*): iguala la actual mejor solución;
- (**): mejora la mejor solución previa;
- **en negrita**: nueva mejor solución;
- en sombreado: nuestra mejor solución.

Instancia	Sin Intensificación		Con Intensificación		Previa Mejor Solución
	Sin Oscilación	Con Oscilación	Sin Oscilación	Con Oscilación	
eil7	2 - 104 (*) 0"02	2 - 104 (*) 0"02	2 - 104 (*) 0"02	2 - 104 (*) 0"02	2 - 104
eil13	4 - 247 (*) 0"27	4 - 247 (*) 0"32	4 - 247 (*) 0"26	4 - 247 (*) 0"32	4 - 247
eil22	4 - 375 (*) 7"56	4 - 375 (*) 4"23	4 - 375 (*) 2"72	4 - 375 (*) 4"29	4 - 375
eil23	3 - 569 (*) 1"81	3 - 569 (*) 2"01	3 - 569 (*) 1"81	3 - 569 (*) 2"01	3 - 569
eil30	**4 - 503 (**)** 16"98	**4 - 503 (**)** 6"81	**4 - 503 (**)** 52"16	**4 - 503 (**)** 6"80	4 - 534
eil31	**7 - 379 (**)** 49"61	7 - 388 (**) 95"23	**7 - 379 (**)** 100"90	**7 - 379 (**)** 147"11	7 - 1212
eil33	4 - 835 (*) 35"81	4 - 835 (*) 9"12	4 - 835 (*) 18"10	4 - 835 (*) 9"12	4 - 835
eil51	5 - 521 (*) 150"22	5 - 521 (*) 30"18	5 - 521 (*) 71"49	5 - 521 (*) 30"37	5 - 521

Instancia	Sin Intensificación		Con Intensificación		Previa Mejor Solución
	Sin Oscilación	Con Oscilación	Sin Oscilación	Con Oscilación	
eilA76	10 - 832 (*) 1237"29	10 - 836 1275"74	10 - 836 1205"91	**10 - 831 (**)** 3766"89	10 - 832
eilB76	15 - 1032 110"86	15 - 1026 (**) 2895"47	**14 - 1023 (**)** 1618"56	15 - 1030 (**) 865"19	15 - 1031
eilC76	8 - 739 637"08	8 - 737 2342"22	8 - 736 865"69	8 - 735 (*) 1990"69	8 - 735
eilD76	7 - 688 524"03	7 - 686 788"52	**7 - 682 (**)** 3369"25	**7 - 682 (**)** 1502"38	7 - 683
EilA101	8 - 818 4602"75	8 - 818 7305"13	8 - 818 2610"48	**8 - 815 (**)** 3541"50	8 - 817
EilB101	14 - 1076 (**) 174"69	14 - 1079 1873"77	14 - 1076 (**) 174"71	**14 - 1075 (**)** 4591"39	14 - 1077

Fig. 3.6 Resultados Computacionales para TSPLIB/VRP.

3.6.2 Instancias de Solomon para el VRPTW.

Esta librería se ha obtenido a través de la página WEB de *APES Group* de la Universidad de StrathClyde: *http:// www.cs.strath.ac.uk/~apes*. Estas instancias son como las usadas por Solomon (1.987). En total son 56 que se dividen en 6 clases: C1, C2, R1, R2, RC1, RC2. El tamaño de todos los problemas es de 100 clientes o puntos de visita (más el origen) e incluyen ventanas de tiempo, capacidad máxima y tiempo de descarga. En C1 y C2 los clientes vienen distribuidos en grupos o clusters. En R1 y R2 los puntos se distribuyen aleatoriamente de forma uniforme. RC1 y RC2 mezclan clientes distribuidos de forma aleatoria y por conglomerados. En las clases C1, R1 y CR1 se usan más vehículos que en las clases C2, R2 y CR2.

Se trata de minimizar el número de vehículos (primer objetivo) y la distancia (segundo). Se usan distancias euclídeas reales. A continuación se muestran los resultados para cada clase. El ordenador utilizado es un PC Pentium II a 350 Mhz. Se adopta la siguiente notación:

- (*) iguala la actual mejor solución;
- (**) mejora la mejor solución previa;
- **en negrita** nueva mejor solución;
- (+) Nueva mejor solución real (sin truncar ni redondear)
- en sombreado, nuestra mejor solución.

Clase C1	Sin Intensificación		Con Intensificación		Previa Mejor Solución
Instancia	Sin Oscilación	Con Oscilación	Sin Oscilación	Con Oscilación	Referencia
c101	10 - 828.94 (*) 0"11	10 - 828.94 (*) 0"66	10 - 828.94 (*) 0"17	10 - 828.94 (*) 0"65	10 - 828.94 RT
c102	10 - 828.94 (*) 1"21	10 - 828.94 (*) 5"32	10 - 828.94 (*) 1"21	10 - 828.94 (*) 5"39	10 - 828.94 RT
c103	10 - 828.06 (*) 81"46	10 - 828.06 (*) 1609"65	10 - 828.06 (*) 56"46	10 - 828.06 (*) 211"57	10 - 828.06 RT
c104	10 - 827.14 869"36	10 - 839.00 36"85	10 - 825.54 978"39	10 - 827.55 696"07	10 - 824.78 RT
c105	10 - 828.94 (*) 0"22	10 - 828.94 (*) 1"05	10 - 828.94 (*) 0"22	10 - 828.94 (*) 1"10	10 - 828.94 RT
c106	10 - 828.94 (*) 0"28	10 - 828.94 (*) 1"32	10 - 828.94 (*) 0"27	10 - 828.94 (*) 1"27	10 - 828.94 RT
c107	10 - 828.94 (*) 0"11	10 - 828.94 (*) 0"44	10 - 828.94 (*) 0"06	10 - 828.94 (*) 0"44	10 - 828.94 RT
c108	10 - 828.94 (*) 76"62	10 - 828.94 (*) 17"35	10 - 828.94 (*) 106"94	10 - 828.94 (*) 17"30	10 - 828.94 RT
c109	10 - 828.94 (*) 14"28	10 - 828.94 (*) 710"30	10 - 828.94 (*) 94"31	10 - 828.94 (*) 325"38	10 - 828.94 RT

Fig 3.7 Instancias de Solomon. Resultados Computacionales para la Clase C1.

Clase R1	Sin Intensificación		Con Intensificación		Previa Mejor Solución Referencia
Instancia	Sin Oscilación	Con Oscilación	Sin Oscilación	Con Oscilación	
r101	19 - 1671.61 41"41	19 - 1653.76 1149"26	19 - 1650.80 317"85	19 - 1654.93 707"88	19 - 1607.7 Des
r102	18 - 1482.06 15"76	17 - 1495.13 655"65	17 - 1493.79 272"16	17 - 1498.66 1261"41	17 - 1434 Des
r103	14 - 1221.16 1080"27	14 - 1233.24 573"92	14 - 1221.03 649"11	13 - 1312.21 2802"91	13 - 1207 Tha
r104	10 - 1002.16 306"32	10 - 1001.61 606"32	10 - 991.55 794"39	10 - 1010.32 689"70	9 – 1007.31 G
r105	15 - 1361.23 271"17	14 -1377.11 (*) 1397"03	15 - 1360.78 353"50	14 - 1377.11 (*) 437"43	14 - 1377.11 RT
r106	13 - 1251.45 1103"12	12 - 1264.19 1841"71	13 - 1251.15 1401"31	12 - 1264.19 645"43	12 - 1252.03 RT
r107	11 - 1076.93 724"08	10 - 1137.10 429"46	11 - 1074.65 1114"22	10 - 1130.91 457"42	10 - 1104.66 G
r108	10 - 953.20 213"44	9 - 1003.95 2310"82	10 - 953.20 285"12	9 - 972.34 1435"98	9 - 963.99 G
r109	12 - 1155.94 344"77	12 - 1157.16 2385"16	12 - 1153.02 501"80	12 - 1157.16 1147"40	11 - 1197.42 G

Clase R1 Instancia	Sin Intensificación		Con Intensificación		Previa Mejor Solución Referencia
	Sin Oscilación	Con Oscilación	Sin Oscilación	Con Oscilación	
r110	12 - 1093.25 75"75	11 - 1106.67 3571"92	12 - 1088.67 126"94	11 - 1084.01 4570"90	11 - 1080.36 RT
r111	11 - 1059.52 1155"19	10 - 1128.64 1382"53	11 - 1057.37 847"17	10 - 1070.65 2407"22	10 - 1096.73 G
r112	10 - 965.02 352"45	10 - 977.36 1633"21	10 - 963.72 461"27	10 - 967.06 5093"02	10 - 953.63 RT

Fig 3.8 Instancias de Solomon. Resultados Computacionales para la Clase R1.

Clase RC1 / Instancia	Sin Intensificación		Con Intensificación		Previa Mejor Solución Referencia
	Sin Oscilación	Con Oscilación	Sin Oscilación	Con Oscilación	
rc101	15 - 1638.71 392"60	15 - 1639.42 2824"76	15 - 1637.40 822"13	15 - 1637.40 1533"25	14 - 1669 Tha
rc102	14 - 1461.33 1090"38	13 - 1500.33 925"33	14 - 1461.23 1494"97	13 - 1490.47 1580"20	12 - 1554.75 TBGGP
rc103	12 - 1313.56 219"64	11 - 1289.81 926"65	12 - 1311.94 315"76	11 - 1318.85 799"22	11 - 1110 Tha
rc104	10 - 1182.05 58"06	10 - 1162.68 128"80	10 - 1167.81 113"59	10 - 1157.34 315"49	10 - 1135.83 G
rc105	15 - 1567.54 523"55	14 - 1563.54 1621"07	15 - 1565.94 705"90	14 - 1552.01 3109"60	13 - 1643.38 TBGGP
rc106	12 - 1416.57 1541"65	12 - 1388.22 4961"26	13 – 1380.44 1846"59	12 - 1386.72 3805"02	11 - 1448.26 TBGGP
rc107	11 - 1249.93 751"76	11 - 1243.67 3866"37	11 - 1243.67 1436"08	11 - 1245.70 989"26	11 - 1230.54 TBGGP
rc108	11 - 1144.05 377"29	10 - 1304.14 1492"22	11 - 1124.92 1602"42	10 - 1214.35 1174"75	10 - 1139.82 TBGGP

Fig 3.9 Instancias de Solomon. Resultados Computacionales para la Clase RC1.

Clase C2 / Instancia	Sin Intensificación		Con Intensificación		Previa Mejor Solución Referencia
	Sin Oscilación	Con Oscilación	Sin Oscilación	Con Oscilación	
c201	3 - 591.56 (*) 0"93	3 - 591.56 (*) 2"97	3 - 591.56 (*) 0"99	3 - 591.56 (*) 2"97	3 - 591.56 RT
c202	3 - 591.56 (*) 229"70	3 - 591.56 (*) 16"03	3 - 591.56 (*) 70"03	3 - 591.56 (*) 16"04	3 - 591.56 RT
c203	3 - 591.17 (*) 4"07	3 - 591.17 (*) 13"73	3 - 591.17 (*) 4"06	3 - 591.17 (*) 13"67	3 - 591.17 RT
c204	3 - 597.26 841"57	3 - 594.07 2746"87	3 - 594.07 1056"88	3 - 594.64 1196"83	3 - 590.60 RT
c205	3 - 588.88 (*) 5"82	3 - 588.88 (*) 5"98	3 - 588.88 (*) 5"82	3 - 588.88 (*) 5"99	3 - 588.88 RT
c206	3 - 588.49 (*) 8"07	3 - 588.49 (*) 50"47	3 - 588.49 (*) 8"13	3 - 588.49 (*) 50"42	3 - 588.49 RT
c207	3 - 588.29 (*) 1"32	3 - 588.29 (*) 3"46	3 - 588.29 (*) 1"32	3 - 588.29 (*) 3"51	3 - 588.29 RT
c208	3 - 588.32 (*) 1"21	3 - 588.32 (*) 3"24	3 - 588.32 (*) 1"21	3 - 588.32 (*) 3"24	3 - 588.32 RT

Fig 3.10 Instancias de Solomon. Resultados Computacionales para la Clase C2.

Clase R2 / Instancia	Sin Intensificación		Con Intensificación		Previa Mejor Solución Referencia
	Sin Oscilación	Con Oscilación	Sin Oscilación	Con Oscilación	
r201	5 - 1193.54 1465"69	4 - 1256.65 3167"55	5 - 1189.41 1824"24	**4 - 1253.26 (**)** 3464"24	4 - 1254.09 G
r202	5 - 1043.96 1274"38	**3 - 1204.50 (**)** 2705"41	5 - 1041.10 1588"61	3 - 1220.51 1972"65	3 - 1214.28 TBGGP
r203	4 - 896.84 815"09	3 - 959.02 1676"39	4 - 895.85 1100"10	3 - 950.76 3920"09	3 - 948.74 RT
r204	3 - 763.31 3450"42	2 - 853.86 (**) 4850"25	3 - 753.42 2606"05	**2 - 853.52 (**)** 2264"14	2 - 867.33 G
r205	4 - 965.03 1358"36	3 - 1028.22 3480"68	4 - 961.37 1504"30	3 - 1023.99 4294"84	3 - 998.72 G
r206	4 - 906.32 866"95	3 - 939.41 5916"63	4 - 887.71 2554"70	3 - 922.75 6787"15	3 - 833 Tha
r207	3 - 834.29 2420"68	2 - 950.29 (**) 7769"01	3 - 824.59 4505"65	**2 - 923.55 (**)** 6787"15	3 - 814.78 RT
r208	3 - 720.36 6866"90	2 - 740.86 10776"05	3 - 719.05 8103"05	**2 - 736.86 (**)** 4453"97	2 - 738.60 RT
r209	5 - 861.61 1013"55	3 - 923.85 9477"76	5 - 861.61 1196"28	3 - 923.81 (+) 1612"67	3 - 855 Tha

Clase R2 Instancia	Sin Intensificación		Con Intensificación		Previa Mejor Solución Referencia
	Sin Oscilación	Con Oscilación	Sin Oscilación	Con Oscilación	
r210	4 - 933.17 647"02	3 - 969.16 806"80	4 - 930.45 842"29	3 - 967.69 1313"49	3 - 963.37 G
r211	4 - 760.25 223"11	2 - 967.38 6412"50	4 - 760.25 223'33	3 - 792.06 1874"27	2 - 923.80 TBGGP

Fig 3.11 Instancias de Solomon. Resultados Computacionales para la Clase R2.

Clase RC2 / Instancia	Sin Intensificación		Con Intensificación		Previa Mejor Solución Referencia
	Sin Oscilación	Con Oscilación	Sin Oscilación	Con Oscilación	
rc201	5 - 1321.38 359"27	4 - 1425.21 25"04	5 - 1324.66 567"82	4 - 1419.27 1648"92	4 - 1249 Tha
rc202	5 - 1130.22 687"66	4 - 1165.44 929"01	5 - 1120.82 1739"05	**4 - 1161.79 (**)** 985"64	4 - 1164.25 TBGGP
rc203	4 - 970.47 1086"37	3 - 1092.11 5662"11	4 - 969.62 1444"54	3 - 1083.03 2786"20	3 - 1068.07 G
rc204	4 - 792.64 1073"79	3 - 836.29 1384"56	4 - 792.64 1265"10	3 - 814.35 1261"15	3 - 803.90 G
rc205	5 - 1240.98 200"04	4 - 1310.03 769"51	5 - 1238.70 336"42	4 - 1311.22 1180"51	4 - 1302.42 G
rc206	4 - 1081.86 159"78	3 - 1170.90 2040"53	4 - 1081.86 160"10	3 - 1188.28 2018"89	3 - 1156.26 G
rc207	4 - 1009.61 209"65	3 - 1097.56 1033"64	4 - 1008.93 289"79	3 - 1085.60 5195"84	3 - 1074.10 G
rc208	4 - 802.39 77"99	3 - 850.01 1640"35	4 - 796.42 1048"91	3 - 839.70 4119"52	3 - 833.97 RT

Fig 3.12 Instancias de Solomon. Resultados Computacionales para la Clase RC2.

Para la mejor solución de referencia (finales 1.999), se indica el trabajo que la aportó; la nomenclatura utilizada es la siguiente:

- Tha = Thangiah S.R., Osman I.H. and Sun T. (1.994).
- RT = Rochat Y. and Taillard E.D. (1.995).
- PB = Potvin J.Y and Bengio S. (1.994).
- G = GreenTrip Project: http:// www.cs.strath.ac.uk / ~apes.
- Des = Desrochers M., Desrosiers J. and Solomon M.M. (1.992).
- TBGGP = Taillard E.D., Badeau P., Gendreau M., Guertin F. and Potvin J.Y. (1.997).

En las tablas anteriores hay que indicar lo siguiente: en Desrochers M., Desrosiers J. y Solomon M.M. (1.992), las distancias fueron truncadas al primer decimal, y en Thangiah S.R., Osman I.H. y Sun T. (1.994) el valor de la solución final fue redondeado al entero más próximo. Por otra parte, algunas soluciones que son factibles con distancias truncadas o redondeadas pueden no serlo con distancias reales o viceversa. De ahí la aparentemente excesiva diferencia entre los valores obtenidos en este trabajo y el de estas referencias. También se ha de indicar el uso de variables reales durante la ejecución de los algoritmos; posteriormente se ha observado que resulta una ejecución más rápida utilizando valores enteros (truncando al segundo o tercer decimal y multiplicando por 100 o 1000), y

comprobando la factibilidad con valores reales iniciales, como se hace en Taillard E.D. y otros (1.997).

En la siguiente tabla (figura 3.13) se comparan los resultados medios con los de otras referencias según clases. Se indican los valores medios de número de vehículos y de distancia.

En este caso, la nomenclatura utilizada es la que sigue:

- CR = Chiang W.C. y Russell R.A. (1.993).
- KPS = Kilby P., Prosser P. y Shaw P. (1.997). Trabajo integrado en GREENTRIP Project.
- Variante 1 = Método propuesto. Sin Intensificación ni Oscilación Estratégica.
- Variante 2 = Método propuesto. Sin Intensificación y con Oscilación Estratégica.
- Variante 3 = Método propuesto. Con Intensificación y sin Oscilación Estratégica.
- Variante 4 = Método propuesto. Con Intensificación y Oscilación Estratégica.

	C1	R1	RC1	C2	R2	RC2
CR	10,00 885,86	12,42 1.289,95	12,38 1.455,82	3,00 658,88	2,91 1.135,14	3,38 1.361,14
PB	10,00 838,01	12,58 1.296,80	12,13 1.446,20	3,00 589,93	3,00 1.117,70	3,38 1.360,57
Tha	10,00 832	12,33 1.238	12,00 1.284	3,00 650	3,00 1.005	3,38 1.229
RT	**10,00** **828,38**	12,25 1.208,50	11,88 1.377,39	**3,00** **589,86**	2,91 961,72	3,38 1.119,59
TBGGP	**10,00** **828,38**	**12,17** **1.209,35**	**11,50** **1.389,22**	**3,00** **589,86**	2,82 980,27	3,38 1.117,44
KPS	10,00 830,75	12,67 1.200,33	12,12 1.388,15	3,00 592,24	3,00 966,56	3,38 1.133,42
Variante 1	10,00 828,64	12,91 1.191,12	12,50 1.371,72	3,00 590,69	4,00 898,06	4,38 1.043,69
Variante 2	10,00 829,96	12,33 1.211,32	12,00 1.386,48	3,00 590,29	**2,73** **981,20**	3,38 1.118,44
Variante 3	10,00 828,46	12,83 1.188,31	12,63 1.361,67	3,00 590,29	4,00 893,16	4,38 1.041,71
Variante 4	10,00 828,68	12,33 1.208,30	12,00 1.375,35	3,00 590,36	2,82 960,79	**3,38** **1.112,91**

Fig 3.13 Resultados medios para las Instancias de Solomon para el VRP.

3.7 Análisis de los Resultados.

Analizamos las tablas correspondientes a las figuras 3.6 a 3.12 desde dos perspectivas:

- En primer lugar los resultados que hemos obtenido con cada una de las cuatro variantes para comprobar la utilidad o no tanto de la oscilación estratégica como de la fase de intensificación.
- En segundo lugar comparamos las soluciones obtenidas con las mejores soluciones conocidas en el momento de la realización de las pruebas.

3.7.1 Eficacia de la Fase de Intensificación y de la Oscilación Estratégica .

Es claro que con el uso de Oscilación Estratégica (variantes 2 y 4) se consiguen mejores resultados que sin ella, especialmente en las Instancias de Solomon. Solamente en 2 casos, C104 y R109, el mejor resultado de las cuatro variantes no se da en las variantes 2 o 4. Especialmente esta mejora se observa en la reducción de los vehículos a usar con respecto a las variantes 1 y 3. De hecho en la

mayoría de los casos se consigue obtener el número de vehículos de la mejor solución obtenida hasta la fecha (excepto en R104, R109, RC101, RC102, RC105 y RC106).

En cuanto a los problemas de la librería TSPLIB/CVRP también se dan los mejores resultados con Oscilación Estratégica (excepto para eilB76). En cualquier caso quizás la diferencia no sea tan clara como con las instancias de Solomon (se consiguen excelentes resultados con la Variante 3 que no usa Oscilación Estratégica). Quizás por el hecho de que al no haber restricciones de tiempo, las regiones factibles no estén tan aisladas entre sí.

En cuanto al uso de Intensificación se pueden hacer dos análisis: Variante 1 frente a Variante 3, y Variante 2 frente a Variante 4. En el primer caso esta claro la eficacia del uso de la fase de Intensificación: solamente en 2 instancias de Solomon (RC106 y RC201) se obtienen, y muy ligeramente, mejores soluciones con la Variante 1 (por el criterio de parada usado).

En el segundo caso también es claro, aunque menos, la eficacia del uso de la fase de Intensificación. Se consiguen mejores soluciones en 27 casos con la variante 4 que con la 2, y 10 mejores soluciones con la 2 (por el criterio de parada); de estas, en solamente 2 la diferencia se puede considerar relativamente significativa (RC103 y R211).

Parecido análisis se puede realizar con los problemas de la librería TSPLIB/CVRP: solamente con la instancia eilA76 la variante 1 consigue mejor solución que la 3 y con la eilB76 la variante 2 alcanza un resultado superior a la 4.

También es interesante observar los tiempos de computación usados en cada caso: en muchos problemas con el uso de la fase de Intensificación se obtienen soluciones, al menos parecidas que sin ella, en un tiempo de computación menor. De todas formas, cualquier conclusión en este sentido debe ir acompañada de un análisis más completo de la evolución de los tiempos de computación.

3.7.2 Comparación con otros resultados.

En la librería TSPLIB/CVRP, y centrándonos en las 6 últimas instancias (con un tamaño aceptable y que no han sido resueltas de forma exacta), la conclusión es clara: se mejoran las soluciones anteriores en 5 de las 6 instancias, y se iguala en la otra (eilC76).

En las clases R1 y RC1 de las instancias de Solomon se consiguen los resultados más pobres comparándolos con las mejores referencias hasta el momento, Rochat Y. and Taillard E.D. (1.995), y Taillard E. y otros (1.997); aunque son aceptables si se comparan con las demás referencias.

En la clase C1 y C2 las 4 variantes analizadas dan resultados muy parecidos entre ellas y parecidos a las mejores referencias anteriores.

En la clase R2 es donde las variantes 2 y 4 consiguen los mejores resultados: se establecen 5 nuevas mejores soluciones: una en la variante 2, (R202) y cuatro en la variante 4 (R201, R204, R207 y R208), a las que habría que añadir (R209) si se consideran sólo distancias reales sin truncar ni redondear. También ambas mejoran los mejores resultados medios anteriores en cuanto a número de vehículos y distancia: 2,73 en la variante 2 y 2,82 vehículos con 960,79 en la variante 4.

En la clase RC2 también se consiguen resultados buenos: se establece una nueva mejor solución con la variante 4 para RC202, dándose en el resto de las instancias, resultados en general muy cercanos a la mejor solución previa. En cuanto a resultados medios: el número de vehículos requeridos por las variantes 2 y 4, 3,38 en ambos casos, iguala al resto de las referencias; la distancia de la variante 3 (1118,44) es similar a la de las mejores referencias obtenidas hasta el momento, Rochat Y. and Taillard E.D., (1.995), y Taillard E. y otros (1.997), siendo mejoradas ligeramente por la variante 4, (1112,91).

En resumen se pueden establecer las siguientes conclusiones:

- Los resultados, sin ser espectaculares, son interesantes: se mejoran los mejores resultados previos en 5 de las 6 instancias de mayor tamaño de TSPLIB/CVRP, y se iguala la otra; se mejoran en 7 instancias (considerando distancias reales) de Solomon igualándose en 16, y acercándose en la mayoría.

- En cuanto al tiempo de computación, en algunos casos puede parecer aparentemente alto. Sin embargo hay que recordar que se han usado ordenadores personales y no estaciones de trabajo especializadas en cálculo, que sería lo ideal en estos casos.

- Es claro el efecto positivo tanto de la Oscilación Estratégica como la de la fase de Intensificación. En cualquier caso, hay que recordar que en el actual trabajo, en la fase de Intensificación sólo se visitan soluciones factibles. En el futuro sería interesante analizar que ocurre si se permitiera también en esta fase la visita de soluciones no factibles.

- También se tiene previsto el uso de vecindarios más amplios como el propuesto en Taillard y otros (1.997) que en nuestra adaptación hemos denominado *vecindario de intercambio entre dos rutas generalizado*, que como ya se ha comentado, extienden el propuesto en este diseño.

En cualquier caso insistir en el efecto positivo que puede tener el uso de un Conjunto de Concentración, con los elementos de las mejores soluciones, para el diseño de una fase de Intensificación en Búsqueda Tabú, como se ha expuesto.

Los experimentos son totalmente reproducibles al no haber componentes aleatorios en los programas y, en cualquier caso, estos quedan a disposición de las personas interesadas (versión PASCAL). A continuación se realizará una descripción del uso del algoritmo descrito a problemas reales de Transporte Escolar (Pacheco J. y otros (2.000)).

En el apéndice 4 se muestran las nuevas mejores soluciones obtenidas.

CAPITULO IV

APLICACIÓN AL TRANSPORTE ESCOLAR

4.1 Descripción del Problema.

El problema real es el del transporte de un conjunto de alumnos, que han de ser recogidos en una serie de localizaciones distribuidas geográficamente, y trasladados a su correspondiente centro de enseñanza. Siguiendo la notación empleada en capítulos anteriores, *1* es el punto correspondiente al centro de enseñanza y *{2, 3, ..., n}* los puntos correspondientes a las localizaciones donde los alumnos han de ser recogidos. El número de alumnos a recoger en cada punto i ($i = 2, ..., n$) viene dado por $q(i)$.

Los alumnos no deben permanecer más de un determinado tiempo máximo en ruta, lo que se denota por *tmax*, y la ruta debe finalizar antes del inicio de las clases en el instante *tinicio*. Se conocen las distancias d_{ij} y tiempos t_{ij} entre cada par de puntos $i, j \in \{1, 2, ..., n\}$. La legislación autoriza diferentes tipos de vehículos, con diferente número de plazas; 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 con 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 impone la restricción de que los alumnos de una misma localidad

sean transportados por el mismo vehículo. En el caso de que en una localización el número de alumnos supere la máxima capacidad de todos los tipos de vehículo, se tratará dicha localización como si fueran dos diferentes: una con un número de alumnos igual a dicha capacidad y otra con el resto. De cualquier forma, 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. El Ministerio de Educación y Cultura, a través de la Secretaría General de Educación y Formación Profesional, sugirió (13 de diciembre de 1.997) una fórmula que serviría de referencia en el cálculo de las cantidades necesarias para las contrataciones de las rutas del transporte escolar. Estas cantidades (en pesetas) vienen descompuestas en los tres apartados antes mencionados (kilómetros, alumnos y paradas) de la siguiente forma:

a) Coste por kilómetros:

$$Pr * k \qquad \text{si } k \leq 35$$

$$Pr * 35 + (k\text{-}35) * 1{,}33 * Pr \qquad \text{si } k > 35$$

donde Pr es el precio de referencia por kilómetro que oscila entre 125 y 163 (en función de la calidad del vehículo) y k el número de kilómetros. La cantidad por este concepto no podría exceder de 14.250 pesetas.

b) Coste por alumnos:

$$100 * (M - 33) \qquad \text{si } M > 33$$

donde M es el número de alumnos.

c) Coste por paradas:

$$75 * (L1 - 6) \qquad \text{si } L1 > 6$$

$$0 \qquad \text{si } L1 \leq 6$$

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

Sin embargo, los responsables en cada provincia han de negociar las cantidades por diferentes sistemas de tarifas. Se paga por kilómetro recorrido, y dicho precio por kilómetro depende del número de alumnos transportados en la ruta. En cualquier caso, la fórmula anterior supone una buena aproximación y es la que se utilizará en este trabajo para calcular el coste de transporte.

En el diseño de la estrategia de solución hay que tener en cuenta dos datos importantes:

- la distancia recorrida comienza a contar desde el primer punto de recogida y no desde el punto desde el que sale el autobús[1];
- no hay un coste fijo en cada ruta por el tipo de vehículo utilizado.

---

[1] A partir de ahora $d_{1i} = t_{1i} = 0$ para $i = 2, .., n$

Teniendo en cuenta esto, no va a ser necesario minimizar el número de vehículos a utilizar, como suele ser lo habitual en problemas de rutas; es más, se ha observado que en muchos casos aumentando el número de rutas se pueden llegar a soluciones menos costosas. 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 sean más baratas que otras con menos. En la figura 4.1 se puede observar como la solución con las rutas A - Colegio y B - Colegio, recorre menos distancia que la solución con la ruta A - B - Colegio.

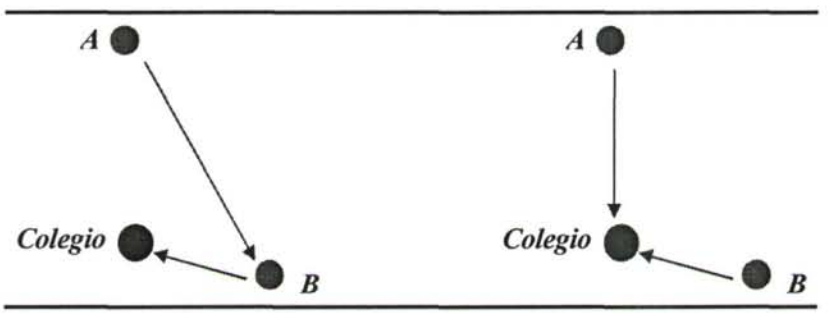


Fig. 4.1 Ruta única (A - B - Colegio) y dos rutas (A - Colegio; B - Colegio).

El modelo así considerado es un caso particular del conocido Problema de Rutas de Vehículos con Ventanas de Tiempo (VRPTW, Vehicle Routing Problem with Time Windows) descrito en el apéndice 1.5. Un trabajo clave sobre este modelo se debe a Solomon M.M.(1.987). Otra consideración que se podría hacer es que es un VRP con restricciones de duración de las rutas.

4.2 Modelización del Problema.

4.2.1 Algoritmo Inicial.

El algoritmo propuesto se compone de dos fases. En la primera se ejecutan varios procedimientos, que en la mayoría de los casos están basados en movimientos vecinales, y constituyen lo que denominamos *algoritmo inicial*; la solución obtenida en esta primera fase será posteriormente mejorada mediante un procedimiento de búsqueda tabú.

Sea $s_0 \in S$ la solución actual en cada momento, el Algoritmo Inicial puede escribirse en seudocódigo de la siguiente forma:

Procedimiento Algoritmo_Inicial (Output $s_0 \in S$)

Obtener una Solución Inicial s_0 por un método constructivo (4.1)

Ejecutar Búsqueda_Local_Ta (3, s_0) (4.2)

Para cada ruta r de s ejecutar* Búsqueda_Local_Or(∞, r) (4.3)

Ejecutar Búsqueda_Local_Or (∞, s_0) (4.4)

Para la obtención de una solución inicial se usa la adaptación para este modelo del algoritmo de Fisher M.L. y Jaikumar R. (1.981), explicada en el apartado 3.1.1 del capítulo tres; 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 B. y otros (1.980).

Tras el paso 4.2, cada una las rutas de la solución s_0 obtenida son óptimos locales con respecto al vecindario N^3_1, es decir, no son mejorables con recolocaciones de tipo Or de cadenas de hasta 3 elementos. Posteriormente se ejecuta el paso 4.3 por si las rutas de la solución s_0 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 4.4 se vuelve a ejecutar el mismo procedimiento *Búsqueda_Local_Or* con $k=\infty$ a la solución obtenida, pero considerada como una única cadena, con el objeto de analizar las posibles recolocaciones que afecten a rutas diferentes.

Para reducir el tiempo de computación a la hora de utilizar los vecindarios de Tailard, se ha empleado la estrategia de Búsqueda Local Rápida propuesta por Bentley J.L. (1.992), tal y como se ha descrito en el capítulo dos apartado 2.5.

4.2.2 Mejora con un Procedimiento de Búsqueda Tabú.

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

Las soluciones obtenidas en la primera fase pueden, en algunos casos, ser ligeramente mejoradas en la segunda. Como ya se ha comentado, se aplica un procedimiento basado en Búsqueda Tabú, en el que se implementan Búsqueda Tabú Básica e Intensificación, como se desarrollaron en el capítulo anterior pero utilizando vecindarios de Taillard. El algoritmo en pseudocódigo puede escribirse como sigue:

```
Procedimiento Búsqueda_Tabú_Principal
    Ejecutar Búsqueda_Tabú_Básico(3)
    Ejecutar Intensificación(3)
```

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 todas las soluciones del vecindario analizado independientemente de contener o no elementos tabú o de cumplir el criterio de aspiración.

Se podría añadir al procedimiento Búsqueda_Tabú_Principal una fase de Diversificación y volver a ejecutarlo una o varias veces; sin embargo, dado que las soluciones obtenidas son suficientemente satisfactorias, no se utilizará la estrategia de diversificación para evitar tiempos de computación excesivos.

4.2.3 Resultados Computacionales.

En este apartado se analizarán las diferentes rutas que se deben realizar en el transporte escolar de secundaria en la Provincia de Burgos. Cada una de ellas viene definida por:

- un Centro Escolar,
- las localizaciones donde se recogen los alumnos que estudian en ese centro, y
- el número de alumnos a recoger en cada una de esas localizaciones.

En todos los casos el tiempo máximo legal de un alumno en ruta, *tmax*, es de sesenta minutos, y la capacidad máxima de los

vehículos es Q=56 alumnos. Para el diseño de estas rutas se han utilizado los algoritmos anteriormente descritos. Para el cálculo de la matriz de distancias y del camino entre cada par de puntos se ha ponderado la distancia de cada tramo según el tipo de carretera, de forma que se favorezca la elección de carreteras nacionales antes que autonómicas, estas antes que carreteras sin revestimiento, etc...; en muchas ocasiones los caminos obtenidos no son los más cortos pero si los más cómodos y rápidos.

A continuación se muestra una tabla con los datos correspondientes a los 16 servicios de transporte escolar necesarios para trasladar a los alumnos a los 16 centros de secundaria correspondientes, y los resultados obtenidos. La información que aporta cada una de las columnas es la siguiente:

- población donde está cada centro;
- el número de localidades donde se recogen los niños que van a cada centro y debajo el número de niños que estudian en cada uno de los centros;
- los costes (en pesetas y teniendo en cuenta la función de costes descrita en el apartado 4.1) y número de vehículos utilizados en la realidad para trasladar a los estudiantes a cada uno de los centros;
- los resultados con los costes (teniendo en cuenta la función de costes descrita en el apartado 4.1), número de vehículos y tiempos de computación (en segundos) de los algoritmos descritos.

Pr.	Centro (Población)	n. loc. alumn	Solución Real	Algoritmo Inicial Paso 1	Paso 2	Paso 3	Paso 4	B. Tabú 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

Centro Pr. (Población)	n. loc. alumn	Solución Real	Algoritmo Inicial Paso 1	 Paso 2	 Paso 3	 Paso 4	B. Tabú Básico	 Intensif.
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 27,3**	**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**

Fig. 4.2 Transporte escolar en secundaria.

Los gráficos siguientes (figuras 4.3 y 4.4) muestran los costes y tiempos de computación de cada algoritmo. En el de costes se aprecia que el paso 2 supone la disminución más significativa y que la búsqueda Tabú también aporta reducciones importantes. El segundo gráfico muestra claramente los insignificantes tiempos de computación de cada una de las partes del Algoritmo Inicial, frente al empleado por la Búsqueda Tabú.

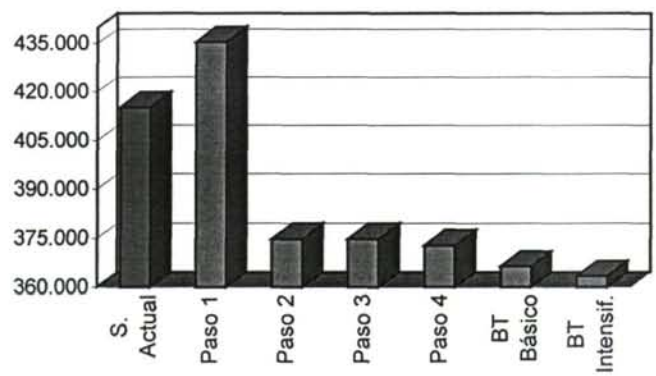


Fig. 4.3 Costes de las soluciones obtenidas (en pesetas).

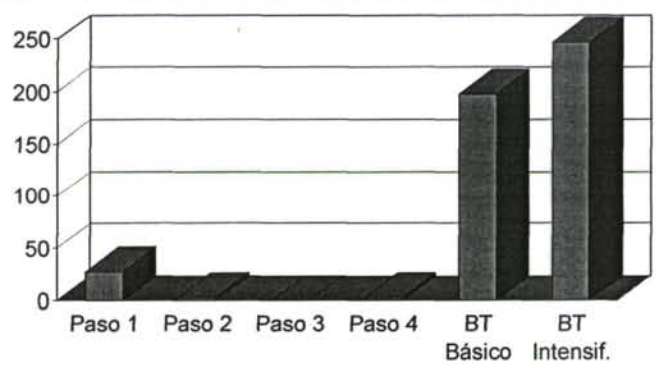


Fig. 4.4 Tiempos de computación (en segundos) empleados.

Se observa que el Algoritmo Inicial aporta soluciones rápidamente (algo menos de treinta y tres segundos para todos los problemas de secundaria) que mejoran el coste de las soluciones empleadas en más de un 10%. Utilizando un poco más de tiempo de computación (disponible en este caso), la Búsqueda Tabú aporta soluciones que mejoran algo más de un 2,5% las del Algoritmo Inicial, y casi un 12,6 % las soluciones actuales. Resultados similares se han registrado para primaria.

Es claro que los algoritmos propuestos (Inicial e Inicial + B.Tabú) aportan globalmente soluciones muy interesantes en cuánto a ahorros de costes. Obsérvese sin embargo que existen servicios o instancias, para las que esta mejora es escasa. Incluso en algún caso, los algoritmos descritos aparentemente aportan soluciones peores que las utilizadas en la realidad: Miranda y Villarcayo. Esto se debe a las siguientes circunstancias:

- Para el cálculo de los caminos, distancias y tiempos se ha usado una 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 algunas imprecisiones (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 la solución real, tienen en realidad una duración mayor que el máximo programado de 60 minutos, en algún caso hasta de 90 minutos, según responsables de la propia Dirección Provincial de Educación. 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. En nuestro caso, hemos supuesto unas velocidades moderadas para cada tipo de carretera, concretamente: 80 km/h. para Autopistas y Autovías, 70 para Nacionales, 60 para Autonómicas de 1er orden, 55 para Autonómicas de 2° orden, 50 para Autonómicas de 3er orden, 40 para Carreteras Sin Revestimiento, 30 para carreteras de Enlace y 20 para Travesías. Velocidades similares e incluso mayores, pueden conseguirse fácilmente en la realidad. Lo importante es que las soluciones planificadas teóricamente, puedan llevarse a cabo en la práctica con cierto margen. (Al contrario de lo que pasa en algunas soluciones usadas en la realidad).

Después de lo expuesto hasta ahora, es conveniente realizar la siguiente reflexión: 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, lo que socialmente es muy valioso tratándose del problema del transporte escolar. Esta es la esencia del siguiente apartado.

4.3 Problema Minmax.

4.3.1 Planteamiento.

El planteamiento para resolver el problema del transporte escolar, tal y como se ha descrito, tiene como objetivo *minimizar el coste del transporte*, respetando las limitaciones impuestas por la ley.

Los responsables del transporte escolar en la provincia de Burgos, insistieron en la necesidad de diseñar rutas que, además de ahorrar costes, acortaran lo máximo posible las duraciones actuales de las rutas. Esto se puso de manifiesto también por las Asociaciones de Padres de Alumnos (APAS), que protagonizaron diferentes manifestaciones de protesta, cuya reivindicación más importante era precisamente la reducción del tiempo que los escolares permanecían en el autobús. Información a cerca de este tema se publicó en varios artículos del Diario de Burgos en marzo de 2.000 (ver apéndice 5).

La necesidad social de que los tiempos de permanencia de los escolares en el autobús sea reducido al mínimo posible, obliga a ajustar el algoritmo a un *planteamiento más social*, siempre

teniendo en cuenta las limitaciones y disponibilidades del servicio de transporte escolar. La idea es minimizar la duración (tiempo) de la ruta más larga de cada uno de los 16 servicios de transporte. Se pretende determinar hasta dónde se puede llegar en el establecimiento de unos límites a la duración de las rutas (menor duración posible); se reduce la duración de las rutas y se elaboran rutas lo más equilibradas posible.

4.3.2 Implementación.

La estrategia propuesta para conseguir este objetivo se implementa en dos fases. En la primera fase (fase A) se resuelve el problema "social", estableciéndose el nuevo tiempo máximo de duración de las rutas; dicho tiempo es usado como restricción para la resolución del problema económico (fase B):

Algoritmo Planteamiento_Social_del_Transporte_Escolar.

Ejecutar el procedimiento adaptado al problema minmax (fase A).
Ejecutar el procedimiento económico con el nuevo tiempo máximo de duración (tmax) obtenido en la fase A (fase B).

Para la implementación del *procedimiento adaptado al problema minmax*, tomamos como base el procedimiento económico y actuamos como se indica en los siguientes párrafos.

Sea:
- $s \in S$ una solución,
- s' una solución $\in N^{\infty}_{2}(s)$ resultado de un intercambio,
- $r1$ y $r2$ las rutas de s implicadas en dicho intercambio,
- $r1'$ y $r2'$ las nuevas rutas a que da lugar y
- $\Delta(s') = \max\{tpo(r1), tpo(r2)\} - \max\{tpo(r1'), tpo(r2')\}$, donde $tpo(r)$ se define como la duración de la ruta r.

Δ va a ser la medida de la "bondad" de cada intercambio s' de $N^{\infty}_{2}(s)$. Un aspecto interesante es que se van a contemplar intercambios entre todos los pares de rutas y no sólo en los que interviene la ruta más duradera. En otras palabras, se permiten cambios aparentemente innecesarios, puesto que no disminuyen la función objetivo (duración de la ruta más larga). Sin embargo entendemos que estos intercambios son interesante por las siguientes razones:

- permiten explorar entornos mayores,
- evitan un "estancamiento" rápido del algoritmo en mínimos locales próximos a la solución inicial, y
- favorecen el equilibrio global del conjunto de las rutas.

De esta forma, sea rf una ruta cualquiera y $k \in \mathbb{N}$ un número entero prefijado, se definen los siguientes procedimientos:

Procedimiento Búsqueda_Local_Or(k, r_0)

Repetir

Determinar $rf \in N^k_1(r_0)$ *verificando* $tpo(rf) = min\{tpo\ (r) / r \in N^k_1(r_0)\}$

Si $tpo\ (rf) < tpo\ (r_0)$ *entonces hacer* $r_0 = rf$

Hasta que $tpo\ (r) \geq tpo\ (r_0),\ \forall r \in N^k_1(r_0)$.

Así mismo sea $s_0 \in S$ una solución se define el siguiente

Procedimiento Búsqueda_Local_Ta(s_0)

Repetir

Determinar $s' \in N_2(s_0)$ *verificando* $\Delta f\ (s') = max\{\ \Delta f\ (s) / s \in N_2(s_0)\}$

Si $\Delta f\ (s') > 0$, *hacer:*

$s_0 = s'$

Ejecutar Búsqueda_Local_Or(3, *r'*) y Búsqueda_Local_Or (3, *r''*) *donde r' y r'' son las 2 rutas de* s_0 *modificadas para dar lugar a s'*

Hasta que $\Delta f\ (s) \leq 0\ ,\ \forall s \in N_2(s_0)$.

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

4.3.3 Pruebas simuladas.

A continuación se muestran los resultados de una serie de pruebas para comparar el funcionamiento del procedimiento *Búsqueda_Local_Ta* cuando se consideran todos los intercambios (por tanto se admiten cambios que no afectan a la ruta más duradera), y cuando se consideran solamente aquellos en los que una de las rutas implicadas es la más duradera (solo admiten cambios que mejoran la función objetivo). Además se consideraran las estrategias de Primer Descenso y Mayor Descenso vistas en el capitulo dos. De esta forma son cuatro las variantes a analizar.

Para ello se han generando instancias de 10 tamaños diferentes: 10 puntos, 20 puntos,... hasta 100 puntos (además del origen). Para cada tamaño se generan 20 instancias. Los datos de cada instancia se definen de la forma siguiente:

- Se asigna a cada punto del problema dos coordenadas x e y, cuyos valores son generados aleatoriamente con distribución uniforme entre 0 y 100. La distancia entre cada par de puntos se define como la distancia euclídea correspondiente. Los tiempos (en minutos) de trayecto se toman igual a la distancia (en kilómetros), es decir, consideramos una velocidad de 60 km/h.

- A cada $q(i)$, $i = 2, ..., n1+n2$, se le asigna un valor entero generado uniformemente de forma aleatoria entre 1 y 10.
- Se supone una capacidad de los vehículos Q igual a 20.

Las cuatro variantes utilizan la misma solución inicial obtenida por una adaptación a este modelo de un algoritmo de *inserción* que se describe en el apéndice 3. La correspondencia es la siguiente:

- variante 1: intercambios que implican sólo a la ruta más larga y estrategia de Mejor Descenso;
- variante 2: intercambios que implican todas las rutas y estrategia de Mejor Descenso;
- variante 3: intercambios que implican sólo a una ruta y estrategia de Primer Descenso;
- variante 4: intercambios que implican todas las rutas y estrategia de Primer Descenso.

En la tabla siguiente se muestran para cada tamaño de problema, los resultados medios de la función f (duración de la ruta más larga), la duración media de todas las rutas, y los tiempos de computación en segundos. El número de vehículos disponibles es el obtenido por el algoritmo de inserción. Todos los procedimientos descritos y programas usados para la realización de este trabajo han sido programados usando los compiladores BORLAND PASCAL (7.0) y BORLAND DELPHI (3.0). El equipo usado es un Ordenador Personal PENTIUM II 300 mhz.

Tamaño	Algoritmo Inserción	Variante 1	Variante 2	Variante 3	Variante 4
11 R.L.:	**229,450**	**178,350**	**176,700**	**175,550**	**175,850**
T.R.:	173,108	161,979	155,888	158,508	157,696
T.C.:	*0,002*	*0,004*	*0,004*	*0,01*	*0,005*
21	**248,300**	**178,100**	**176,700**	**177,900**	**177,100**
	173,387	158,460	150,951	157,654	150,103
	0,011	*0,004*	*0,023*	*0,01*	*0,023*
31	**253,700**	**191,550**	**189,550**	**191,150**	**188,400**
	175,832	165,876	152,163	165,622	151,136
	0,017	*0,012*	*0,055*	*0,01*	*0,046*
41	**271,600**	**193,150**	**190,900**	**193,050**	**191,800**
	178,821	167,021	150,766	166,210	154,785
	0,037	*0,022*	*0,097*	*0,02*	*0,077*
51	**261,300**	**189,050**	**186,500**	**189,300**	**187,650**
	181,447	164,112	150,583	161,330	150,749
	0,067	*0,034*	*0,117*	*0,03*	*0,084*
61	**268,550**	**188,350**	**186,250**	**189,850**	**186,350**
	179,557	162,896	145,709	165,256	147,158
	0,080	*0,043*	*0,194*	*0,03*	*0,135*
71	**271,050**	**187,650**	**186,350**	**187,200**	**186,350**
	179,077	160,537	144,348	162,329	143,400
	0,123	*0,055*	*0,276*	*0,050*	*0,187*

Tamaño	Algoritmo Inserción	Variante 1	Variante 2	Variante 3	Variante 4
81	**276,400**	**199,000**	**198,300**	**200,950**	**198,150**
	185,059	170,833	151,984	174,350	153,952
	0,180	*0,063*	*0,378*	*0,04*	*0,239*
91	**268,400**	**189,750**	**188,100**	**190,450**	**188,100**
	182,438	164,383	145,309	164,315	146,665
	0,257	*0,088*	*0,505*	*0,060*	*0,290*
101	**272,300**	**197,300**	**195,800**	**197,050**	**196,100**
	183,836	166,876	146,312	169,140	146,494
	0,347	*0,093*	*0,640*	*0,050*	*0,399*

Fig. 4.5 Duración de la ruta más larga (R.L. en negro); duración media de todas las rutas (T.R.); tiempo de computación (T.C. en itálica).

Los resultados anteriores se muestran gráficamente en las figuras 4.6 y 4.7; la primera muestra la duración de la ruta más larga, mientras que la segunda la duración media de todas las rutas.

Se comprueba que los resultados obtenidos considerando intercambios de todas las rutas (variantes 2 y 4), además de mejores resultados en cuanto a duración de la ruta más larga, mejoran los resultados de duración media de todas las rutas. Este efecto es muy importante e interesante para el problema del transporte escolar.

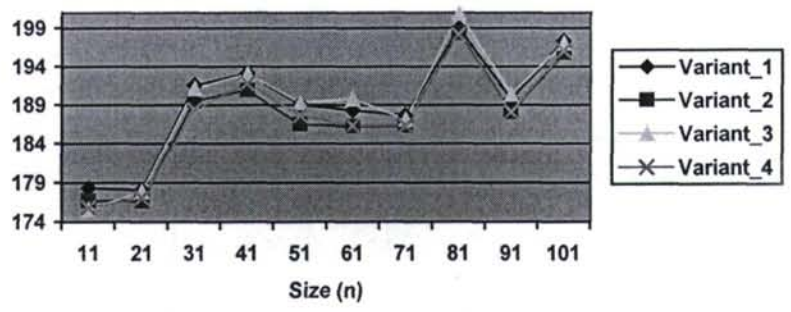


Fig. 4.6 Duración de la ruta más larga.

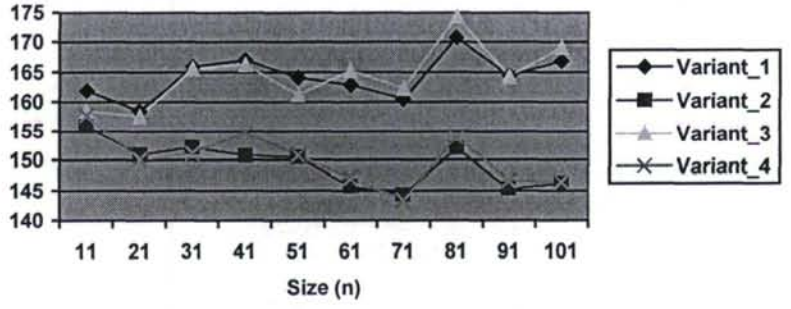


Fig. 4.7 Duraciones medias de todas las rutas.

La variante 2 obtiene mejores resultados que la variante 4: en doce resultados la variante 2 es mejor que la 4, en dos son iguales y en seis es mejor la 4 que la 2.

4.3.4 Resultados computacionales.

Se ejecuta el algoritmo considerando intercambios entre todas las rutas y la estrategia de Mejor Descenso, que como se ha visto en el apartado anterior, es la variante que mejores resultado aporta.

En la tabla correspondiente a la figura 4.8, se muestran:

- las duraciones de las rutas más largas para cada uno de los 16 servicios de transporte, tanto en la fase A como en la B, denotado como M.D.;
- el coste resultante en la fase B (según la función de costes descrita anteriormente) denotado por C., y
- los tiempos de computación, denotado por T.C.

Evolución de los resultados de la fase A:

Como se ve en la tabla de resultados (figura 4.8), en la Parte I se consiguen soluciones de forma rápida (poco más de 21 segundos para todos los problemas de secundaria) que rebajan sustancialmente la media de la duración de la ruta más larga para cada uno de los 16 servicios: de 60,62 minutos a poco más de 41.

Centro	Solución Aplicada	FASE A					FASE B	
		Parte 1			Búsqueda Tabú			
		P 1	P 2	P 3	Básico	Intens.		
1	70	*M.D.* 60 *T.C. 6,42*	51 *0,39*	51 *0,05*	48 *44,82*	48 *29,28*	*C.* 66.792,13 *T.C. 123,86*	*M.D.* 48
2	45	46 *0,65*	29 *0,06*	29 *0*	29 *5,27*	29 *5,71*	17.962,50 *12,52*	29
3	60	60 *0,77*	47 *0,11*	47 *0*	45 *5,22*	45 *4,06*	25.160,00 *12,09*	45
4	70	53 *0,22*	36 *0,05*	36 *0*	36 *2,69*	36 *2,64*	21.212,50 *5,11*	36
5	60	55 *0,27*	45 *0,11*	45 *0*	44 *6,43*	44 *4,06*	16.291,13 *9,45*	43
6	80	60 *0,99*	51 *0,22*	51 *0*	47 *11,97*	47 *9,28*	31.743,38 *25,71*	45
7	60	60 *1,04*	38 *0,17*	38 *0,05*	37 *17,3*	37 *12,97*	26.845,25 *40,59*	37
8	75	60 *3,57*	51 *0,38*	51 *0*	51 *43,5*	51 *27,9*	41.753,63 *101,29*	51
9	90	60 *1,97*	53 *0,28*	53 *0*	53 *23,07*	52 *16,42*	31.373,88 *72,39*	52

Centro	Solución Aplicada	FASE A					FASE B	
		Parte 1			Búsqueda Tabú			
		P 1	P 2	P 3	Básico	Intens.		
10	**60**	58 *0,5*	41 *0,11*	41 *0*	41 *4,06*	41 *3,68*	19.975,00 *8,02*	41
11	**60**	55 *0,11*	45 *0*	45 *0*	45 *1,04*	45 *2,26*	21.338,88 *2,96*	45
12	**25**	15 *0*	9 *0*	9 *0*	9 *0,22*	9 *0,72*	3.312,50 *0,71*	9
13	**45**	53 *0,66*	32 *0,16*	32 *0*	29 *11,87*	29 *8,57*	22.000,00 *11,80*	29
14	**60**	60 *0,39*	53 *0,05*	53 *0*	48 *5,39*	48 *3,9*	19.191,00 *11,97*	46
15	**50**	59 *1,48*	42 *0,06*	42 *0*	40 *12,91*	40 *6,7*	30.812,50 *15,32*	40
16	**60**	55 *0,06*	40 *0*	40 *0*	38 *0,65*	38 *1,32*	13.526,13 *1,27*	38
Totales	**970**	**869** ***19,1***	**663** ***2,15***	**663** ***0,1***	**640** ***196,41***	**639** ***139,47***	**409.290,41** ***455,05***	**634**
Medias	**60,62**			**41,43**		**39,93**		**39,62**

Fig 4.8 Tabla: duraciones de las rutas más largas.

Si se dispone de más tiempo de computación, la búsqueda Tabú consigue reducir estos tiempos a menos de 40 minutos.

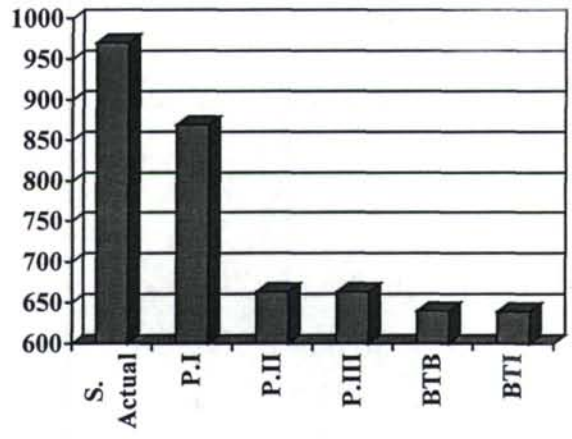


Fig 4.9 Suma de la duración de las rutas más largas (en tiempo) para cada solución resultante en la fase A.

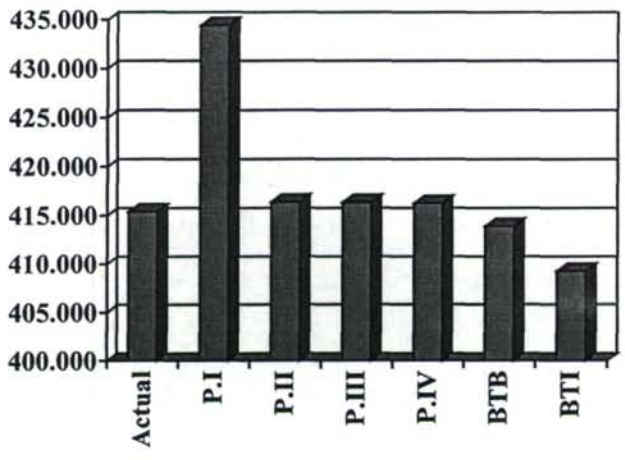


Fig. 4.10 Costes resultantes en la fase B.

Evolución de los resultados de la fase B:

En la fase B se consigue rebajar el coste de las soluciones reales en un 1,5% (figura 4.10) con unas condiciones en cuanto a la duración de las rutas mucho mejores que las empleadas en la realidad. La suma de las duraciones de las rutas más largas asciende a 634 minutos, menos de 40 minutos de media.

4.3.5 Utilización de diferentes números de autobuses: comparación con otros metaheurísticos.

Corberán A. y otros (2.000) trabajan con estos mismos datos utilizando Scatter Search. La figura 4.11 muestra una tabla con los resultados obtenidos por dichos autores y los obtenidos en el presente trabajo.

Las soluciones que se aportan son el resultado de ejecutar el algoritmo utilizando el criterio minmax de optimización y fijando un determinado número de autobuses para cada servicio: desde m-1 hasta m+3 en algunas de las rutas, siendo m es el número de autobuses que se emplearon en la realidad. Inicialmente el algoritmo se ejecuta con la restricción de 60 minutos máximos de duración de las rutas, obteniéndose unas determinadas soluciones iniciales. En ocasiones para que el algoritmo no de soluciones

infactibles con respecto a la duración y finalice la ejecución, se ablanda dicha restricción y se utilizan 90 minutos (soluciones señaladas con (*)) o 120 (soluciones señaladas con (**)). Como se comprobará las soluciones así obtenidas, en ocasiones violan la norma legal de 60 minutos de duración máxima, pero nos sirven a efectos de comparación con las obtenidas utilizando la implementación de Scatter Search realizada por Corberán A. y otros (2.000).

Los resultados en los que inicialmente Búsqueda Tabú supera a Scatter Search se han escrito en negrita; aquellos en los que lo iguala están en letra normal y los que son peores en cursiva y subrayados. Se contabilizan inicialmente:

- 26 soluciones mejores,
- 28 soluciones iguales,
- 10 soluciones peores.

Una posterior ejecución de los 10 resultados que eran peores en Búsqueda Tabú que en Scatter Search, ablandando la restricción de duración de las rutas (de 60 a 90 o de 90 a 120 minutos) da lugar a una mejora en la comparativa a favor de Búsqueda Tabú:

- 31 soluciones mejores,
- 30 soluciones iguales,
- 3 soluciones peores.

	Solución Aplicada		Scatter Search/Tabu Search									
	Autobuses	Ruta más										
Problema	(m)	Larga	m-1		m		m+1		m+2		m+3	
S1	12	70	57	**53**	56	**48**	53	**51**	51	**48**	48	**47**
S2	5	45	46	46	36	36	32		29	29		
S3	6	60	54	**53**	46	**45**	43	*45* **39(*)**	39	39		
S4	3	70			52	52(*)	42	**41**	37	**36**	33	33
S5	4	60	55	55	44	**43**	39	*41*				
S6	4	80	81	**78(*)**	62	62(*)	53	**51**	47	**45**	44	**43**
S7	6	60	51	**49**	45	**44**	38	**37**	36			
S8	9	75	61	**59**	59	**50**	50	50	47	*50* **46(*)**	44	*45* **40(*)**
S9	5	90	82	**79(*)**	65	*70(*)* **62(**)**	53	**52**	50	*52*	48	*49* **45(*)**
S10	6	60	44	44	41	41	40	*41*				
S11	4	60	67	67(*)	51	51	45	45	39	39		
S12	2	25			15	15	14	14	9	9		
S13	6	45	39	39	34	**33**	32	**29**	29	*32* 29(*)		
S14	5	60	53	53	46	46	37	37				
S15	7	50	50	**47**	44	**42**	42	**40**	40	*42* ***40(*)***		
S16	2	60	84	84(*)	51	51	38	38	35	35		

Fig. 4.11 Comparación con otros Metaheurísticos.

4.3.6 Reflexiones sobre los resultados obtenidos.

Llama la atención el mayor número de vehículos usados en la solución obtenida por el algoritmo propuesto que el usado en la realidad, y por tanto, podrían hacerse las siguientes cuestiones: ¿por qué se ha considerado este mayor número de autobuses?, ¿hasta que punto se ha visto favorecido el coste de la solución obtenida por este mayor número? o en otras palabras ¿es posible mantener la calidad de las soluciones obtenidas considerando el número de vehículos usados en la realidad en cada problema?.

En cuanto a la primera cuestión, el uso de un mayor número de autobuses es debido a las siguientes razones: Esta cantidad de autobuses existe en la realidad, concretamente, alrededor de 120 autobuses realizan el transporte escolar de los alumnos de primaria (que entran a las 10 horas), por tanto, entendemos que se podría disponer de esta oferta. Por otra parte se han detectado 2 problemas, los correspondientes a los centros 6 y 9 (Diego de Siloé en Burgos y Medina de Pomar), donde con el número de autobuses disponibles en la realidad y con la restricción legal de 60 minutos de tiempo máximo de permanencia de los escolares en el autobús, no se puede obtener ninguna solución factible, al menos con los algoritmos propuestos en este trabajo (obsérvese los valores correspondientes en la tabla 4.11).

Centro	Solución Real	Algoritmo Inicial				B.Tabú	
		Paso 1	Paso 2	Paso 3	Paso 4	Básico	Intensific.
1	61177,5 12	66142,0 11 3	60395,88 12 0,219	60395,88 12 0,016	60395,88 12 0,062	60309,75 12 32,063	59015,50 12 21,844
2	21045,0 5	21724,88 5 0,141	17750,00 5 0,046	17750,00 5 0	17750,00 5 0,016	17750,00 5 4,5	17375,00 5 2,719
3	25051,3 6	24303,37 5 0,265	23404,00 6 0,031	23404,00 6 0	23404,00 6 0,016	23238,38 6 7,109	23238,38 6 2,782
4	18608,8 3	20740,38 3 0,016	18238,13 3 0,031	18238,13 3 0	18238,13 3 0	18238,13 3 1,64	18238,13 3 1,469
5	15902,5 4	14773,25 3 0,125	14614,88 3 0,016	14614,88 3 0	14614,88 3 0,015	14614,88 3 3,204	14614,88 3 2,562
7	34835,0 6	28667,25 5 0,688	25181,87 6 0,094	25181,87 6 0	23916,88 6 0,156	23766,38 6 17,672	23626,13 6 10,125
8	51112,5 9	51528,75 9 1,672	45925,50 9 0,297	45925,50 9 0	45782,25 9 0,156	44598,25 9 58,109	44531,63 9 18
10	19402,5 6	19758,88 4 0,157	18887,00 5 0,015	18887,00 5 0	18887,00 5 0,016	18847,88 6 2,875	18847,88 6 2,562

Centro	Solución Real	Algoritmo Inicial				B.Tabú	
		Paso 1	Paso 2	Paso 3	Paso 4	Básico	Intensific.
11	16622,54	19038,13 4 0,031	18893,75 4 0	18893,75 4 0	18893,75 4 0,016	18893,75 4 0,687	18893,75 4 0,688
12	8025,02	6925,00 2 0	6925,00 2 0	6925,00 2 0	6925,00 2 0	6925,00 2 0,157	6925,00 2 0,281
13	22975,06	26040,38 5 0,218	21000,00 6 0,063	21000,00 6 0	19975,00 6 0,047	19862,50 6 6,843	19862,50 6 5,907
14	21488,85	21371,13 5 0,171	18586,75 4 0,016	18586,75 4 0,016	18586,75 4 0	18512,50 4 4	18512,50 4 2,172
15	29088,87	29606,50 6 0,5	27843,50 7 0,047	27843,50 7 0	27843,50 7 0,015	25000,00 7 7,141	25000,00 7 4,453
16	8252,52	14385,88 2 0,016	10847,00 2 0	10847,00 2 0	10847,00 2 0	10847,00 2 0,406	10847,00 2 0,406
Totales	**353.587,77**	**365.005,78 69 7**	**328.493,26 74 0,875**	**328.493,26 74 0,032**	**326.060,02 74 0,515**	**321.404,40 75 146,406**	**319.528,28 75 75,97**

Fig 4.12 Soluciones para un número de autobuses igual al real (sin problemas 6 y 9).

En cuanto al tema del coste se han repetido las pruebas considerando como número máximo de vehículos los usados en la realidad para cada problema. Por lo comentado en el párrafo anterior no se realizan estas pruebas para los problemas correspondientes a los centros 6 y 9. En la tabla correspondiente a la figura 4.12 se muestra la evolución de los resultados obtenidos. Obsérvese que el ahorro en los costes totales sigue siendo destacable, se han reducido de 353.387 pesetas a 319.528, lo que supone un 9,63 % de ahorro empleando incluso dos autobuses menos (de 77 a 75). En cada celda se indica el coste y debajo el número de vehículos y el tiempo de computación.

Incluso en este sentido, se podría ir más lejos y determinar cual es el número mínimo de vehículos necesario para cada problema. Para ello se vuelven a repetir las pruebas (con todos los puntos) incorporando un coste ficticio por vehículo utilizado alto. De esta forma el algoritmo trata de minimizar el número de vehículos en primer lugar y la función de coste original en segundo.

Los resultado obtenidos son los que muestra la tabla de la figura 4.13. Obsérvese que de esta forma se ha conseguido reducir el número de vehículos de 86 a 71, es decir un 17,44% e incluso el coste sigue siendo menor que el real, bajando de 415.382,7 a 407.267,16 lo que supone una reducción del 1,95%. Como en el caso anterior, en cada celda se indica el coste y debajo el número de vehículos y el tiempo de computación.

Centro	Solución Real	Algoritmo Inicial				B.Tabú	
		Paso 1	Paso 2	Paso 3	Paso 4	Básico	Intensificación
1	61177,5	66325,63	63490,50	63490,50	63490,50	66123,00	65810,63
	12	10 3,016	10 0,172	10 0	10 0,062	9 61	9 19,047
2	21045,0	22960,50	21179,25	21179,25	21179,25	19015,38	19015,38
	5	4 0,14	4 0,032	4 0,015	4 0	4 5,469	4 2,906
3	25051,3	24303,37	25542,75	25542,75	25542,75	25542,75	25675,75
	6	5 0,375	4 0,031	4 0	4 0	4 3,329	4 3,187
4	18608,8	20740,38	18238,13	18238,13	18238,13	18238,13	18238,13
	3	3 0,063	3 0,046	3 0	3 0	3 2,032	3 1,906
5	15902,5	14773,25	14614,88	14614,88	14614,88	14614,88	14614,88
	4	3 0,141	3 0,015	3 0	3 0,016	3 5,094	3 2,937
6	30720,0	34831,63	30384,75	30384,75	30384,75	28097,75	27873,00
	4	5 0,469	5 0,14	5 0	5 0,016	5 22,109	5 7,516
7	34835,0	28667,25	26556,88	26556,88	26556,88	28567,63	28567,63
	6	5 0,703	5 0,078	5 0	5 0,016	4 21,312	4 9,36
8	51112,5	53328,88	51073,38	51073,38	51073,38	47356,63	47029,50
	9	8 1,687	8 0,078	8 0	8 0,047	8 48,797	8 19,172
9	31075	34834,88	31763,88	31763,88	31763,88	30813,63	30805,25
	5	6 0,937	6 0,156	6 0	6 0,032	6 35,531	6 12,203

Centro	Solución Real	Algoritmo Inicial				B.Tabú	
		Paso 1	Paso 2	Paso 3	Paso 4	Básico	Intensificación
10	19402,56	19758,88 4 0,172	19449,50 4 0,016	19449,50 4 0	19449,50 4 0,015	19387,25 4 2,719	19387,25 4 2,734
11	16622,54	19038,13 4 0,047	18893,75 4 0,015	18893,75 4 0	18893,75 4 0	18893,75 4 0,844	18893,75 4 0,906
12	8025,02	6925,00 2 0,016	6925,00 2 0	6925,00 2 0	6925,00 2 0	6925,00 2 0,188	6925,00 2 0,437
13	22975,06	29949,13 4 0,14	29949,13 4 0	29949,13 4 0	29949,13 4 0,016	26520,88 4 8,969	26903,38 4 5,593
14	21488,85	21510,25 4 0,188	18512,50 4 0,016	18512,50 4 0	18512,50 4 0,015	18512,50 4 3,328	18512,50 4 2,516
15	29088,87	29606,50 6 0,5	28481,00 6 0,046	28481,00 6 0,016	28481,00 6 0,016	28168,13 5 17,734	28168,13 5 4,906
16	8252,52	12873,00 2 0,016	10847,00 2 0	10847,00 2 0	10847,00 2 0	10847,00 2 0,562	10847,00 2 0,641
Totales	**415.382,786**	**440.426,66 75 8,61**	**415.902,28 74 0,841**	**415.902,28 74 0,031**	**415.902,28 74 0,251**	**407.624,29 74 239,017**	**407.267,16 71 95,967**

Fig. 4.13 Resultados de algoritmo minimizando primero el número de vehículos y luego el coste.

CAPITULO V

CONCLUSIONES

5.1 Conclusiones.

5.2 Nuevas Líneas de Investigación.

5.1 Conclusiones.

Son muchas las técnicas que podían haberse incluido en el **capítulo 1** de este trabajo y que no se han considerado por la necesidad de tener que parar en algún momento; alguna de esas técnicas será objeto de futuros trabajos. El reciente manual titulado *Nuevas Ideas en Optimización* (Corne D. y otros (1.999)), incluye colaboraciones de muchos de los más importantes investigadores y las últimas tendencias en el campo de los Metaheuríscos.

En el **capítulo 2** han sido objeto de estudio tres tipos diferentes de entornos para problemas de rutas con ventanas de tiempo y carga y/o descarga (VRPTW Mixto/Puro). El primero de ellos es una adaptación a problemas de rutas, del método de intercambio propuesto por Or I. (1.976) para el TSP; las soluciones del entorno se obtienen mediante modificaciones realizadas considerando la solución completa como una única secuencia de puntos: *variante de los intercambios r-óptimos* (*vecindarios tipo Or*). Los dos siguientes consideran la solución como un conjunto de rutas; las soluciones del entorno se obtienen mediante el intercambio de elementos entre pares de rutas: *intercambio entre dos rutas limitado* (*vecindarios tipo Gendreu-Clark*) e *intercambio entre dos rutas generalizado* (*CROSS-intercambios* o *vecindaros tipo Taillard*).

1. Se ha estudiado en que situaciones puede un vecindario ser más adecuado que otro. En general el que mejores resultados aporta es el *vecindario de Taillard*, seguido muy de cerca por el *vecindario tipo Gendreu-Clark*; las soluciones aportadas por el segundo son tan solo un 0,705% peores que las aportadas por el primero para instancias con puntos de carga y descarga simultánea (*problemas mixtos*) y un 0,182% para instancias con sólo puntos de carga o sólo de descarga (*problemas puros*). La ventaja del vecindario Gendreu-Clark con respecto al de Taillard es que el tiempo de computación se reduce en un 74,788% para problemas mixtos y en un 56,226% para problemas puros.

Dentro de la Búsqueda por Entornos, el método de referencia para las nuevas técnicas dadas a conocer a partir de los ochenta es la *Búsqueda Local*. Es importante destacar su importancia en muchos de los Metaheurísticos: *Búsqueda Local Guiada*, *GRASP*, *Concentración Heurística* (Fase I), *Búsqueda en Entorno Variable*, son en realidad repeticiones de procedimientos de Búsqueda Local (donde se modifica la función objetivo, la solución inicial, etc...); muchos de los *Algoritmos Meméticos* añaden a las operaciones de *Algoritmos Basados en Población* la Búsqueda Local; también puede aparecer en la fase de intensificación en *Búsqueda Tabú* y en *Búsqueda Dispersa*. Se estudian dos estrategias para elegir una solución mejor que la actual dentro del entorno: Mayor Descenso y Primer Descenso.

2. En nuestro caso se comprueba que la estrategia de *Mayor Descenso*, explorar todo el entorno y elegir la mejor solución encontrada, mejora ligeramente los resultados aportados por *Primer Descenso*, elegir la primera solución encontrada que mejore la actual. En concreto, primer descenso es tan solo un 0,211% peor que mayor descenso. La desventaja de Mayor Descenso es que el tiempo de computación que emplea es un 67,613% más que el empleado por primer descenso.

3. Se ha adaptado la estrategia de *Búsqueda Local Rápida* propuesta por Bentley J.L. (1.992) y utilizada también por Voudouris C. y Tsang E. (1.995) para el TSP. Esta estrategia permite reducir el tiempo de computación que se utiliza en la exploración de los vecindarios. Se trata de aprovechar el uso de memoria y ganar de esa forma en velocidad de cálculo. Con esta adaptación conseguimos reducciones medias en el tiempo de computación del 60,61 %. En concreto, los ahorros son mayores cuando se utiliza mayor descenso (67,46% frente a 53,76% en primer descenso), en problemas puros (65,04% frente a 56,18% en mixtos) y cuanto mayor es el tamaño del problema. Mejor Descenso continúa empleando más tiempo de computación que Primer Descenso pero la diferencia se reduce a un 24,07% más.

4. En este trabajo se añade una aportación a la *Búsqueda Local Rápida*, que consiste en guardar el resultado de determinados

cálculos para no tener que repetirlos. En este caso la mejora no se verifica para problemas de pequeño tamaño, pero si para problemas con 60 o más puntos de visita en el caso de problemas puros, y con 150 o más puntos de visita para problemas mixtos. (Se utiliza la estrategia de Mayor Descenso).

En el **capítulo 3** se desarrolla un algoritmo de *Búsqueda Tabú* que utiliza *vecindarios* descritos en el capítulo dos y que incorpora la filosofía de *Concentración Heurística* para la fase de intensificación.

5. Utilizando *GRASP* y *Concentración Heurística* se desarrolla, en el apartado 3.1, un algoritmo para el VRPTW Mixto. Dicho algoritmo permite comprobar la capacidad de mejorar soluciones, ya de por si buenas, que tiene el hecho "concentrar" la búsqueda en un conjunto de arcos pertenecientes a esas mismas soluciones.

6. Hay que destacar los buenos resultados que da el hecho de combinar por un lado la capacidad de diversificación que aporta la Búsqueda Tabú Básica, con la de intensificación que aporta la fase de intensificación basada en el uso del *conjunto de concentración*. Las pruebas realizadas en los problemas de TSPLIB/CVRP y en las instancias de Solomon muestran que tan sólo en 5 casos de los 70, la mejor solución la consigue una variante que no utiliza intensificación, en 28 se igualan y en 37 se consigue la mejor solución con una de las variantes que

emplea intensificación. Entendemos que el uso del conjunto de concentración, ya sea como se ha propuesto en este trabajo o de otra forma, es además de novedoso, una manera interesante y eficaz de intensificación que podría ser usada en otros tipos de problemas.

7. La *Oscilación Estratégica*, según las pruebas realizadas, se muestra también como una buena herramienta. En 36 casos de los 70 analizados la mejor solución la aporta una variante que utiliza Oscilación Estratégica, en 7 una que no la utiliza y en 27 casos coinciden las soluciones de ambos tipos de variantes. Lo más interesante es observar como su uso hace que el algoritmo sea más robusto: si nos fijamos en las soluciones, son muchos los casos en los que las variantes sin oscilación no consiguen igualar el número de vehículos de la mejor solución conocida, y sin embargo las variantes con oscilación lo igualan e incluso en algún caso lo mejoran.

8. En conjunto, señalar que el algoritmo desarrollado consigue mejorar los mejores resultados previos en 5 de las 6 instancias de mayor tamaño de TSPLIB/CVRP e igualar la otra; en cuanto a la librería de Solomon, mejora 7 instancias, iguala 16, y de las 33 restantes tan sólo en 2 del tipo R1 y 4 del tipo RC1 no consigue igualar el número de vehículos utilizados.

9. En el **capítulo 4** se diseña un algoritmo para la resolución del problema de transporte escolar de secundaria en la provincia de Burgos, que utiliza varios métodos de los vistos. En primer lugar se ha generado una solución que posteriormente es mejorada mediante diferentes procedimientos de *Búsqueda Local* estudiados y adaptados en el capítulo dos. A dicha solución se la aplica el procedimiento de *Búsqueda Tabú* diseñado en el capítulo tres y que como se ha dicho tiene la novedad de incorporar la filosofía de *Concentración Heurística* en la fase de intensificación.

Se han considerado dos objetivos: uno económico, con el propósito de minimizar el coste de transporte y otro social cuyo fin es minimizar la duración de la ruta más larga para cada uno de los 16 servicios de transporte que se realizan en secundaria (*problema minmax*).

10. Si atendemos únicamente al objetivo económico conseguimos un ahorro del 12,538% (18.228.245 pts.) usando un 20,93% de vehículos más que los usados en la realidad; estos vehículos están disponibles y se utilizan en primaria, donde el número de alumnos es superior. Si se resuelve el problema usando el mismo número de vehículos que los utilizados en la realidad, el ahorro es del 9,63 %; aunque algo inferior continua siendo importante. En este último caso no se han tenido en cuenta dos de los 16 problemas, que según nuestros cálculos no pueden cumplir la restricción de sesenta minutos fijada por la ley.

11. Si tenemos en cuenta el problema social, el algoritmo propuesto consta de dos fases: primero se ejecuta el procedimiento descrito anteriormente pero adaptado al problema minmax, según lo indicado en el apartado 4.3.2; después se ejecuta el algoritmo considerando el objetivo económico y teniendo en cuenta el nuevo tiempo máximo de duración obtenido en la fase anterior.

12. En la primera fase ha resultado muy fructífero el hecho de considerar todos los intercambios posibles entre pares de rutas y no sólo aquellos intercambios en los que interviene la ruta más duradera (que es la que se quiere minimizar). Se permiten cambios aparentemente innecesarios, puesto que no disminuyen la función objetivo (duración de la ruta más larga), pero que son interesantes por las siguientes razones:

    - permiten explorar entornos mayores,
    - evitan un "estancamiento" rápido del algoritmo en mínimos locales próximos a la solución inicial, y
    - favorecen el equilibrio global del conjunto de las rutas y reducen su duración global, lo cual es económica y sobretodo socialmente muy deseable.

Con respecto a los resultados obtenidos (punto 4.3.4) hay que tener en cuenta lo siguiente: la duración de las rutas empleadas en la realidad son en muchos casos mayores que las inicialmente

previstas por las autoridades, que son las que se han utilizado en este trabajo. Esto se desprende de las noticias que informaban sobre quejas y manifestaciones realizadas por las Asociaciones de Padres de Alumnos (apéndice 5). Sin embargo en nuestro caso, se han supuesto unas velocidades moderadas para cada tipo de carretera, lo que asegura que dichas rutas puedan llevarse a la práctica.

13. Se consigue el objetivo social, minimizar la duración de la ruta más larga en cada uno de los 16 servicios realizados en secundaria; además se consigue minimizar la duración media de todas las rutas. Se utiliza la función de costes que el Ministerio de Educación y Cultura, a través de la Secretaría General de Educación y Formación Profesional, sugirió (13 de diciembre de 1.997) para que sirviera de referencia en el cálculo de las cantidades necesarias para las contrataciones de las rutas del transporte escolar. Teniendo en cuenta dicha función, además de conseguir las mejoras sociales, se logra una reducción en el coste del 1,5%.

14. Se contrastan las soluciones obtenidas en este trabajo, con las obtenidas en Corberán A. y otros (2.000) (punto 4.3.5). Se realizan pruebas variando el número de autobuses utilizados para cada uno de los 16 servicios, aportando un total de 64 soluciones; el resultado de la comparativa es que conseguimos 31 soluciones mejores, 30 soluciones iguales y tan solo 3 de

nuestras soluciones son peores que las aportadas por Corberán A. y otros (2.000).

15. Es importante tener en cuenta que la utilización de herramientas que incorporan algoritmos como los propuestos en este trabajo, permiten realizar simulaciones de diferentes situaciones, lo cual puede facilitar en muchos casos la toma de decisiones. Algunas de estas situaciones son las siguientes:

- Reducción o aumento del número de alumnos.
- Reducción o aumento del número de poblaciones, con la consiguiente variación del número de alumnos.
- Reducción o aumento de centros.
- Aumentos o disminuciones en la flota de autobuses escolares.
- Eliminación o incorporación de carreteras o tramos de carreteras.
- Cambios en las restricciones impuestas por la ley: tiempo máximo de permanencia de los escolares en los autobuses, usos de determinado tipo de carreteras, velocidades máximas permitidas, número de alumnos por autobús, etc...

En base a los resultados expuestos se puede afirmar que es de gran interés económico y sobre todo social, la incorporación de este tipo de herramientas en la planificación del transporte escolar.

5.2 Nuevas Líneas de Investigación.

Los resultados obtenidos en este trabajo animan a apuntar nuevas líneas de investigación de cara al futuro. Algunos de los aspectos en los que se tiene previsto trabajar son los siguientes:

- Estudio y adaptación de nuevos entornos como el propuesto por Congram R.K. y Potts C.N. (1.998). Los autores proponen movimientos realizados considerando el conjunto de intercambios que mejor solución aporte en cada iteración, en lugar de movimientos realizados considerando el mejor intercambio.

- Uso del Conjunto de Concentración como método de Intensificación en Búsqueda Tabú aplicado a otros problemas, como puede ser el diseño de clusters, y comparación y/o conexión con otras técnicas de Intensificación como Path Relinking.

- Análisis del equilibrio entre Diversificación e Intensificación. Esa dualidad está presente en forma explícita en el diseño de estrategias como Scatter Search o Búsqueda Tabú, pero entendemos que también lo está en prácticamente todas las técnicas Metaheurísticas, aunque de forma implícita. Así en

Algoritmos Genéticos la selección de padres según su valor de fitness es una forma de Intensificación y el uso del operador de mutación es una forma de diversificación. En GRASP la generación de soluciones ávido-aleatoria tiene como propósito dar al proceso un equilibrio entre calidad y diversidad. En Concentración Heurística, se generan óptimos locales a partir de soluciones aleatorias (diversificación) y después se concentra la búsqueda (intensificación).

- Estudio de procedimientos de aceleración de los algoritmos. En este trabajo se ha adaptado la estrategia de Búsqueda Local Rápida a problemas de rutas y actualmente se está trabajando en adaptar esta idea a otros problemas. También sería interesante el caso de estructuras de memoria como árboles binarios que podrían ser adaptados a algoritmos que usan movimientos vecinales no necesariamente de mejora (como Búsqueda Tabú y Temple Simulado).

- En cuanto al tratamiento de problemas reales, por la trayectoria profesional de la autora, se pretende seguir trabajando en problemas relacionados con la logística, no necesariamente problemas de rutas puros. En este sentido se quiere retomar el estudio de un modelo que entendemos de interés en este campo: la Planificación y Programación de las entregas y recogidas de mercancía para la racionalización del transporte. En un trabajo

anterior, Pacheco y Delgado (1.998) ya se hizo una primera aproximación a este problema. Se analizó un caso concreto con una empresa de Repostería; permitiéndose adelantos de uno o dos días con respecto a la fecha inicial de entrega/recogida se conseguía disminuir el número de vehículos necesarios.

Se pretende fomentar y animar a la investigación con la elaboración de una página WEB en la que se incluirán los trabajos más interesantes que se realicen en el departamento, en cuanto a temas de optimización. De esta forma también se posibilitará que personas que estén en nuestra misma línea de investigación puedan informarse de nuestros avances, permitiendo una mayor colaboración con investigadores externos al departamento. Así mismo se incluirán *links* con otras páginas que puedan ser de interés.

CAPITULO VI

APÉNDICE

Apéndice 1. Variantes del VRP.

Apéndice 2. Descripción de las variables utilizadas para el VRPTW Mixto.

Apéndice 3. Algoritmo de Inserción de máxima diferencia.

Apéndice 4. Nuevas mejores soluciones.

Apéndice 5. Artículos del Diario de Burgos.

Apéndice 6. Comportamiento asintótico del Temple Simulado.

Apéndice 1.- Variantes del VRP

1.1 VRP, Problema de Rutas de Vehículos.

Se utilizan *m* vehículos de capacidad conocida para visitar a *n-1* clientes repartidos geográficamente; la distancia a recorrer ha de ser la mínima.

1.2 VRP con Partición de Demanda o Demanda Compartida.

Un cliente puede ser visitado por más de un vehículo, es decir, la demanda de los clientes puede ser satisfecha por más de un vehículo.

1.3 Problema de Rutas de Vehículos con Carga y Descarga Simultánea o VRP Mixto.

Los vehículos pueden realizar en una misma ruta entregas y recogidas de mercancía, siempre que la capacidad máxima del vehículo no sea superada en ningún momento.

1.4 El Problema de Carga y Descarga o PDP.

La última mercancía que ha entrado en el vehículo es la primera en salir. La mercancía objeto de transporte no se encuentra almacenada en un depósito, sino que está repartida geográficamente entre los clientes.

1.5 Modelos con Ventanas de Tiempo, VRPTW.

En este caso los puntos de visita llevan asociados un intervalo de tiempo, de forma que si se llega antes de la hora de apertura se produce una espera y nunca se podrá llegar más tarde pues en ese caso la entrega no se podrá efectuar.

1.6 Modelos con Flota Heterogénea.

Las capacidades de los vehículos varían según el tipo de vehículo.

1.7 Modelos con más de un origen.

La mercancía a entregar se almacena en más de un punto de partida, por lo que los vehículos no comienzan sus rutas desde un punto común.

Apéndice 2.- Descripción de las variables utilizadas para el VRPTW Mixto.

Las variables que intervienen en el Problema de Rutas de Vehículos con Ventanas de tiempo y con Carga y Descarga Simultánea o VRPTW Mixto (Mixed VRPTW) con flota heterogénea pueden ser denotadas de la forma siguiente:

- $\{2, 3, ..., n1\}$: conjunto de puntos donde hay que entregar unas determinadas cantidades de mercancía, desde un origen 1.

- *q(i)*, $i = 2, ..., n1$: cantidades de mercancía a entregar en los puntos de entrega; parten del origen.

- $\{n1+1, n1+2, ..., n1+n2\}$: conjunto de puntos donde hay que recoger determinadas cantidades de mercancía y llevarlas al origen 1.

- *q(i)*, $i = n1+1, ..., n1+n2$: cantidades de mercancía a recoger en cada punto de recogida para llevarlas al origen.

- $[e_i, l_i]$, $i = 2, ..., n1+n2$: intervalo de tiempo de visita asociado a cada punto. Si se llega a i antes del instante e_i se produce un tiempo de espera, y si se llega más tarde del instante l_i no se puede realizar el servicio, es decir, la solución no es factible.

- d_{ij} y t_{ij}: distancias y tiempos entre cada par de puntos $i, j \in \{1, 2, ..., n1+n2\}$, $(n = n1+n2)$.

- *ntipos:* número de tipos de vehículos disponibles.

- *capactipo(i)* y *costetipo(i)*, $i = 1, ...,$ *ntipos*: capacidades y costes de cada tipo de vehículo; obviamente a más capacidad mas coste.

Apéndice 3.- Algoritmo de Inserción de máxima diferencia.

Se trata de un sencillo algoritmo constructivo que en cada paso inserta un elemento en la solución actual. 2.3; 4.3.2

Sea $N = \{2, 3, ..., n\}$ el conjunto de puntos de visita (carga y descarga). Se define en cada paso:

- S = Conjunto de puntos sin insertar en la solución actual;
- R_j= Ruta j-ésima, $j = 2, 3,..., n$; (en principio tantas rutas como puntos);
- $d(j, i)$ = Incremento de distancia que resulta de insertar i en la ruta j (la mejor posición); si no existe posición factible donde colocar j entonces $d(j, i) = \infty$;
- $A_i = d(j^*, i) - d(j(i), i)$, donde:

$$d(j(i), i) = \min \{d(j, i) / j = 2,...,n\} \quad \text{y}$$

$$d(j^*, i) = \min \{d(j, i) / j = 2,...,n; j \neq j(i)\}.$$

Procedimiento Solución_Inicial

Inicialmente hacer $S = N$; $R_j = 1$—**1 (ruta formada solamente** *por el origen),* $j = 2,...,n$.

Repetir

Determinar $A_{i^*} = max \{A_i / i \in S\}$

Insertar i^* *en* $R_{j(i)}$

Ejecutar Busqueda_Local_Or(3, $R_{j(i)}$);

Hacer $S = S - \{i^*\}$

Hasta $S = \varnothing$

Apéndice 4.- Nuevas mejores soluciones obtenidas.

Problema eil 30: $n = 30$, Q = 4500

Solución: dist (Rounded) = 503, número de vehículos = 4

Ruta 1: 1 - 21 - 4 - 5 - 6 - 3 - 23 - 1 : (dist.Rnd.= 134 / qt=925)

Ruta 2: 1 - 27 - 29 - 28 - 30 - 26 - 25 - 2 - 7 - 20 - 1:
(184 / 3700)

Ruta 3: 1 - 16 - 17 - 14 - 8 - 18 - 10 - 15 - 9 - 13 - 12 - 11 - 24 - 19 - 1:
(139 / 3625)

Ruta 4: 1 - 22 - 1: (46 / 1500)

Problema eil 31: $n = 31$, Q = 140

Solución: distancia = 379, número de vehículos = 7

Ruta 1: 1 - 27 - 5 - 18 - 6 - 17 - 1 : (dist = 76 / qt=134)

Ruta 2: 1 - 11 - 19 - 13 - 8 - 28 - 1 : (57 / 137)

Ruta 3: 1 - 21 - 7 - 23 - 9 - 3 - 20 - 1 : (45 / 130)

Ruta 4: 1 - 25 - 4 - 15 - 2 - 26 - 1 : (82 / 128)

Ruta 5: 1 - 16 - 29 - 10 - 14 - 12 - 22 - 1 : (54 / 126)

Ruta 6: 1 - 31 - 1 : (30 / 123)

Ruta 7: 1 - 30 - 24 - 1 : (35 / 125)

Problema eilA76: $n = 76$, Q = 140

Solución: distancia (Rounded) = 831, número de vehículos = 10

Ruta 1: 1 - 13 - 73 - 32 - 26 - 56 - 19 - 51 - 45 - 4 - 52 - 1:
(dist.round.=118 / qt=138)

Ruta 2: 1 - 6 - 38 - 21 - 71 - 61 - 72 - 37 - 48 - 49 - 1 :
(93 / 135)

Ruta 3: 1 - 68 - 53 - 28 - 46 - 5 - 76 - 1 : (40 / 137)

Ruta 4: 1 - 31 - 75 - 22 - 70 - 62 - 29 - 3 - 1: (89 / 138)

Ruta 5: 1 - 12 - 67 - 60 - 15 - 36 - 1 : (90 / 139)

Ruta 6: 1 - 7 - 34 - 64 - 24 - 57 - 25 - 50 - 17 - 1 :
(92 / 140)

Ruta 7: 1 - 35 - 47 - 9 - 20 - 55 - 14 - 58 - 16 - 30 - 1:
(83 / 140)

Ruta 8: 1 - 74 - 2 - 44 - 42 - 43 - 65 - 23 - 63 - 69 - 1 :
(94 / 136)

Ruta 9: 1 - 40 - 10 - 33 - 41 - 18 - 1 : (57 / 126)

Ruta 10: 1 - 8 - 54 - 66 - 39 - 11 - 59 - 27 - 1: (75 / 135)

Problema eilB76: $n = 76$, Q = 100

Solución: distancia (Rounded)= 1023, número de vehículos = 14

Ruta 1: 1 - 74 - 44 - 42 - 43 - 65 - 23 - 69 - 1 :
(dist.round.=94 / qt=100)

Ruta 2: 1 - 3 - 29 - 63 - 34 - 1 : (59 / 100)

Ruta 3: 1 - 59 - 39 - 11 - 32 - 73 - 1: (86 / 97)

Ruta 4: 1 - 4 - 45 - 33 - 41 - 1: (51 / 89)

Ruta 5: 1 - 75 - 22 - 48 - 49 - 31 - 1: (61 / 99)

Ruta 6: 1 - 51 - 26 - 56 - 19 - 25 - 50 - 52 - 1: (109 / 100)

Ruta 7: 1 - 46 - 30 - 6 - 16 - 58 - 28 - 1: (65 / 94)

Ruta 8: 1 - 17 - 64 - 24 - 57 - 2 - 7 - 1: (81 / 99)

Ruta 9: 1 - 18 - 10 - 40 - 13 - 27 - 1: (53 / 99)

Ruta 10: 1 - 62 - 70 - 37 - 72 - 61 - 71 - 21 - 38 - 1: (115 / 98)

Ruta 11: 1 - 9 - 15 - 60 - 54 - 1 : (81 / 93)

Ruta 12: 1 - 8 - 12 - 67 - 66 - 1 : (76 / 98)

Ruta 13: 1 - 53 - 14 - 55 - 20 - 36 - 47 - 1 : (67 / 99)

Ruta 14: 1 - 68 - 35 - 5 - 76 - 1 : (25 / 99)

Problema eilD76: $n = 76$, $Q = 220$

Solución: distancia (Rounded) = 682, número de vehículos = 7

Ruta 1: 1 - 47 - 9 - 20 - 55 - 14 - 58 - 16 - 6 - 30 - 46 - 28 - 53 - 35 - 1 : (dist.round.=97 / qt= 218)

Ruta 2: 1 - 27 - 13 - 73 - 59 - 11 - 32 - 56 - 26 - 10 - 40 - 41- 1: (116 / 206)

Ruta 3: 1 - 68 - 5 - 76 - 1 : (20 / 80)

Ruta 4: 1 - 7 - 34 - 74 - 2 - 44 - 42 - 43 - 65 - 23 - 62 - 29 - 63 -1: (116 / 216)

Ruta 5: 1 - 36 - 15 - 60 - 67 - 66 - 39 - 12 - 54 - 8 - 1 : (105 / 209)

Ruta 6: 1 - 69 - 3 - 75 - 22 - 70 - 72 - 61 - 71 - 21 - 38 - 37 - 48 – 49 - 31 - 1 : (111 / 218)

Ruta 7: 1 - 18 - 4 - 45 - 33 - 51 - 19 - 25 - 50 - 57 - 24 - 64 - 17 - 52 - 1 : (117 / 217)

Problema eilA101: $n = 101$, $Q = 200$

Solución: distancia (Rounded) = 815, número de vehículos = 8

Ruta 1: 1 - 93 - 38 - 99 - 101 - 92 - 17 - 87 - 39 - 45 - 15 - 43 - 44 - 16 - 58 - 3 - 59 - 1: (dist.round.=125 / qt=198)

Ruta 2: 1 - 54 - 27 - 5 - 26 - 56 - 55 - 25 - 30 - 69 - 81 - 13 - 29 - 1 : (97 / 165)

Ruta 3: 1 - 95 - 96 - 98 - 88 - 14 - 1 : (40 / 108)

Ruta 4: 1 - 28 - 70 - 2 - 71 - 31 - 21 - 67 - 33 - 91 - 64 - 11 - 63 - 89 - 32 - 1 : (111 / 199)

Ruta 5: 1 - 22 - 73 - 76 - 57 - 40 - 68 - 24 - 42 - 23 - 75 - 74 - 41 - 1: (104 / 194)

Ruta 6: 1 - 53 - 8 - 83 - 49 - 20 - 12 - 65 - 50 - 37 - 48 - 47 - 9 - 84 - 19 - 1 : (138 / 199)

Ruta 7: 1 - 77 - 78 - 4 - 80 - 79 - 35 - 36 - 66 - 72 - 10 - 52 - 82 - 34 - 51 - 1 : (119 / 199)

Ruta 8: 1 - 90 - 61 - 6 - 85 - 46 - 18 - 62 - 86 - 94 - 60 - 100 - 97 - 7 - 1 : (81 / 196)

Problema eilB101: $n = 101$, $Q = 112$

Solución: distancia (Rounded) = 1075, número de vehículos = 14

Ruta 1 : 1 - 93 - 38 - 92 - 45 - 39 - 87 - 17 - 62 - 1 : (dist.round.= 91 / 112)

Ruta 2 : 1 - 7 - 97 - 60 - 96 - 95 - 14 - 1 : (40 / 112)

Ruta 3 : 1 - 41 - 74 - 75 - 42 - 23 - 76 - 57 - 73 - 22 - 1 :

(72 / 109)

Ruta 4 : 1 - 77 - 78 - 80 - 82 - 34 - 2 - 70 - 1 : (63 / 103)

Ruta 5 : 1 - 24 - 68 - 40 - 26 - 56 - 5 - 1 : (105 / 112)

Ruta 6 : 1 - 11 - 64 - 91 - 33 - 67 - 21 - 31 - 71 - 1 :

(99 / 112)

Ruta 7 : 1 - 83 - 49 - 20 - 8 - 89 - 32 - 1 : (75 / 110)

Ruta 8 : 1 - 100 - 94 - 86 - 101 - 99 - 98 - 1 : (53 / 111)

Ruta 9 : 1 - 90 - 61 - 6 - 85 - 18 - 46 - 84 - 19 - 53 - 1 :

(74 / 101)

Ruta 10 : 1 - 54 - 59 - 3 - 58 - 16 - 44 - 15 - 43 - 88 - 1 :

(83 / 112)

Ruta 11 : 1 - 51 - 52 - 10 - 72 - 66 - 36 - 35 - 79 - 4 - 1 :

(110 / 112)

Ruta 12 : 1 - 27 - 55 - 25 - 30 - 69 - 81 - 13 - 1 : (72 / 108)

Ruta 13 : 1 - 9 - 47 - 48 - 37 - 50 - 65 - 12 - 63 - 1 :

(120 / 112)

Ruta 14 : 1 - 28 - 29 - 1 : (180 / 32)

Problema: r201

Número de Rutas: 4; Distancia Total: 1253.26

n. 1 : 0 95 59 92 42 15 14 98 61 16 44 38 86
85 99 84 6 94 96 97 37 43 13 58 0

(Dist. 227.16)

n. 2 : 0 33 65 63 31 69 52 27 28 12 29 76 30
71 9 51 81 79 78 34 3 50 20 10 32 66 35

70 1 0 (Dist. 404.18)

n. 3 : 0 5 83 45 82 47 36 64 11 19 62 88 7 8
18 90 49 46 48 60 17 91 100 93 89 0
(Dist. 317.07)

n. 4 : 0 2 72 39 75 23 67 21 73 40 53 87 57
41 22 56 26 68 54 74 4 55 25 24 80 77
0 (Dist. 304.86)

Problema: r202

Número de Rutas: 3; Distancia Total: 1204.50

n. 1 : 0 92 42 14 91 45 47 36 63 64 11 19 62
88 30 71 78 79 81 9 51 90 49 46 48 82
7 10 20 66 35 32 70 31 52 0
(Dist. 458.76)

n. 2 : 0 27 28 26 39 23 15 38 44 16 61 99 18
8 84 86 96 6 94 97 43 74 72 21 2 13 89
60 83 17 5 93 85 100 37 98 59 95 58 0
(Dist. 378.54)

n. 3 : 0 33 65 34 29 3 50 1 69 76 67 73 40
53 87 57 41 22 75 56 4 55 54 68 24 25
80 77 12 0 (Dist. 367.20)

Problema: r204

Número de Rutas: 2; Distancia Total: 853.52

n. 1 : 0 52 7 88 31 70 30 32 90 63 11 62 10
69 1 76 12 67 39 53 79 81 9 51 20 66
65 71 35 34 78 29 24 55 4 21 73 72 74

22 75 56 25 54 80 68 77 3 33 50 27 0

(Dist. 452.75)

n. 2 : 0 89 6 94 96 59 92 98 37 42 57 2 41
23 15 43 14 44 38 86 16 61 99 97 87 95
83 8 84 17 45 46 47 36 49 64 19 48 82
18 60 5 93 85 91 100 13 58 40 26 28 0

(Dist. 400.77)

Problema: r207

Número de Rutas: 2; Distancia Total: 923.55

n. 1 : 0 42 92 59 60 45 46 36 64 11 62 88 69
27 76 3 79 78 9 51 30 1 28 53 40 2 87
57 41 22 73 21 72 74 75 56 4 25 55 54
80 68 77 12 26 58 13 97 98 91 100 37 95
94 0 (Dist. 430.10)

n. 2 : 0 50 33 81 65 34 29 24 39 67 23 15 43
14 44 38 86 16 84 5 99 6 89 7 82 8 48
47 49 19 10 63 90 32 66 71 35 20 70 31
52 18 83 17 61 85 93 96 0

(Dist. 493.45)

Problema: r208

Número de Rutas: 2; Distancia Total: 736.86

n. 1 : 0 27 69 31 88 7 82 48 19 11 62 10 70
30 51 9 81 33 79 3 76 28 53 94 99 5 84

17 45 8 46 47 36 49 64 63 90 32 20 66
65 35 34 78 29 24 80 68 77 50 1 0
(Dist. 390.95)

n. 2 : 0 52 18 83 60 59 92 98 85 61 16 86 38
44 14 42 43 15 41 22 67 39 56 75 72 21
73 2 87 57 74 4 25 55 54 12 26 40 58
13 97 37 100 91 93 96 6 89 0
(Dist. 345.92)

Problema: rc202

Número de Rutas: 4; Distancia Total: 1161.79

n. 1 : 0 65 82 12 14 47 16 15 11 73 88 98 53
78 79 7 6 8 46 42 55 68 0
(Dist. 207.03)

n. 2 : 0 45 5 3 1 42 39 36 44 61 81 90 99 57
86 87 9 10 97 59 74 13 17 60 100 70 0
(Dist. 338.33)

n. 3 : 0 91 92 95 85 63 33 28 26 27 29 31 30
62 67 71 72 41 38 40 43 35 37 54 96 93
94 80 0 (Dist. 248.58)

n. 4 : 0 69 64 19 23 21 48 18 76 51 84 49 22
20 83 66 56 50 34 32 89 24 25 77 75
58 52 0 (Dist. 367.85)

Problema: r209

Número de Rutas: 3; Distancia Total: 923.81

n. 1 : 0 52 83 5 99 95 92 98 100 44 14 38 86

16 61 85 59 94 87 42 57 15 43 41 22 74
56 26 4 25 55 54 24 80 68 77 1 0
(Dist. 302.53)

n. 2 : 0 27 69 31 88 82 47 19 11 63 90 30 51
71 9 81 33 50 3 79 78 34 35 65 66 20
32 70 10 48 60 89 0 (Dist. 289.12)

n. 3 : 0 28 29 76 12 39 67 23 75 72 21 73 2
40 53 6 18 7 62 64 49 36 46 8 45 17 84
93 37 91 97 96 13 58 0 (Dist. 332.16)

Apéndice 5.- Artículos del Diario de Burgos.

Los artículos seleccionados se publicaron todos ellos en el mes de Marzo de 2000 en el Diario de Burgos, y son los siguientes:

- Diario de Burgos; Martes, 14 de Marzo de 2000; página 12.
- Diario de Burgos; Viernes, 24 de Marzo de 2000; página 24.
- Diario de Burgos; Viernes, 24 de Marzo de 2000; página 18.
- Diario de Burgos; Domingo, 26 de Marzo de 2000; página 20.
- Diario de Burgos; Jueves, 30 de Marzo de 2000; página 23.
- Diario de Burgos; Viernes, 31 de Marzo de 2000; página 18.

12 Diario de Burgos PROVINCIA

ARLANZA Las Apas convocan paros de diez minutos entre los días 20 y 24 de marzo

Cerca de cien alumnos se verán afectados por las movilizaciones del transporte escolar

Las movilizaciones del transporte escolar convocadas por las Asociaciones de Padres para los próximos días 20, 21, 22, 23 y 24 de marzo afectarán en la comarca del Arlanza a cerca de un centenar de alumnos. Esta medida de presión, que ayer fue respaldada en Lerma por las Apas de la zona, consistirán en la organización de paros de diez minutos en diferentes puntos de la rutas. Con estas actuaciones se pretende concienciar a la Administración de la necesidad de reducir el tiempo que los escolares permanecen en los autobuses.

BURGOS/LUIS CABAÑES.- Alrededor de un centenar de escolares de la comarca del Arlanza sufrirán, entre los días 20 y 24 de marzo, retrasos de diez minutos en sus llegadas a los colegios, como consecuencia de las movilizaciones convocadas por las organizaciones de padres.

«Algunas rutas son demasiado largas, lo que obliga a los escolares a permanecer en el interior de los autobuses durante más de una hora», explicó el presidente de Fapa-Burgos, Fernando Vélez, que ayer mantuvo una reunión en Lerma con representantes de las asociaciones de padres del Instituto Santo Domingo de Guzmán (Lerma), Colegio Público Pons Sorolla (Lerma), Colegio Público Alejandro R. de Valcárcel (Covarrubias), Montearlanza (Villalmanzo) y Virgen de Escuderos (Santa María del Campo).

Después de analizar la problemática singular que presenta el transporte escolar en esta zona de la provincia «donde existen rutas con un reducido número de alumnos», todos los asistentes a esta reunión respaldaron la celebración de estas movilizaciones, que consistirán en parar las rutas durante diez minutos en las localidades en las que se suba un mayor número de estudiantes.

«No queremos -dijo Vélez- que los alumnos pierdan horas lectivas. Con esta medida, expresaremos nuestro deseo de solucionar la situación, sin que los escolares pierdan demasiada clase».

Un momento de la reunión de asociaciones de padres de la comarca celebrada ayer en Lerma.

Al mismo tiempo que reconocían que el parque móvil «no es especialmente viejo», los padres de la comarca del Arlanza analizaron la problemática generada alrededor de la construcción del polideportivo del instituto de Lerma.

Al parecer, se están generando una serie de problemas respecto a la ubicación definitiva de la instalación, por lo que algunos padres han llegado a plantear la posibilidad de construir un nuevo centro educativo.

Finalmente, hay que destacar que las Apas del Arlanza también señalaron que, en algunos colegios, los alumnos deben permanecer a la entrada de sus centros después de que hayan finalizado el recorrido de la ruta.

Diario de Burgos; Martes 14 de Marzo de 2000; página 12.

Los padres demandan mayor seguridad

Trespaderne, Frías y Quintana Martín-Galíndez vivieron protestas por el transporte escolar

❑ **TRESPADERNE/A.C.-** Padres de alumnos del Instituto de Enseñanza Secundaria de Medina se concentraron ayer por la mañana en Trespaderne, Frías y Quintana Martín-Galíndez para exigir mayor seguridad en el transporte escolar. Todos ellos secundan la demanda de la Confederación Española de Asociaciones de Padres de Alumnos (CEAPA), que plantea la necesidad de un acompañante para el conductor de los vehículos donde viajan alumnos de menos de 14 años.

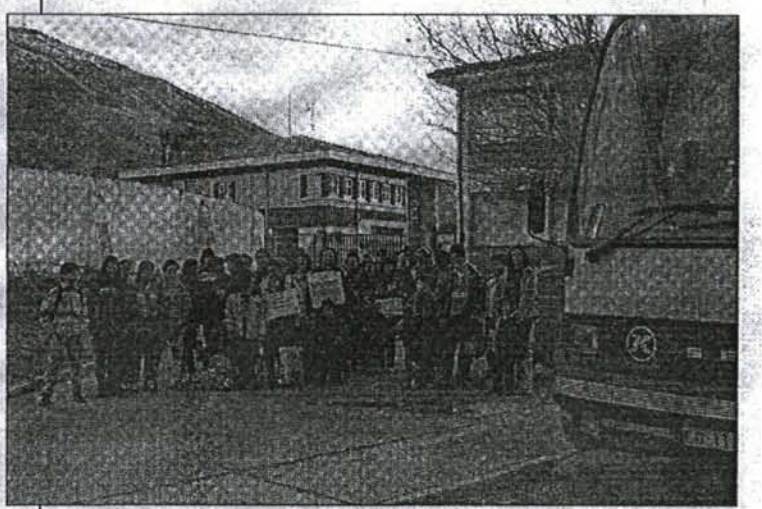
Los padres, que se concentraron a las horas de salida de los autobuses hacia Medina piden también la reducción del tiempo de llegada a los centros. En este sentido, los padres de los alumnos de Quintanaseca están muy descontentos, porque aseguran que el vehículo sale de esta localidad con más de una hora de antelación a la llegada al instituto –8,30 horas–, de modo que se incumple la norma que marca los recorridos en un máximo de una hora. Los 18 chicos y chicas de esta localidad junto con los de Frías, Valderrama y Montejo de Cebas deben, además, esperar unos diez minutos de transbordo en Quintana Martín-Galíndez, según explican desde la APA Río Trueba, lo que extiende el tiempo de viaje innecesariamente.

Por su parte, los 34 alumnos de Orbañanos, Pangusión, Barcina y otras localidades del Valle de Tobalina también pasan una hora de viaje para llegar a Medina, así como los 30 procedentes de la zona del Valle de Losa y los pertenecientes al área de Quintanilla Montecabezas, Pedrosa y Cadiñanos.

A.C.

Concentración celebrada en Trespaderne a las ocho de la mañana.

Diario de Burgos; Viernes, 24 de Marzo de 2000; página 24.



JESÚS ALCALDE DE HOYOS

Madres de alumnos durante la movilización celebrada ayer.

Medio centenar de personas solicitan en Belorado la mejora del transporte escolar

❑ **BURGOS**/JESÚS ALCALDE DE HOYOS.- Más de medio centenar de personas se concentraron ayer a la puerta del Colegio Público «Raimundo de Miguel», de Belorado, para reivindicar mejoras en la seguridad del transporte escolar. Al igual que se ha producido en otras zonas como Briviesca o Lerma, los padres de alumnos manifestaron su disconformidad con los actuales tiempos de viaje «que son excesivamente largos y, en algunos casos, superan los cuarenta y cinco minutos», además de solicitar la existencia de un acompañante que atienda a los escolares durante el trayecto de casa al colegio y del colegio a casa. La concentración de padres se produjo a las cinco de la tarde, coincidiendo con la finalización de la jornada escolar, lo que motivo el retraso de diez minutos en la salida de las cinco rutas del centro escolar. Finalmente, los manifestantes también solicitaron que los usuarios del transporte escolar puedan acceder a las actividades extraescolares que se organizan.

Diario de Burgos; Viernes, 24 de Marzo de 2000; página 18.

Un total de 4.367 estudiantes se desplazan cada día en autobús hasta su centro educativo

Los trayectos del transporte escolar alcanzan una media superior a los cuarenta minutos

La situación del transporte escolar en la provincia de Burgos ha provocado, durante las últimas semanas, el enfrentamiento entre las diferentes asociaciones de padres y la Dirección Provincial de Educación. Mejorar la realidad que diariamente viven más de cuatro mil escolares es el principal objetivo que buscan tanto la administración como los propios afectados. La orografía de la provincia burgalesa, así como su gran extensión provoca importantes dificultades que, hasta el momento, no parecen tener una solución definitiva.

❑ BURGOS/LUIS CABAÑES.- La seguridad y el bienestar de más de 4.300 escolares burgaleses se decidirá en las próximas semanas. Ellos son los que diariamente se ven obligados a coger el autobús para desplazarse hasta sus centros educativos. Un servicio que, a pesar de su evidente necesidad, está creando importantes quebraderos de cabeza a la Administración.

En estos momentos, la provincia de Burgos cuenta con una importante red de transporte escolar, distribuida a través de 206 rutas que recorren todas las comarcas, aunque con especial incidencia en la zona de Las Merindades y en el área de influencia de la capital burgalesa.

Las movilizaciones que se han registrado durante la pasada semana en localidades como Lerma, Briviesca, Trespaderne o Belorado han puesto de manifiesto algunas de las deficiencias que, según los padres, existen en este servicio que utilizan diariamente 2.042 alumnos de enseñanza primaria y 2.325 de enseñanza secundaria.

Desde la Dirección Provincial de Educación se han mostrado totalmente abiertos a la mejora de ese servicio, aunque no comparten algunas de las reivindicaciones realizadas desde las asociaciones de padres de alumnos.

ANA CASTELLANOS

Padres de alumnos y escolares durante las movilizaciones celebradas la semana pasada.

AUTOBUSES EN LA ESCUELA (1999-2000)

- Crédito: 485.068,00 ptas.
- Rutas: Primaria: 114 (10 a 5 tarde) Secundaria: 92
- Empresas: Total: 52
- Kilómetros diarios: Primaria: 6.540 Secundaria: 5.868 Total: 12.408 Media diaria: 60 Km. (ida y vuelta)
- Costes

De todas formas, existen algunos recorridos en diferentes comarcas burgalesas que se aproximan a una hora de trayecto, mientras que algunas no superan los diez minutos.

«Tenemos un gran problema con la gran dispersión de los núcleos de población. Por ejem-

REIVINDICACIONES

RECORTE DE LOS TIEMPOS DE VIAJE. Las asociaciones de padres de alumnos de la provincia han manifestado reiteradamente su disconformidad con los actuales recorridos de las diferentes rutas que, en muchos casos, obligan a los escolares a permanecer más de cincuenta minutos en el autobús. La intención de los padres es que estos tiempos sean reducidos al máximo. De hecho sus propuestas apuntan a que no existan rutas, en ningún punto de la provincia, que superen los treinta minutos de viaje, lo que supondría un verdadero alivio para muchos escolares.

ACOMPAÑANTES EN LAS RUTAS. La presencia de un acompañante en cada uno de los autobuses supone, en opinión de los padres, algo fundamental para el buen funcionamiento del transporte escolar. La presencia de estas personas evitaría que el conductor tenga que estar pendiente del comportamiento de los escolares que, en todos los casos, suelen estar todo el tiempo moviéndose durante el trayecto. Este acompañante sería el responsable de todos los movimientos que se produzcan en el interior del autobús, lo que evitaría toda la atención del conductor en la carretera.

RENOVACIÓN DEL PARQUE MÓVIL. La intención de los padres es acabar con aquellos autobuses que siguen efectuando el servicio diario y que, sin embargo, ya cuentan con

JUEVES, 30 DE MARZO DE 2000

CASTILLA Y LEON

Se contratará a acompañantes para los desplazamientos de los niños más pequeños

Educación estudia mejorar todas las rutas del transporte escolar para el 2000-2001

La Junta incrementará su presupuesto para el transporte escolar durante el próximo curso, con el objetivo de mejorar las rutas a través de medidas como la inclusión de este servicio en los niveles de enseñanza no obligatoria y la introducción de un acompañante para los niños de entre tres y cuatro años. Este fue uno de los asuntos que abordó ayer el pleno de las Cortes regionales, donde se trataron también cuestiones relacionadas con la atención a enfermos mentales o las subvenciones nominativas que concede directamente el presidente de la Junta.

MARIBEL BARRIENTE/ICAL
VALLADOLID

El consejero de Educación y Cultura, Tomás Villanueva, aseguró ayer en el pleno de las Cortes regionales, que su departamento ha encargo a las delegaciones territoriales de la Comunidad un estudio de las actuales rutas de transporte escolar, con el objetivo de mejorarlas de cara en el próximo curso académico, 2000-20001.

Villanueva respondió así a una pregunta formulada por el procurador socialista por Burgos, Octavio Granado, en la que se pedían explicaciones sobre este asunto, asegurando que su departamento tendrá que pensar en «realizar créditos extraordinarios» para financiar posibles mejoras en la red. Asimismo, señaló que durante el primer año de gestión de la educación no universitaria la Consejería había dedicado 3.400 millones de pesetas a este capítulo, frente a los 1.000 que invertía en él el Ministerio de Educación.

En concreto, el titular de Educación hizo referencia a que, una vez conocidos los informes de las respectivas delegaciones, la Junta se planteará la introducción de un acompañante para niños de entre 3 y 4 años de edad, así como la inclusión de de los niveles de enseñanza no obligatoria en este servicio, es decir, Educación Infantil y Formación Profesional.

Por su parte, Granado calificó la noticia como «buena», aunque recordó que su grupo ya había presentando enmiendas al presupuesto regional del 2000 en este aspecto, que fueron rechazadas por el PP.

El presidente de la Junta, Juan José Lucas, intervino ayer en la sesión plenaria, para responder a una pregunta formulada por el portavoz socialista, Jaime González. Este acusó al ejecutivo regional de «opacidad» y «falta de transparencia» en la concesión de las ayudas previstas en el Fondo de Acción Especial, una partida que permita a Lucas distribuir determinadas cantidades nominativamente y según su criterio.

RUBÉN CACHO/EFE

Lucas, conversando ayer en las Cortes con González Vallvé (I.) y Fernández Santiago.

Mancha. Otro de los aspectos abordados ayer fue la política de atención a enfermos mentales de la Junta, de manera que el consejero de Sanidad y Bienestar Social, Carlos Fernández Carriedo, anunció que la Comunidad contará con un Equipo de Salud Mental Infantil, en negociación con el Insalud y una Fundación Tutelar de Enfermos Mentales regional.

En el pleno, que comenzó ayer y continuará durante el día de hoy, tomaron posesión como procuradores por las provincias de Zamora y Segovia, Manuel Lozano (PSOE) y Juan José San Vitorio (PP), respectivamente. Lo hicieron en sustitución de Jesús Cuadrado y Jesús Merino, ya que ambos han resultado elegidos como diputados nacionales.

MEDIO AMBIENTE

Sin vertedero en Dueñas

El titular de la Consejería de Medio Ambiente, José Manuel Fernández Santiago, confirmó ayer la decisión de la empresa Hera Holding de retirar su proyecto de instalación de un Complejo Integral de Tratamiento Ambiental (CITA) en la localidad palentina de Dueñas.

Fernández Santiago precisó que esta decisión

CENTROS SOCIALES

Apoyo de la administración

La Junta apoyará la creación de cualquier centro de atención social que se realice a través del diálogo, facilitando la suficiente información al ciudadano y en la ubicación adecuada, según manifestó ayer el titular de Sanidad y Bienestar Social de la Junta, Carlos Fernández Carriedo, al responder a una pregunta del procurador burgalés, Octavio Granado

VINOS DE LA TIERRA

Denominación de Calidad

José Valín, consejero de Agricultura de la Junta, anunció ayer que, en breve, estará lista una Orden para desarrollar una denominación de calidad de los vinos de la tierra de Castilla y León, al que podrán acogerse todos aquellos caldos que no estén incluidos en Denominaciones de Origen.

Valín respondió así a una pregunta formulada por el

Diario de Burgos; Jueves, 30 de Marzo de 2000; página 23.

Giner, destacó, al inicio de la | los viajeros, las ciudades que to o Lliria.

Vélez califica como «positiva» la movilización de la semana pasada

La FAPA acusa al director provincial de Educación de «mediatizar y manipular» a las asociaciones

BURGOS.- La Federación de Asociaciones de Padres de Alumnos de Burgos (FAPA), a través de su presidente, Fernando Vélez, acusa al director provincial de Educación, Juan Carlos Rodríguez, de «mediatizar y manipular» a las diferentes APAS de la provincia.

En un escrito remitido a los medios de comunicación, el representante de los padres manifiesta que desde este organismo se ha tratado de atribuir a la FAPA informaciones erróneas sobre el transporte escolar, «cuando la realidad es que los datos que siempre se han manejado son los propios datos de los contratos que a la Federación ha proporcionado la Dirección Provincial».

En este mismo sentido y con motivo de las recientes movilizaciones protagonizadas por las asociaciones para reivindicar mejoras en el transporte escolar, Vélez considera que las únicas «informaciones erróneas» que se han vertido han sido las facilitadas por la Dirección Provincial.

Intimidación

«Los datos que se han mandado sobre las rutas de Lerma a las asociaciones de la comarca-insiste el presidente de FAPA-Burgos- difieren, casualmente, entre cinco y diez minutos en su duración, de los trayectos respecto a los contratos iniciales».

Asimismo, desde la representación de los padres de alumnos se señala que las cartas que llegaron a Lerma, lo hicieron con carácter urgente y en coche oficial, «en un afán claramente intimidador. Lástima que no sirviera de nada, Los tiempos del miedo y de la represión quedaron atrás».

Después de calificar como «muy importantes» las movilizaciones de la pasada semana, critica la lectura que el director provincial de Educación realiza de estas concentraciones, aludiendo a que han sido promovidas por intereses y posturas no esencialmente vinculadas al mundo de la educación.

«Quiere dejar entrever extraños y oscuros movimientos políticos que en ningún caso han existido. Cuando se convocaron las movilizaciones en Oviedo, una de las premisas que se tuvieron en cuenta precisamente fue el realizar estas movilizaciones después de las elecciones del día 12 de marzo, para evitar cualquier tipo de manipulación política», terminó diciendo el presidente de la Federación, Fernando Vélez.

no disponen de otros recursos. | radores del FAPE.

M.J.F.

Movilización de padres de alumnos en la zona de Briviesca.

Diario de Burgos; Viernes, 31 de Marzo de 2000; página 18.

Apéndice 6.- Comportamiento Asintótico de Temple Simulado.

Cada problema de Optimización combinatoria va a venir definido por el par (S, f), donde S representa el conjunto de soluciones factibles y f la función objetivo. Sin pérdida de generalidad se supondrá que los problemas a tratar son de minimización.

Teorema Principal.

Consideremos un problema de optimización combinatoria (S, f), con una "adecuada" estructura vecinal N. Después de un número suficiente de iteraciones en cada valor fijo de c, el proceso descrito en el algoritmo de temple simulado converge a cada solución $i \in S$ con una probabilidad:

$$P(X = i) = \frac{1}{N_0(c)} \exp\left(-\frac{f(i)}{c}\right) \quad (= q_i(c) \text{ def.}) \qquad (1.4.5)$$

donde X es la variable aleatoria que toma el valor de la actual solución, obtenida en este proceso tras realizar estas iteraciones; y

$$N_0(c) = \sum_{j \in S} \exp\left(-\frac{f(j)}{c}\right) \qquad (1.4.6)$$

denominada *constante de normalización.-*

A la anterior distribución de probabilidad (1.4.5) se la llama distribución *estacionaria* o *en equilibrio* y es equivalente a la

distribución de Boltzmann (1.4.1), y la *constante de normalización* (1.4.6) es equivalente a la *función de partición* (1.4.2).

Corolario.

Considérese un problema de optimización combinatoria (S, f), con una "adecuada" estructura vecinal N. Así mismo, sea la distribución estacionaria dada en (1.5.1.5), entonces:

$$\lim_{c\to 0} q_i(c) = \frac{1}{|S_{opt}|}\chi(S_{opt})(i) \quad [1] \quad (= q_i^* \text{ def.}) \qquad (1.4.7)$$

donde S_{opt} es el conjunto de soluciones donde se alcanza el óptimo global.-

Demostración: Obsérvese que $\forall\ a \leq 0$,

$$\lim_{x\to 0} e^{a/x} = \begin{cases} 1 & a = 0 \\ 0 & a < 0 \end{cases}$$

Entonces, se obtiene:

$$\lim_{c\to 0} q_i(c) = \lim_{c\to 0} \frac{\exp\left(-\frac{f(i)}{c}\right)}{\sum_{j\in S}\exp\left(-\frac{f(j)}{c}\right)} = \lim_{c\to 0} \frac{\exp\left(\frac{f_{opt}-f(i)}{c}\right)}{\sum_{j\in S}\exp\left(\frac{f_{opt}-f(j)}{c}\right)}$$

---

1 Sean dos conjuntos A y A', tales que A' $\subset$ A, se define la función $\chi_{A'}$: A $\rightarrow$ R, de la siguiente forma: $\chi_{A'}(a) = 1$ si $a \in A'$, y $\chi_{A'}(a) = 0$ si $a \notin A'$.

$$= \lim_{c \to 0} \frac{1}{\sum_{j \in S} \exp\left(\frac{f_{opt} - f(j)}{c}\right)} \chi(s_{opt})(i) + \lim_{c \to 0} \frac{\exp\left(\frac{f_{opt} - f(i)}{c}\right)}{\sum_{j \in S} \exp\left(\frac{f_{opt} - f(j)}{c}\right)} \chi(s - s_{opt})(i)$$

$$= \frac{1}{|S_{opt}|} \chi(s_{opt})(i) + 0$$

El resultado de este corolario es muy importante, ya que demuestra la convergencia asintótica del algoritmo de temple simulado al conjunto de óptimos globales, siempre que la *distribución estacionaria* (3.5) se alcance en cada valor de c.

El término "adecuada", que se utiliza en ambos enunciados, hace referencia a las condiciones que deben cumplir las estructuras vecinales N para poder demostrar la convergencia a la distribución en *equilibrio*; se requieren las siguientes condiciones:

- *Uniformidad*: Todas las soluciones tienen el mismo número de vecinas;
- *Simetría*: i es vecina de j si y sólo si j es vecina de i;
- *Alcanzabilidad*: Se ha de poder llegar desde una solución a cualquier otra tras un número finito de movimientos.

Sin embargo, según Dowsland (1.996), en recientes trabajos se ha conseguido demostrar esta convergencia solamente requiriendo la última condición.

Por otra parte, el número de transiciones necesarias en cada valor de c, para alcanzar dicha distribución estacionaria, es muy elevado. Concretamente Aarts y Van Laarhoven (1.985), demostraron que la distribución *estacionaria* sólo puede ser aproximada de forma arbitraria con un número de transiciones que sea al menos función cuadrática en el tamaño de S.

Propiedad 1 (Aarts y Van Laarhoven, (1.985)).

Sean :

$$a(k) = (a_i(k))_{i \in S} \qquad \text{y} \qquad q(c) = (q_i(c))_{i \in S}$$

donde $a_i(k)$ es la probabilidad de alcanzar la solución i después de k iteraciones para cada valor concreto de c. Considérese ε un número positivo arbitrariamente pequeño, entonces:

$$\|a(k) - q(c)\| < \varepsilon \qquad (1.4.8)$$

si:

$$k < K\left(1 + \frac{\ln\left(\frac{1}{2}\varepsilon\right)}{\ln\left(1 - \gamma^K(c)\right)}\right) \qquad (1.4.9)$$

donde $\gamma(c) = \min^{+}_{i,j \in S} P_{ij}(c)$; $P_{ij}(c)$ la probabilidad de pasar de la solución i a la j en una determinada transición para un determinado valor de c; y $K = |S|^2 - 3|S| + 3$.-

Por otra parte, el tamaño de S es para muchos problemas, al

menos exponencial en el tamaño del problema en si mismo; por ejemplo en el TSP, $|S| = (n-1)!$ siendo n el número de ciudades. Por tanto, según el análisis presentado anteriormente, sólo se podrá conseguir una aproximación arbitraria a la distribución estacionaria tras un tiempo de computación exponencial.

También se demuestra la convergencia asintótica al conjunto de soluciones óptimo, mediante al realización de un análisis del temple simulado que considere un número de transiciones fijo L para cada valor de c. Sin embargo, como en el caso del análisis que se acaba de presentar, el número de transiciones necesarias k es muy alto (Miltra y otros (1.986)); en concreto para el problema del viajante:

$$k = \theta\left(n^{n^{2n-1}}\right) \qquad (1.4.10)$$

Resulta que para el TSP, la completa enumeración del conjunto de soluciones utiliza menos tiempo de computación que la aproximación a una distribución, arbitrariamente cercana a la óptima, mediante temple simulado. Para otros problemas se llega a la misma conclusión.

CAPITULO VII

REFERENCIAS BIBLIOGRAFÍCAS

ALFA A. S., HERAGU S. S., CHEN M. (1.991).
A 3-opt based simulated annealing algorithm for the vehicle routing problem, Computers Industrial Engineering, vol. 21, pp. 635-639.

AARTS E.H.L. y KORST J.H.M. (1.990).
Simulated Annealing and Boltzmann Machines: A Stochastic Approach to Combinatorial Optimization and Neural Computing. Jhon Wiley & Sons. Chichester.

BACKER (DE) B., FURNON V., KILBY P., PROSSER P., HAW P. (1.997).
Solving Vehicle Routing Problems using Constraint Programming and Metaheuristics. Journal of Heuristics. Vol. 1- 16.

BADEAU P., GUERTIN F., GENDREAU M., POTVIN J.-Y., TAILLARD E. (1.997).
A Parallel Tabu Search Heuristic for the Vehicle Routing Problem with Time Windows. Transportation Research – Part C, vol. 5 no. 2, pp. 109-122.

BAKER J.E. (1.985).
Adaptive Selection Methods for Genetic Algorithms. Proceedings of the First International Conference on Genetic Algorithms.

BARBAROSOGLY G. y OZGUR D. (1.999).
A Tabu Search Algorithm for the Vehicle Routing Problems. Computers and Industrial Engineering, 21:635-639.

BARD J.F. y FEO T.A. (1.991).
An algorithm for the manufacturing equipment selection problem. IEE Transactions, Vol. 23, pp. 249-255.

BARKER J.A. y HENDERSON D. (1.976).
What is "liquid"? Understanding the states of matter. Reviews of Modern Physics 48, 587-671.

BATTITI R. (1.996).
Reactive Search: Toward Self-Tuning Heuristics. In V. J. Rayward-Smith, editor, Modern Heuristic Search Methods, chapter 4, pp. 61-83. John Wiley and Sons Ltd, 1.996.

BEAN J.C. (1.994).
Genetic Algorithms and Random Keys For Sequencing and Optimization. ORSA Journal on Computing 6(2): 154.

BEASLEY J. E. (1.983).
Route-First Cluster-Second Methods for Vehicle Routing. Omega, 11:403-408, 1.983

BENTLEY J.L. (1.992).
Fast Algorithms for Geometric Traveling Salesman Problems. ORSA Journal on Computing. Vol. 4, pp. 387-411.

BENYAHIA I. y POTVIN J.-Y. (1.995).
Decision Support for Vehicle Dispatching using Genetic Programming. Tech. Rep., Centre de Recherche sur les Transport.

BICKLE T. y THIELE L. (1.995).
A Comparison of Selection Schemes used in Genetic Algorithms. Computer Engineering and Communication Networks Lab, Swiss Federal Institute of Technology, TIK-Report-11.

BINDER, K. (1.978).
MonteCarlo Methods in Statistical Physics. Springer-Verlag. New York.

BLANTON J.L. y WAINWRIGHT R.L. (1.993).
Multiple Vehicle Routing with Time and Capacity Constraints using Genetic Algorithms. S. Forrest, editor, Proceedings of the Fifth International Conference on Genetic Algorithms, pp. 452-459. Morgan Kaufmann, San Mateo, CA.

BODIN L.D. y GOLDEN B.L. (1.981).
Classification in Vehicle Routing and Scheduling. Networks. Vol.11, n. 2, 97-108.

BOOKER L.B. (1.987).
Improving search in genetic algorithms. In Davis L. (Ed.) (1.987), pp. 61-73.

BULLNHEIMER B. (1.999).
Ant Colony Optimization in Vehicle Routing, tesis doctoral leída en la Universidad de Vienna en Enero de 1.999.

BULLHEIMER B., HARTI R. F., STRAUSS C. (1.997).
Applying the Ant System for the Vehicle Routing Problem. 2nd Metaheuristics International Conference (MIC-97), Sophie-Antipolis, France, July.

BULLNHEIMER B., HARTL R.F., STRAUSS C. (1.998).
Applying the Ant System to the Vehicle Routing Problem. En: Voss S., Martello S., Osman I.H., Roucairol C. (eds.), Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization, pp. 109-120 Kluwer: Boston.

BULLNHEIMER B., HARTL R.F., STRAUSS C. (1.999).
An Improved Ant system Algorithm for the Vehicle Routing Problem. Sixth Viennese workshop on Optimal Control, Dynamic Games, Nonlinear Dynamics and Adaptive Systems, Vienna (Austria), May 21-23, 1.997, aparecerá en: Annals of Operations Research (Dawid, Feichtinger and Hartl (eds.): Nonlinear Economic Dynamics and Control, 1.999.

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

CAMPOS V., GLOVER F., LAGUNA M., y MARTÍ R. (1.999). *An Experimental Evaluation of a Scatter Search for the Linear Ordering Problem.* Página Webb de Manuel Laguna: http://bus.colorado.edu/faculty/laguna/

CERNY V. (1.985). *Thermodynamical approach to the Traveling Salesman Problem: An efficient simulation algorithm.* Journal of Optimization Theory and Applications, 45, 41-51.

CHIANG W. C. y RUSSELL R. A. (1.993). *Hybrid Heuristics for the Vehicle Routing Problem with Time Windows.* Working Paper, Dpto. Quantitative Methods. University of Tulsa.

CHRISTOFIDES N., MINGOZZI A., TOTH P. (1.979). *The Vehicle Routing Problem.* N. Christofides, A. Mingozzi, P. Toth, and C. Sandi, editors, Combinatorial Optimization, 315-338. Wiley, Chichester.

CLARKE G. y WRIGHT J.W. (1.964). *Scheduling of Vehicles from a Central Depot to a Number of Delivery Points.* Oper. Res. Vol. 12, pp. 568-581.

COLLINS N.E., EGLESE R.W., GOLDEN B.L. (1.988). *Simulated Annealing: An annotated bibliography.* AJMMS, 8, 329-359.

CORBERÁN A., FERNÁNDEZ E., LAGUNA M. y MARTÍ R. (2.000). *Heuristic Solutions to the Problem of Routing School Buses with Multiple Objectives.* 8th International Conference. Computer-Aided Scheduling of Public Transport, CASPT-2000. 21-23 June, Berlin, Germany.

CORNE D., DORIGO M. y GLOVER F. (1.999). *New Ideas in Optimization.* McGraw Hill.

COTTA-PORRAS C. (1.998). *Un estudio de Técnicas híbridas y su Aplicación al diseño de Algoritmos Evolutivos.* Tesis leída en el departamento de I.O. Universidad de Málaga. 28 de Mayo.

CRAMER M.L. (1.985). *A Representation for the Adaptive Generation of Simple Sequential Programs.* Proceedings of the First International Conference on Genetic Algorithms. Grefenstette J.J. (ed.), Lawrence Erlbaum Associates, Hillsdale NJ.

CROES G. (1.958). *A Method for Solving Traveling Salesman Problems.* Operations Research, 6, pp. 791-812.

CROWSTON W.B., GLOVER F., THOMPSON G.L. y J.D. (1.963).
Probabilistic and Parametric Learning Combinations of Local Job Shop Scheduling Rules. ONR Research Memorandum No. 117, GSIA, Carnegie Mellon University. Pittsburgh, PA.

DARWIN (1.859).
On the Origin of Species by Means of Natural Selection. John Murray, Londres.

DAVIS L. (Editor) (1.987).
Genetic Algorithms and Simulated Annealing. Morgan Kauffmann, Los Altos, CA.

DAVIS L. (Editor) (1.991).
Handbook of Genetic Algorithms. Van Nostrand Reinhold, New York.

DELGADO C. R. (1.998).
Diseño de metaheurísticos híbridos para problemas de rutas con flota heterogénea (2 parte): GRASP y Concentración Heurística. ASEPUMA, Santiago de Compostela.

DELGADO C. R. (1.999).
Comparación de dos Tipos de Vecindarios en Problemas de Optimización de Rutas a través de Diferentes Librerías de Problemas. XIII Reunión Asepelt-España, Burgos.

DELGADO C. R. y PACHECO J. (2.000).
Problemas de Rutas Minmax: Aplicación al Transporte Escolar en la Provincia de Burgos. XXV Congreso Nacional de Estadística e Investigación Operativa, Vigo.

DESROCHERS M., DESROSIERS J. y SOLOMON M.M. (1.992).
A New Optimization Algorithm for the Vehicle Routing Problem with Time Windows. Operations Research 40, 342-354.

DESROCHERS M., LENSTRA J. K. SAVELSBERGH M. W. P. y 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., North-Holland, 65-84.

DIAZ A. (Coordinador), GLOVER F., GHAZIRI H.M., GONZÁLEZ J.L., LAGUNA M., MOSCATO P. y TSENG F.T. (1.996).
Optimización Heurística y Redes Neuronales en Dirección de Operaciones e Ingeniería. Ed. Paraninfo..

DIAZ J.A. y FERNANDEZ E. (1.998-a).
A Hibrid GRASP-Tabu Search Algorithm for the Single Source Capacitated Plant Location Problem. Document de Recerca, Departament D´EIO. Secció Informàtica. Universitat Politècnica de Catalunya. DR 98/04

DIAZ J.A. y FERNANDEZ E. (1.998-b).
A Tabu Search Heuristic for the Generalized Assigment Problem. Document de Recerca, Departament D´EIO. Secció Informàtica. Universitat Politècnica de Catalunya. DR 98/08

DORIGO M. y GAMBARDELLA L.M. (1.997-a)
Ant Colonies for the traveling salesman problems. TR/IRIDIA/1.996-3. Université Libre de Bruxelles. Belgium. Publicado en BioSystem, 43:73-81, 1997.

DORIGO M. y GAMBARDELLA L.M. (1.997-b)
Ant Colonies System: A Cooperative Learning Aproach to the Traveling Salesman Problems. TR/IRIDIA/1.996-5. Université Libre de Bruxelles. Belgium. Publicado en IEEE Transactions on Evolutionary Computation, 1 (1):53-66, 1.997.

DORIGO M. y GIANNI DI CARO (1.999)
The Ant Colony Optimization Meta-Heuristic. IRIDIA. Université Libre de Bruxelles. Belgium. Publicado en *New Ideas in Optimization,* D. Corne, M. Dorigo and F. Glover, McGraw-Hill, 1.999.

DORIGO M., MANIEZZO V. y COLORNI A. (1.996).
The Ant System: Optimization by a Colony of Cooperating. Agents. IEEE Transactions on Systems, Man, and Cybernetics-Part B, 26(1):29-41

DOWSLAND K.A. (1.993).
Some experiments with Simulated Annealing techniques for Packing problems. EJOR. Vol. 68, nº 3, pp. 389-399.

DOWSLAND K.A. (1.996).
Simulated Annealing in REEVES C.R. (ed.) Modern Heuristic Techniques for Combinatorial Problems. Blackwell Scientific Pub., Oxford.

DUHAMEL C., POTVIN F.-Y. y ROUSSEAU M. (1.997).
A Tabu Search Heuristic for the Vehicle Routing Problem with Backhauls and Time Windows. Transp. Science, vol. 31, pp. 49-59.

ESHELMAN L.J., CARUANA R.A. y SCHAFFER J.D. (1.989).
Biases in the crossover landscape. In Schaffer J.D. (Ed.) (1.987), pp.10-19.

FEO T. A. y RESENDE M. G. C. (1.989).
A Probabilistic heuristic for a computationally difficult Set Covering Problem. Operations Research Letters. Vol. 8, pp. 67-71.

FEO T. A. y RESENDE M. G. C. (1.995).
Greedy Randomized Adaptive Search Procedures. Journal of Global Optimization. Vol. 2, pp. 1-27.

FEO T.A. y GONZÁLEZ VELARDE J.L. (1.992).
The intermodal trailer assignment problem. Technical report, Operations Research Group, Department of Mechanical Engineering, The University of Texas at Austin, Austin, TX 78712-1063, April 1.992.

FISHER M. L. y JAIKUMAR R. (1.981).
A Generalized Assignment Heuristic for Vehicle Routing. Networks, vol.11, nº 2, 109-124.

FISHER H. y THOMPSON G. L. (1.963).
Probabilistic Learning Combinations of Local Job-Shop Scheduling Rules. Industrial Scheduling, J.F. Muth and G. L. Thompson (eds.) Prentice-Hall, pp. 225-251.

FOGEL L.J., OWENS A.J. y WALSH M.J. (1.966).
Artificial Intelligence Through Simulated Evolution. Wiley, New York.

GAMBARDELLA L., TAILLARD E. y AGAZZI G. (1.999).
New Ideas in Optimization, ch. MACS-VRPTW: A Multiple Ant Colony System for Vehicle Routing Problems with Time Windows, McGraw-Hill. También disponible como Technical report IDSIA-06-99, IDSIA, Lugano, Switzerland.

GARCÍA B.L., POTVIN J.Y. y ROUSSEAU J.M. (1.994).
A Parallel Implementation of the Tabu Search Heuristic for Vehicle Routing Problems with Time Window Constraints. Computers and Operations Research 21, 1025.

GENDREU M., HERTZ A. y LAPORTE G. (1.991).
A Tabu Search Heuristic for Vehicle Routing Problem. Report CRT-777. Centre de Recherche sur les Transports. Univ. Montréal.
GENDREU M., HERTZ A. y LAPORTE G. (1.992).
New Insertion and Postoptimization Procedures for the Traveling Salesman Problem. Operations Research, 40:1086-1094, 1.992.

GENDREAU M, GUERTIN F., POTVIN J.-Y. y TAILLARD E. (1.994-a).
Tabu Search for real-time Vehicle Routing and Dispatching. Tech. Rep. CRT 94-47, Tech. Rep. Centre de Recherche sur les Trasport, Universite de Montreal, December.

GENDREAU M., HERTZ A. y LAPORTE G. (1.994-b).
A Tabu Search Heuristic for Vehicle Routing Problem. Management Sci. 40 (10), pp.1276-1290.

GENDREAU M, LAPORTE G. y SEGUIN R. (1.994-c).
A Tabu Search Heuristic for the Vehicle Routing Problems with Stochastic Demands and Customers. Tech. Rep. Centre de Recherche sur les Trasport, Universite de Montreal.

GENDREAU M, LAPORTE G. y SEMET F. (1.997).
Solving an Ambulance Location Model by Tabu Search. Location Science, vol. 5, no. 2, pp 75-88.

GENDREAU M., LAPORTE G. y POTVIN J.Y. (1.999).
Metaheuristics for the Vehicle Routing Problem. Les Cahiers du GERARD, G-98-52. Sept., 1.998. Revisado: Ag., 1.999.

GLOVER F. (1.965).
A Multiphase Dual Agorithm for the Zero-One Integer Programming Problem. Operations Research, vol. 13, no. 6 pp. 879-919.

GLOVER F. (1.968).
Surrogate Constraints. Operations Research, vol. 16, pp. 741-749.

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

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

GLOVER F. (1.990-b).
Tabu Search: A Tutorial. Interfaces, Vol 20, No. 4, pp. 79-94.

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

GLOVER F. (1.998).
A Template for Scatter Search and Path Relinking. J.k. Hao, E. Lutton, E. Ronald, M. Schoenauer, D. Syners (eds.). artificial Evolution. Springer LNCS 1363, pp. 13-64

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 M. LAGUNA (1.997).
Tabu Search. Kluwer Academic Publishers, Boston.

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.

GLOVER F. KELLY J.P. y LAGUNA M. (1.995).
Genetic Algorithms and Tabu Search: Hybrids for Optimization. Computers Ops. Res. Vol. 22, No. 1, pp. 111-134.

GOLDBERG D.E. (1.989).
Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley, Reading, Massachusetts.

GOLDBERG D.E. (1.994).
Genetic and Evolutionary Algorithms Come of Age. Communications of the ACM. Vol.37, No.3. March.

GOLDBERG D.E. y LINGLE R. (1.985).
Alleles, Loci and the Trabeling Salesman Problem. In J.J.Grefenstette, editor, Proceedings of the Frist International Conference on Genetic Algorithms, pages 154-159. Lawrence Erlbaum, Hillsdale, NJ.

GOLDEN B., BODIN L., DOYLE T. y STEWART W. Jr. (1.980).
Approximate Traveling Salesman Algorithms. Operations Research, vol. 28, nº 3, parte II, 649-711.

HANSEN P. y MLADENOVIC N. (1.999).
First Improvement may be Better than Best Improvement: An Empirical Study. Les Cahiers du GERARD. G-99-40, October.

HAOUARI M., DEJAX P. y 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, N 3, pp 217-244.

HINTON G.E y SEJNOWSKI T.J. (1.986).
Learning and Relearning in Boltzmann machines. RUMELHART, D.E., McCLELLAND, J.L., and the PDP Research Group (Eds.), Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Bradford Books, 1, Cambridge (MA), pg. 283-317.

HIQUEBRAN D., ALFA A., SHAPIRO J., y GITTOES D. (1.994).
A revised simulated annealing and cluster-first route-second algorithm applied to the vehicle routing problem, Eng. Opt., vol. 22, pp. 77-107.

HJORRING C. (1.993).
Using Genetic Algorithms and the Petal Method to solve Vehicle Routing Problems. 29TH Annual Conference of the Operational Research Society of New Zealand, pp. 242-248.

HOLLAND J. (1.975).
Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Abor.

HOPPE J. (1.994).
Tabu Search and Vehicle Routing. Master's thesis, IMM. The Department of Mathematial Modeling. The Tecnical University of Denmark.

HUNTLEY C. L. y BROWN D. E. (1.991).
Parallel genetic algorithms with local search. Report, University of Virigina.

JANSSENS G. K., y BREEDAM A. V. (1.995).
A simulated annealing prostprocessor for the vehicle routing problem. Rayward-Smith, V.J. (Ed.), Applications of Modern Heuristic Techniques, Alfred Waller Ltd., UK, pp. 175-191.

KAWAMURA H., YAMAMOTO T., MITAMURA T., SUZUKI K. y OHUCHI A. (1.998).
Cooperative search on Pheromone Communication for vehicle routing problems. IEEE Transactions on Fundamentals, E81-A:1089-1096.

KILBY P., PROSSER P. y SHAW P. (1.997).
Guided Local Search for the Vehicle Routing Problem. 2nd Metaheuristics International Conference (MIC-97). Sophie-Antipolis, France, July, 21-24.

KIRKPATRICK S., GELATT JR. C. D. y VECCHI M. P. (1.983).
Optimization by Simulated Annealing. Science, Vol. 220, 1.983, pp. 671-680.

KLINCEWICZ J. G. y RAJAN A. (1.992).
Using GRASP to solve the component grouping problem. Technical report, AT&T Bell Laboratories, Holmdel, NJ.

KONTORAVDIS G. y BARD J. F. (1.995).
A GRASP for the Vehicle Routing Problem with Time Windows. ORSA Journal on Computing. Vol. 7, pp. 10-23.

KOZA J. R. (1.992).
Genetic Programming: On the programming of computers by means of natural selection. MIT Press, Cambridge MA, USA.

KUIK R. y SALOMON M. (1.990).
Multi-level lot-sizing problem: evaluation of a Simulated Annealing heuristic. EJOR 45, pp. 25-37.

LAGUNA M. (1.997).
Metaheuristic Optimization with Evolver, Geoncop and OptQuest. Graduate School of Business Administration. University of Colorado, Boulder, CO 80309-0419.

LAGUNA M. (1.999).
Scatter Search. Aparecerá en Handbook of Applied Optimization, P. M. Pardalos and M. G. C. Resende (eds). Oxford Academic Press.

LAGUNA M., FEO T. y ELROD H. (1.994).
A Greedy Randomized Adaptive Search Procedures for the 2-Partition Problem. Operations Research. Vol. 42, N 4, pp 677-687.

LAGUNA M., MARTI R. y CAMPOS V. (1.998).
Intensification and Diversification with Elite Tabu Search Solutions for the Linear Ordering Problem. Aceptada para su publicación en Computers & Ops. Res.

LANGTON C.G. (1.989).
Artificial Life. Artificial Life I (pp. 1-47), Langton C.G. (ed.), Addison-Wesley, Santa Fe NM.

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

LAPORTE G. y OSMAN I. H. (1.995).
Routing Problems: A Bibliography. Ann. Oper. Res. Vol. 61, pp 227-262.

LECLERC F. y POTVIN J.-Y. (1.997).
Genetic Algorithms for Vehicle Dispatching. Int. Trans. Opl. Res., vol. 4, no. 5/6 pp. 391-400.

LENSTRA J. K. y 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, pp 2245-2269.

LIN S. y KERNIGHAN B. W. (1.973).
An Effective Heuristic Algorithm for the Traveling Salesman Problem. Operations Research. Vol. 20, pp 498-516.

LUNDI M. y MEES A. (1.986).
Convergence of an Annealing algorithm. Math. Progr. 34, pp. 111-124.

MALUMBRES L. (1.994):
Condiciones de Convergencia en Algoritmos Genéticos Mediante Modelos Dinámicos. Tesis doctoral. Facultad de Informática. Universidad Politécnica de Madrid.

MARTELLO S. y TOTH P. (1.990).
Knapsack Problems. John Wiley & Sons. Chichester.

MENDEL G. (1.865).
Versuche über Pflanzen-Hybriden. Verhandlungen des Naturforschendes Vereines in Brünn 4.

METROPOLIS N., ROSENBLUTH A., ROSENBLUTH M., TELLER A. y TELLER E. (1.953).
Equation of state calculations by fast computing machines. Journal of Chemical Physics, 21, pp. 1087-1092.

MICHALEWICZ Z. (1.992).
Genetic Algorithms + Data Structures = Evolution Programs. Springer-verlag, 1992.

MICHALEWICZ Z. (1.993).
A Hierarchy of Evolution Programs: An Experimental Study. Evolutionary Computation 1(1):51-76.

MINSKI M. (1.994).
Negative Expertise. International Journal of Expert Systems 7 (1), pp.13-19.

MONTANA D.J. y DAVIS L. (1.989).
Training feedforward neural networks using genetic algorithms. In Proceedings of the 1989 International Conference on Artificial Intelligence, San Mateo, CA, USA, 1986. Morgan Kaufmann Publishers.

MOSCATO P. (1.989).
On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts: Towards Memetic Algorithms. Caltech Concurrent Computation Program, C3P Report 826.

NORMAN M.G. y MOSCATO P. (1.989).
A Competitive and Cooperative Approach to Complex Combinatorial Search, Report 790, Caltech Concurrent Computation Program, California Institute of Technology, Pasadena. California, USA (expanded version published in the Proceedings of the 20th Informatics and Operations Research Meeting, Buenos Aires (20th JAIIO), Aug. 1.991, pp. 3.15-3.29.

NOWICKI E. y SMUTNICKI C. (1.993).
A Fast Taboo Search Algorithm for the Job Shop Problem. Report 8/93, Institute of Engineering Cybernetics, Technical University of Wroclaw.

NURMI K. (1.991).
Traveling Salesman Problem Tools for Microcomputers. Computers & Ops.Res. Vol. 18, N. 8, 741-749.

OLIVER I.M., SMITH D.J. y HOLLAND J.R.C. (1.987).
A Study of Permutation Crossover Operators on the Traveling Salesman Problem. In J.J. Grefenstette, editor Proceedings of the Second International Conference on Genetic Algorithms, pages 224-230. Lawrence Erlbaum, Hillsdale, NJ.

OR I. (1.976).
Traveling Salesman Type Combinatorial Problems y their Relations to the Logistics of Blood Banking. Ph. Thesis. Dpt. of Industrial Engineering 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. Vol. 41, pp. 421-451.

OSMAN I.H. (1.995).
Heuristics for the Generalized Assignment Problem: Simulated annealing and Tabu Search approaches. OR Spektrum, 17, n. 4, pp. 211-225.

OSMAN I.H. y KELLY J.P. (1.996):
Meta-Heuristics: Theory & Applications. Kluwer Academic Publisher, Boston.

PACHECO J. (1.994).
Problemas de Rutas con Ventanas de Tiempo. Tesis doctoral leída en el Departamento de Estadística e Investigación Operativa de la Universidad Complutense de Madrid. Mayo.

PACHECO J. (1.997-a).
Metaheuristic based on a Simulated Annealing Process for one Vehicle Pick-Up and Delivery Problem in LIFO unloading Systems. Joint International Meeting EURO XV/ INFORMS XXXIV. Barcelona , Julio 1.997.

PACHECO J. (1.997-b).
El Temple Simulado: Descripción y Aplicaciones en Optimización Combinatoria. Lección de la Cátedra de Escuela de la Universidad de Burgos, Julio 1.997.

PACHECO J. y DELGADO C. (1.995).
El Problema del Viajante con Carga y Descarga y Ventanas de Tiempo: Algoritmos heurísticos. XXII Congreso Nacional de Estadística e Investigación Operativa. Sevilla, Noviembre 1.995.

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. (1.997-a).
Metaheurístico para el VRPTW con Carga y Descarga Simultánea. XXII Congreso Nacional de Estadística e Investigación Operativa. Valencia, Abril 1.997.

PACHECO J., ARAGON A. y DELGADO C. (1.999).
Diseño de un Sistema que Facilite Soluciones Racionales al Problema del Transporte Escolar en la Provincia de Burgos. XIII Reunión ASEPELT-España. Burgos, Junio.

PACHECO J., ARAGON A. y DELGADO C. (2.000).
Diseño de Algoritmos para el Problema del Transporte Escolar en la Provincia de Burgos. Qüesttió, vol. 24, nº 1, pp.55-82.

PAZOS A. (1.996).
Redes de Neuronas Artificiales y Algoritmos Genéticos, N. 19. Colección: Cursos, Congresos e Simpos. Universidade da Coruña.

POTVIN J. Y. (1.996).
Genetic Algorithms for the Traveling Salesman Problem. Annals of Operations Research, 63:339-370.

POTVIN J.Y. y BENGIO S. (1.994).
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. y BENGIO S. (1.996).
The vehicle routing problems with time windows-Part II:Genetic search. INFORMS Journal on Computing, 8:165-172.

PACHECO J. y DELGADO C. (1.997-b).
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, n. 7 , pp. 79-100.

PACHECO J. y DELGADO C. (1.998).
Diseño de Metaheurísticos Híbridos para Problemas de Rutas con Flota Heterogénea. XII Reunión ASEPELT-ESPAÑA. Córdoba. Junio 1.998.

PACHECO J. y DELGADO C. (1.999).
Diseño de Metaheurísticos Híbridos para Problemas de Rutas con Flota Heterogénea: GRASP. Cuadernos de Estudios Empresariales, Universidad Complutense de Madrid, nº 9, pp. 173-192.

PACHECO J. y DELGADO C. (2.000-a).
Diseño de Metaheurísticos Híbridos para Problemas de Rutas con Flota Heterogénea: Concentración Heurística. Estudios de Economía Aplicada nº 14, Asepelt-España, Abril 2000.

PACHECO J. y DELGADO C. (2.000-b).
Resultados de Diferentes Experiencias con Búsqueda Local Aplicadas a Problemas de Rutas. Rect@-Asepuma, Vol.2 - nº 1, Primer Semestre 2000.

POTVIN J.Y. y ROSSEAU J.M. (1.991).
A Parallel Route Building Algorithm for the Vehicle Routing and Scheduling Problem with Time Windows. European Journal of Operational Research, 66:331-340.

POTVIN J.Y., KERVAHUT T. y ROUSSEAU J.-M. (1.992).
A Tabu Search Heuristic for the Vehicle Routing Problem with Time Windows. Pubication #855, Centre de recherche sur les transports, Montreal.

POTVIN J.Y., KERVAHUT T., GARCIA B.L. y ROUSSEAU J.M. (1.993).
A Tabu Search Heuristic for the Vehicle Routing Problem with Time Windows. Technical Report, Centre de Recherche sur les Transports, Université de Montreal, Montreal, Canada. También disponible en Report CRT-777. Management Sci. Vol. 40 (10), pp. 1276-1290.

POTVIN J.Y., KERVAHUT T., GARCIA B.L. y ROUSSEAU J.M. (1.996).
The Vehicle Routing Problem with Time Windows-part 1: Tabu Search. ORSA J. Comp-, vol. 8, pp. 158-164.

PUREZA V.M. y FRANCA P.M. (1.991).
Vehicle Routing Problems via Tabu Search Metaheuristic. Technical Report CRT-347, Centre for research on transportation, Montreal, 1.991.

RECHENBERG I. (1.973).
Evolutionsstrategie: Optimirerung technischer Systeme nach Prinzipien der biologischen Evolution, Frommann-Holzboog, Stuttgart.

REEVES C.R. (1.995)
Modern Heuristic technique for combinatorial problems. Blackwell, 193.

REGO C. (1.998).
A Subpath Ejection Method for the Vehicle Routing Problem. Management Science. Vol. 44, N. 10, pp 1447-1459.

REGO C. y ROUCAIROL C. (1.995).
Using Tabu Search for Solving a Dynamic Multi-Termal Truck Dispatching Problem. EJOR, vol. 83, pp. 411-429.

REGO C. y ROUCAIROL C. (1.996).
A Parallel Tabu Search Algorithm using Ejection Chains for the Vehicle Routing Problem. I. H. Osman and J. P. Kelly, editors, Meth-Heuristics: Theory and Applications, pp. 661-675. Kluwer, Boston, 1.996.

REINELT G. (1.991).
TSPLIB: A Travelling Salesman Problem Library. ORSA Journal on Compouting, 3, 376-384.

REINELT G. (1.994).
The Traveling Salesman Problem: Computational Solutions for TSP Applications, Springer-Verlag, Berlin.

RESENDE M.G.C. y FEO T.A. (1.994).
A GRASP for Satisfiability. Technical report, AT&T Bell Laboratories, Murray Hill, NF.

ROBUSTÉ F., DAGANZO C.F., y SOULEYRETTE II R. (1.990).
Implementign Vehicle Routing Models. Transportation Research, 24B: 263-286.

ROCHAT Y. y SEMET F. (1.993).
A Tabu Search Approach for Delivery Petfood and Flour in Switzerland. Tech. Rep. Ecole Polytechnique Federale de Lausanne.

ROCHAT Y. y TAILLARD E. D. (1.995).
Probabilistic Diversification and Intensification in Local Search for Vehicle Routing. Journal of Heuristics. Vol. 1 (1), pp. 147-167.

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. y REVELLE C.S. (1.997).
Heuristic Concentration: Two Stage solution Construction. European Journal of Operational Research 97, pp.75-86.

ROSING K.E., REVELLE C.S., ROLLAND E., SCHILLING D.A. y CURRENT J.R. (1.998).
Heuristic Concentration and Tabu Search: A head to head comparison. European Journal of Operational Research 104, 93-99.

SAVELSBERGH M.W.P. (1.985).
Local Search for Routing Problems with Time Windows. Ann.Oper.Res., 4, 285-305.

SELMAN B., LEVESQUE H. y MITCHEL D. (1.992).
A new method for solving hard Satisfiability problems. Proceedings of the Tenth National Conference on Artificial Intelligence (AAAI-92), pp.440-446.

SEMET F. y TAILLARD E. (1.993).
Solving Real-Life Vehicle Routing Problems Efficiently Using Tabu Search. Ann. Oper. Res., vol. 41, pp 469-488.

SILVER E.A., VIDAL R.V. y WERRA (DE) D. (1.980).
A Tutorial on Heuristic Methods. European Journal of Operational Research, Vol. 5, 1.980.

SCHAFFER J.D. (Ed.) (1.987).
Proccedings of 3rd International Conference on Genetic Algorithms. Morgna Kaufmann, Los Altos, CA.

SCHMITT L.J. (1.994).
An Empirical Computational Study of Genetic Algorithms to Solve Order Based Problems; An Emphasis on TSP and VRPTC. P.D. Dissertation, Fogelman College of Business and Economics, University of Memphis.

SCHMITT L.J. (1.995).
An Evaluation of a Genetic Algorithmic Approach to the Vehicle Routing Problem. Working Paper, Department of Information Technology Management, Christian Brothers University, Memphis.

SMITH S. y FEO T.A. (1.991).
A GRASP for coloring sparse graphs. Technical report, Operations Research Group Department of Mechanical Engineering, The University of Texas at Austin, Austin, TX 78712-1063, January.

SOLOMON M.M. (1.987).
Algorithms for the Vehicle Routing and Scheduling Problem with Time Window Constraints. Operations Research, 35: 254-265.

SYSWERDA G. (1.989).
Uniform Crossover in Genetic Algorithms. Proceedings of the Third International Conference on Genetic Algorithms. Morgan Keufmann Publishers.

TAILLARD E.D. (1.993).
Parallel Iterative Search Methods for Vehicle Routing Problems. Networks, 23:661-673, 1.993.

TAILLARD E.D. (1.997).
A Tabu Search Heuristic for the Vehicle Routing Problem with Soft Time Windows. Transportation Science, vol. 31, p 170-186, May.

TAILLARD E.D., BADEAU P., GENDREU M., GUERTAIN F. y POTVIN J.Y. (1.995).
A new Neighborhood structure for the Vehicle Routing Problem with Time Windows. Technical Repport CRT-95-66. Centre de Recherche sur les Transports. Univ. Montréal.

TAILLARD E.D., BADEAU P., GENDREU M., GUERTAIN F. y POTVIN J.Y. (1.997).
A Tabu Search heuristic for the Vehicle Routing Problem with Time Windows. Transportation Science. Vol. 31, pp 170-186.

TEODOROVICH D. y PAVKOVIC G. (1.992).
A simulated annealing technique approach to the vehicle routing problem in the case of stochastic demand. Transp. Plan. & Tech., vol. 16, pp. 261-273.

THANGIAH S.R. (1.991).
Genetic Algorithms System for Vehicle Routing with Time Windows. North Dakota State University of Agriculture and Applied Sciences.

THANGIAH S.R. (1.993).
Vehicle Routing with Time Windows using Genetic Algorithms. Technical report SRU-CpSc-TR-93-23, Slippery Rock, PA.

THANGIAH S.R., NYGARD K.E. y JUELL P.L., GIDEON (1.991).
A Genetic Algorithms System for Vehicle Routing with Time Windows. Proceedings, Seventh IEEE Conference on Artificial Intelligence Applications, IEEE Computer SocietyPress. Los Alamitos, CA p 332.

THANGIAH S.R. y NYGARD K.E. (1.992).
School Bus Routing Using Genetic Algorithms. Applications of Artificial Intelligence X: Knowledge Based Systems, pp. 387-398. Proceedings of the SPIE. Orlando, FL, April.

THANGIAH S. R., OSMAN I. H. y SUN T. (1.994).
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., y SUN T. (1.993).
Vehicle Routing Problem with Time Deadlines using Genetic and Local Algorithms. In 5th International Conference on Genetic Algorithms.

THANGIAH S. R., OSMAN I. H., y SUN T. (1.994).
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.

TODA M., KUBO R. y SAITÔ,N. (1.983).
Statistical Physics. Springer-Verlag. Berlin.

TOTH P. y VIGO D. (1.998).
The Granular Tabu Search (and its Application to the Vehicle Routing Problem). Working Paper, DEIS, University of Bologna.

ULDER N.L.J. y OTROS (1.991).
Genetic local search algorithm for the traveling salesman problem. En R. Maenner y H.P. Schwefel, editores, "Parallel Problem Solving from Nature Lecture Notes in Computer Science", Vol 496, pp. 109—116. Springer Verlag.

VAN BREEDAM A. (1.995).
Improvement heurisitcs for the vehicle routing problem based on simulated annealign. European Journal of Operational Research, 86:480-490.

VAN BREEDAM A. (1.996).
An Analysis of the Effect of Local Improvement Operators in Genetic Algorithms and Simulated Annealing for the Vehicle Routing Problem. RUCA Working Paper 96/14, University of Antwerp, Belgium.

VOLGENANT A. y JONKER R. (1.983).
The Symetric Traveling Salesman Problem and Edge Exchange in Minimal 1-trees. European Journal of Operational Research, 12: 394-403.

VOSS S. (1.993).
Solving Quadratic Assignment Problems Using the Reverse elimination Method. Technische Hochschule Darmstadt, Germany.

VOUDOURIS C. y TSANG E. (1.995).
Guided Local Search. Technical Report CSM-247. Department of Computer Science. University of Essex, Colchester, C04 3SQ, UK. August 1.995.

VOUDOURIS C. y TSANG E. (1.999).
Guided Local Search for the Traveling Salesman Problem. European Journal of Operations Research. Vol. 113, pp 469-499.

WHITLEY L.D. (1.987).
Using reproductive evaluation to improve genetic search and heuristic discovery. Proceedings of the Second International Conference on Genetic Algorithms (pp. 116-121), Grefenstette J.J. (ed.) Lawrence Erlbaum Associates, Hillsdale NJ.

WHITLEY D., STARKWEATHER T. y FUQUAY D. (1.989).
Scheduling Problems and Traveling Salesmen: The Genetic Edge Recombination Operator. K.S. Schaffer, editor, Proceedings of the Third International Conference on Genetic Algorithms, pp. 133-140. Morgan Kaufmann, San Mateo, CA.

WILLIARD J.A.G. (1.989).
Vehicle Routing Using r-Optimal Tabu Search. M.Sc. Dissertation, The Managenemt School, Imperial College, London, 1.989.

WREN A. y WREN D.O. (1.995).
A Genetic Algorithm for public transport driver scheduling. Computers and Operations Research, 22(1):101.

XU J. y KELLY J.P. (1.996).
A network Flow-Based Tabu Search Heuristic for the Vehicle Routing Problem. Transportation Science, 30:379-393.

ZANAKIS S.H. y EVANS J.R. (1.981).
Heuristic "Optimization" : Why, When, and How to Use It. Interfaces Vol. 11, no. 5, October 1.981.

Páginas Web consultadas:
http://rtm.science.unitn.it/~battiti/reactive.html
http://densis.fee.unicamp.br/~moscato/memetic_home.html
http://bus.colorado.edu/faculty/laguna/
http://iridia.ulb.ac.be/~mdorigo/ACO/ACO.html
http://www.mailbase.ac.uk/lists-h-o/modern-heurisitcs/
http://cswww.essex.ac.uk/Research/CSP/gls.html.