Cuadernos de Estudios Empresariales
1999, número 9, 173-192

ISSN: 1131-6985

Diseño de metaheurísticos para problemas de rutas con flota heterogénea: GRASP

JOAQUÍN A. PACHECO
CRISTINA R. DELGADO SERNA
Departamento de Economía (Area de Economía Aplicada)
Facultad de C. EE y EE

RESUMEN

En este trabajo se propone un algoritmo Metaheurístico para el problema de rutas con ventanas de tiempo, carga y descarga simultánea y flota heterogénea, basado en un proceso de tipo GRASP. Este trabajo es la continuación de uno anterior reciente —Pacheco y Delgado (1998)—, en el que se proponían para este modelo Metaheurísticos basados en procesos de Temple Simulado y Búsqueda Tabú, así como un híbrido de estos dos. Se simulan una serie de problemas y se comparan las soluciones con las obtenidas por cada una de las estrategias anteriormente mencionadas, así como las obtenidas por otros algoritmos adaptados a este modelo. Posteriormente este trabajo se ampliará con otro en el que se propone un algoritmo de tipo *Concentración Heurística* —un tipo de Metaheurístico dado a conocer muy recientemente por Rosing (1997) y Rosing y ReVelle (1997)—, así como una estrategia híbrida que combina las ideas de GRASP y *Concentración Heurística*.

Palabras clave: Metaheurísticos, GRASP, *Concentración Heurística*, Problemas de Ruta con Flotas Heterogéneas.

1. INTRODUCCIÓN

El problema de Rutas de Vehículos con Ventanas de tiempo y con Carga y Descalga Simultánea o sencillamente VRPTW Mixto (Mixed VRPTW) con flota heterogénea puede ser descrito de la forma siguiente: considérese un conjunto de puntos {2, 3,..., n1} donde hay que entregar unas determinadas cantidades de mercancía q(i), i=2,...,n1, en forma de *palés*, desde un origen 1; además considérese otro conjunto de puntos {n1+1,n1+2,...,n1+n2} donde hay que recoger otras cantidades de *palés* q(i), i = n1+1,..., n1+n2, y llevarlas al origen 1. Para cumplir estos requeri-

mientos se dispone de una flota heterogénea con diferentes tipos de vehículos; cada tipo de vehículo tiene unas capacidades de carga diferente; cada punto del problema lleva asociado un intervalo de tiempo de visita $[e_i, l_i]$, i=2,...,n1+n2 (si se llega a i antes del instante e_i hay espera, y no puede visitar más tarde del instante l_i). Las distancias d_{ij} y tiempos t_{ij} entre cada par de puntos i, j $\in$ {1,2,...,n1+n2} son conocidas. (A partir de ahora n = n1+n2). El número de tipos de vehículos disponibles se denota por *ntipos* y las capacidades y costes de cada tipo por *capactipo(i)* y *costetipo(i)*, i = 1,..., *ntipos;* obviamente, a más capacidad más coste.

En este problema se ha de diseñar un conjunto de rutas de coste mínimo verificando las siguientes restricciones:

- Cada ruta comience y finalice en el punto 1;
- Se lleve la mercancía correspondiente a cada uno de los puntos del conjunto {2,..,n1}, y se recoja de cada uno de los puntos del conjunto {n1+1,n1+2,...,n1+n2};
- El número de *palés* que en cada momento deba llevar cada vehículo no supere su capacidad;
- Cada punto i, i = 2,..,n1+n2, sea visitado exactamente una vez;
- Se respeten los intervalos de tiempo de visita.

El coste total 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).

Existen muchos algoritmos de solución para el VRP *(Problemas de Rutas de Vehículos)* y el VRPTW (VRP con *Ventanas de Tiempo)* en la literatura. Se pueden encontrar recopilaciones de los principales en trabajos como los de Bodin y Golden (1981), Desrochers y otros (1988), Haouri y otros (1990) y Laporte (1992).

En cuanto al VRP (o VRPTW) Mixto con flota homogénea existen referencias en los trabajos de Pacheco (1994, 1997) y Pacheco y Delgado (1995, 1996, 1997a, 1997b). En un trabajo reciente de Pacheco y Delgado (1998), se abordó un modelo más general al contemplar flotas heterogéneas (como en el trabajo actual); concretamente se proponían Metaheurísticos basados en procesos de Temple Simulado y Búsqueda Tabú.

En la sección siguiente se describen brevemente los algoritmos tipo GRASP; en la sección 3 se describe el diseño de la fase constructiva; en la 4 se describe la fase de mejora; en la 5 se proponen una serie de variables globales que ayudan a reducir el tiempo de computación en ambas fases; en el apartado 6 se realizan pruebas para determinar el grado de aleatoriedad más adecuado en la fase constructiva, y; finalmente; en el apartado 7 se muestran los resultados computacionales.

2. GRASP

GRASP son las iniciales en inglés de Procedimientos de Búsqueda Ávidos Aleatorios y Adaptativos (Greedy Randomize Adaptive Search Procedures), y aunque inicialmente se dieron a conocer en el trabajo de Feo y Resende (1989) han tenido un desarrollo más reciente que los otros metaheurísticos. Una amplia descripción se puede encontrar en el trabajo de los mismos autores (1995). Una aplicación al VRPTW lo tenemos en el trabajo de Kantoravdis (1995). Básicamente actúan de la forma siguiente

Algoritmo GRASP

Repetir
Construir una solución Avida Aleatoria (Fase de Construcción)
Aplicar Búsqueda local a la solución obtenida (Fase de mejora)
hasta alcanzar una condición de parada

En la Fase de Construcción se va añadiendo en cada paso un elemento hasta obtener la solución completa. La selección de cada elemento que se añade se realiza de forma aleatoria entre los mejores candidatos *(Lista Restringida de Candidatos)* según una función ávida o voraz que valore en cada paso su inclusión en la solución teniendo en cuenta la función objetivo. Esto supone un equilibrio entre calidad y variabilidad de las soluciones obtenidas en la Fase de Construcción. 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.

En nuestro caso la condición de parada se produce cuando transcurren 100 iteraciones sin mejora en las soluciones encontradas. Por otra parte, la *Lista Restringida de Candidatos* estará formada por los elementos que se alejen menos de una determinada cantidad del mejor candidato según la función voraz.

A continuación se describe con detalle cada una de ambas fases.

3. FASE CONSTRUCTIVA

Habitualmente, cuando se hace referencia a estrategias que den rápidamente soluciones aceptables a problemas de rutas se piensa en algoritmos de inserción, como los que se aescriben en Golden y otros (1980) para el TSP, o de ahorros, como el popular método *Saving* propuesto por Clark & Wright (1964). Sin embargo, se cree conveniente para esta fase constructi-

va buscar funciones ávidas basados en criterios que hayan demostrado tener una mayor bondad para este tipo de problemas.

Fisher y Jaikumar (1981) diseñan un algoritmo para el VRP considerado por varios autores —Haouri (1990), Laporte (1992)...— 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 n.° 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, 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

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

sujeto a:

$$\sum^{n}_{i=2} x_{ij} \cdot q(i) \leq Q \quad j = 1,...,m, \quad \text{(las cargas de los puntos asignados a cada ruta no superan la capacidad de los vehículos)} \qquad (2)$$

$$\sum^{m}_{j=1} x_{ij} = 1 \quad i = 2,...,n, \quad \text{(cada envío se debe realizar por una ruta)} \qquad (3)$$

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

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

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

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

$c_{ij} = d_{1,i} + d_{1,2j} - d_{1,2j}$, ($d_{1,i}$ distancia del origen al punto de visita i; $d_{1,2j}$ distancia de i al punto-semilla e_j; $d_{i,ej}$ 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 éste, en una segunda fase se resuelve el TSP para cada conjunto de puntos encontrado y el origen.

Obviamente, ésta 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 y Toth (1990), que por lo general da buenos resultados. Básicamente actúa de la forma siguiente:

Algoritmo de Martello y Toth, MTHG

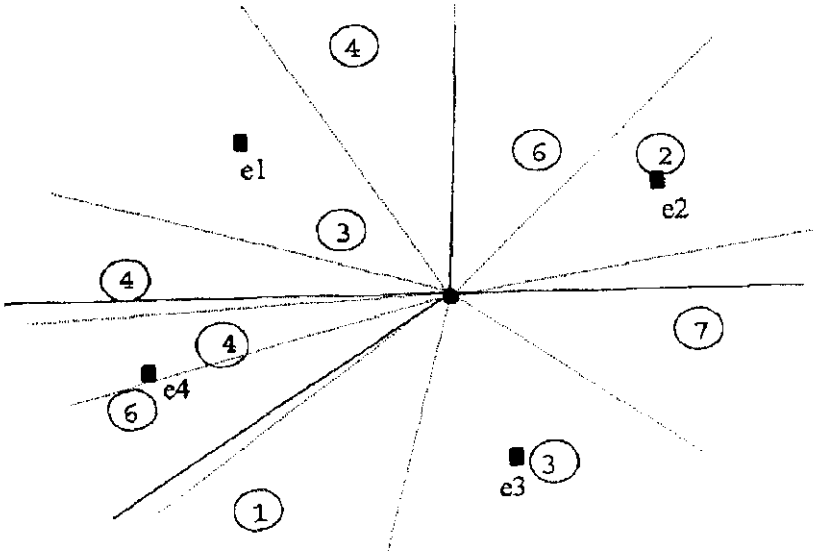
Hacer $Q(j) = Q$, $j = 1,...,m$; $U = \{2,.., n\}$ y $x_{ij} = 0$
Mientras $U \neq \varnothing$ *hacer:*
  $\forall i \in U$: *determinar* $c_{ij'(i)} = min\ \{c_{ij}/q(i) \leq Q(j),\ j = 1,...,m\}$
  determinar $c_{ij''} = min\ \{c_{ij}/q(i) \leq Q(j),\ j = 1,...,m;\ j \neq j'(i)\}$
  hacer $bj = c_{ij''} - c_{ij'(i)}$
determinar $b_{i'} = min\ \{b_i / i \in U\}$
Hacer $c_{i'j'(i)}$ *1*, $U = U - \{i'\}$, $Q(j'(i')) = Q(j'(i')) - q(i')$

Una cuestión importante es dónde situar los puntos-semilla. En algunos problemas donde haya 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 cuáles pueden ser esos puntos significativos. En cualquier caso, Fisher y Jaikumar incorporan en su trabajo un método para su cálculo de forma general[1]:

— Determinar las semirrectas que unen el origen con cada punto i $\in \{2,..., n\}$.
— Hallar las bisectrices de los ángulos que forman cada par de semirrectas 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.
— Comenzando por un punto cualquiera y 'barriendo' en el mismo sentido ir formando ángulos mayores *beta*(j), con los ángulos *alfa*(i) o con partes proporcionales de éstos, de forma que el peso total de estos nuevos ángulos sea $K = (\Sigma_{i=2.N}\ q(i))/m$. Obviamente habrá m ángulos *beta*(j).

— 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 0,75 K. Esta proporción es A/(A+B), donde A y B son las distancias a los puntos más cercanos al arco por dentro y por fuera, respectivamente.

FIGURA 1
Ejemplo de obtención de los puntos semilla (cuadros en negro), con m=4 y las cantidades de cada punto en círculo



En nuestro caso, se van a adaptar las ideas anteriores al modelo propuesto en este trabajo pero con las siguientes consideraciones:

---

[1] Este método de determinación de puntos-semilla exige conocer la situación 'geográfica' de los puntos del problema. Si ésta no está disponible se sugiere utilizar métodos factoriales, como los propuestos en Cuadras (1990), para obtener una representación en un plano de los puntos del problema a partir de la matriz de distancias (de forma que la distancia euclídea en ese plano de los puntos del problema sea una buena aproximación de la distancia original).

- 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 qué 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 el 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 la sección 5 proponemos un método, basado en el uso de variables globales, para realizar estos pasos en un número de operaciones constante 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 qué ser el óptimo para ese conjunto de puntos (aunque se espera sea aceptable). Se pueden mejorar aplicando los métodos de intercambio que se explican en la sección 4 a cada una de las rutas individualmente.

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 (Resolver Problema de Asignación)

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 es posible asignarlo 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:

Algoritmo GRASP - Fase Constructiva

Hacer m = m1
Repetir
Elegir aleatoriamente un elemento i1 $\{2,...,n\}$
Comenzando por el ángulo alfa(i1) Construir los ángulos beta(j)
Determinar puntos-semilla
Calcular coeficientes de asignación c_{ij}
Resolver Problema de Asignación
Si (no factible) entonces m = m + 1
hasta factible
Mejorar cada ruta formada por Búsqueda local (ver siguiente sección)

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*).

4. FASE DE MEJORA

La fase de mejora va a ser la aplicación de un método de búsqueda local o vecinal con estrategia de mayor descenso. Al igual que en trabajos anteriores se van a definir como soluciones vecinas las obtenidas por el método de intercambio propuesto por Or (1976), que sean factibles. El método de intercambio Or es una variante de los conocidos intercambios r-óptimos desarrollados por Lin (1965) y Lin & Kernighan (1973), para el Problema del Viajante (TSP) simétrico. Como se verá más adelante, el método de Or se puede utilizar en problemas asimétricos. La eficacia de este método para el TSP ha sido contrastada en trabajos como el de Nurmi (1991).

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

ruta 1: 1 - 3 - 5 - 1; y *ruta 2:* 1 - 4 - 2 - 1

puede expresarse como:

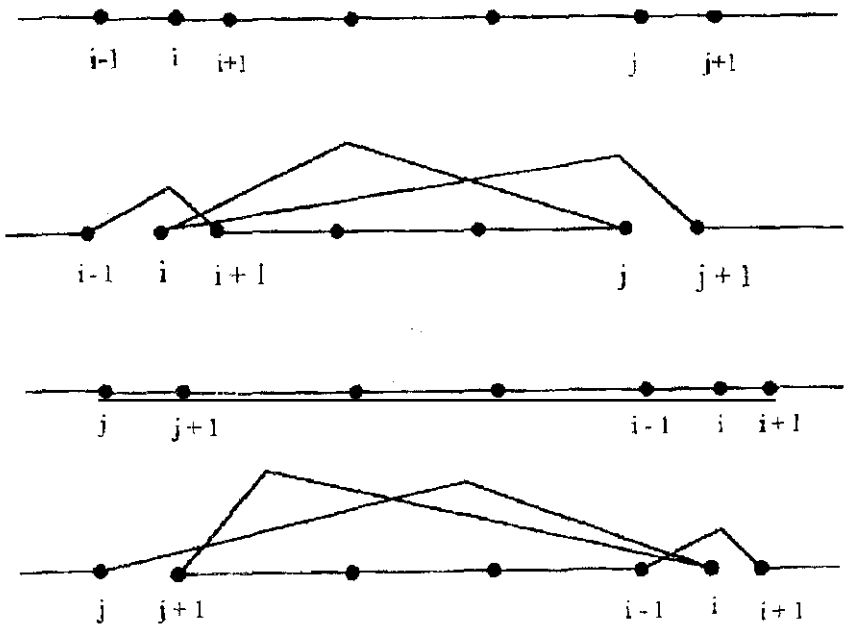
1 - 3 - 5 - 1 - 4 - 2 - 1

los '1' representan la vuelta de un vehículo al origen y la salida del siguiente (obviamente, el último y el primer elemento siempre serán '1'). De esta forma, como se ilustra a continuación, se puede aplicar los Or-inter-

cambios para obtener soluciones vecinas de la actual, considerando a ésta como una única secuencia.

Or (1976) 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.

FIGURA 2
Posible recolocación del elemento i y entre j y j + 1 hacia adelante y atrás



En este caso se sigue la misma idea, pero consideraremos sólo recolacaciones hacia adelante; además, se debe chequear la factibilidad de cada posible recolocación respecto a las restricciones del problema.

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

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

considérese un problema con cuatro puntos de descarga A, B, C y D con las siguientes cantidades: 6, 8, 4 y 8 palles. Considérense la solución

1 - A - B - C - 1 - D - 1

y su vecina:

1 - A - B - 1 - D - C - 1

Es claro que varían los vehículos requeridos por las rutas de cada solución (capacidades mínimas de 18 y 8 en la primera solución, y 14 y 12 en la segunda). Incluso recolocaciones dentro de la misma ruta pueden hacer cambiar el tipo de vehículo requerido, como se explicó en el apartado anterior.

Se denota N(s) el conjunto de soluciones vecinas de s, es decir, el conjunto de soluciones factibles obtenidas por intercambios hacia adelante de cadenas de a lo sumo 3 elementos. La fase de mejora queda de la siguiente forma:

Fase de Mejora

Leer s_0 solución obtenida en la fase constructiva
Repetir
    Determinar $N(s_0)$
    Encontrar $f(s') = min \{f(s) / s \in N(s)\}$
    Si $f(s') < f(s_0)$ entonces $s_0 = s'$
hasta $f(s')$ $f(s_0)$

Más concretemente se define $N_k(s)$ como el conjunto de soluciones factibles obtenidas por intercambios hacia adelante de cadenas de a lo sumo k elementos realizados en s. En este caso $N(s) = N_3(s)$, y por otra parte, sea nt el número total de puntos que componen una solución (incluyendo los 1 intermedios) se comprueba fácilmente $N_{nt-3}(s)$ coincide con el conjunto de todos los intercambios factibles de Or que se pueden realizar en s.

El chequeo de la factibilidad y la valoración de cada intercambio puede suponer un tiempo de computación excesivo. En el apartado siguiente se propone un método basado en la definición de variables globales para reducir este tiempo de computación.

5. DEFINICION DE VARIABLES GLOBALES

Según se ha comentado en las dos secciones anteriores el chequeo de la factibilidad y la valoración de las posiciones donde un elemento puede ser insertado (en la fase constructiva) o de los posibles intercambios (en la fase de mejora) puede suponer un tiempo de computación excesivo. El número de inserciones que se han de considerar en cada paso es del orden $\theta(n^2)$: a lo sumo n elementos a insertar y a lo sumo 2n posiciones donde se pueden colocar (añadimos los puntos de partida de cada ruta). De igual forma se han de considerar del orden $\theta(n^2)$ de intercambios: a lo sumo 2n elementos iniciales de cada cadena (como se explicó antes se anaden '1' que representen el final de una ruta y el comienzo de la siguiente), 3 es el tamaño máximo de cada cadena (por tanto, se consideran a lo sumo 6n cadenas a recolocar), y a lo sumo 2n elementos detrás del cual deben ser recolocados.

Una forma de chequear cada inserción es la siguiente: guardar la ruta parcial resultante en una variable auxiliar y examinar dicha ruta elemento a elemento; esto supone un número de operaciones de orden $\theta(n)$ para cada inserción, y por tanto de orden de $\theta(n^3)$ operaciones para cada paso. El mismo análisis se puede realizar para los intercambios.

Una alternativa es el uso de variables globales, como las que se proponen a continuación, con las que el número de operaciones para chequear y evaluar cada inserción e intercambio sea constante, es decir, independiente del tamaño del problema. A continuación se definen dichas variables y se presentan de forma breve alguno de los resultados más importantes (un análisis más amplio se puede encontrar en Pacheco y Delgado (1996 y 1997b).

5.1. Factibilidad respecto a la carga en el vehículo

Sea s un elemento en la solución actual, se define:

carga(s): Carga que lleva el vehículo tras visitar s en la solución actual;

y a partir de estos valores se calculan fácilmente y de forma recursiva los siguientes:

maxcarga_atrás(s): Máxima carga que lleva el vehículo desde el origen hasta la visita de s (incluida) en la solución actual;

maxcarga_delan(s): Máxima carga que lleva el vehículo después de la visita de s hasta el origen.

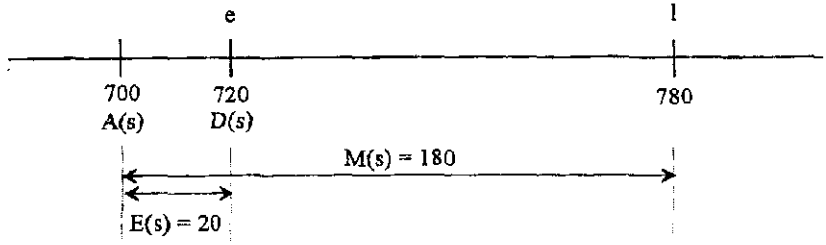
5.2. Factibilidad respecto las ventanas de tiempo

Sea s un elemento en la solución actual, se define:

A(s): Tiempo de llegada del vehículo a s,
D(s): Tiempo de salida de s; se tiene que $D(s) = \max\{A(s), e_s\}$;
M(s): Margen de tiempo en la llegada a s, es decir, $M(s) = l_s - A(s)$;
E(s): Tiempo de espera del vehículo en s, es decir, $E(s) = \max\{e_s - A(s), 0\}$;

estas variables son inmediatas de calcular en cada iteración; a continuación se muestra un gráfico con un ejemplo que ilustra estas definiciones:

FIGURA 3
Ejemplo de la llegada del vehículo a un punto s, con $e_s = 720$ y $l_s = 780$; si la llegada A(s) se produce en 700, entonces el tiempo de salida es D(s) = 720; el tiempo de espera, E(s), es 20, y el margen, M(s), es 80



Sea *cadena* = *p* – p+1 – ... q una cadena en una ruta en la solución actual, se define

maxretraso(cadena) = Máximo retardo en la llegada a p, que no produce *violaciones* de las ventanas de tiempo en los puntos de *cadena;*

se tiene que los valores de *maxretraso* se pueden obtener de la forma siguiente:

$$maxretraso(cadena) = \min_{s=p...q} \{M(s) + \Sigma_{r=p,...,s-1} E(r)\};$$

análogamente se define:

maxadelanto(cadena) = Máximo adelanto en la llegada a r(p), es decir, disminución de A(p), que no produce tiempo de espera en dicha cadena (i.e. decir, se mantiene A(s) e_s para s = p,...,q); obviamente si en algún punto de *cadena* existe espera *entonces* el valor de *maxadelanto(cadena)* es 0;

los valores de *maxadelanto* se pueden calcular como sigue:

$$maxadelanto(cadena) = \min_{s=p...q} \{\max[0, A(s)-e_{r(s)}]\}.$$

Obviamente, tanto como el cálculo de *maxretraso* como de *maxadelanto* se puede plantear de forma recursiva. Finalmente, sea para una determinada cadena p – p+1 –...– q, *adelanto-inicial* el tiempo en que se adelanta la llegada a p, y *adelanto-final* el tiempo en que se adelanta la salida de q, se tiene que

$$adelanto\text{-}final = \min \{adelanto\text{-}inicial, maxadelanto\},$$

análogamente sean *retraso-inicial y retraso-final,* respectivamente, los atrasos en los tiempos de llegada a p y salida de q, se tiene que

$$retraso\text{-}final = \max \{retraso\text{-}inicial - \Sigma_{s=p...q} E(s), 0\}.$$

6. ELECCIÓN DEL PARÁMETRO α

El parámetro α que aparece en la descripción de Procedimiento Variante MTHG (Resolver Problema de Asignación) en al apartado 3, va a determinar el grado de aleatoriedad o 'avidez' de la solución inicial: cuanto mayor sea α mayor es la Lista Restringida de Candidatos en cada paso y por tanto más aleatoria es la solución final.

Para determinar qué valor de α es el que aporta mejores soluciones se han simulado 50 problemas con 40 puntos de descarga, y otros 50 problemas con 20 puntos de descarga y 20 de recogida, ejecutándose el algoritmo GRASP descrito en los apartados anteriores para cada problema y para diferentes valores de α (0,05, 0,10, 0,15,..., 0,95). Los datos de cada problema 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 kms.). (Es decir, consideramos una velocidad de 60 kms/hora). El coste por km. es de 1 u.m.
- A e_i y l_i, para i = 2,..,n1+n2, se les asigna respectivamente dos valores enteros aleatorios generados uniformemente: entre 600 y 720 (minutos) en el primer caso y entre 960 y 1080 en el segundo. Se hace e_i igual a 480 y l_i igual a 1200.
- A cada q(i), i = 2,..,n1+n2, se le asigna 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 1000 y 1200 u.m. respectivamente.

A continuación se muestra las tabla con los resultados del coste medio para cada valor de a (en itálica los mejores resultados):

TABLA 1
40 puntos de descarga

α	C. medio	α	C. medio	α	C. medio	α	C. medio
0,05	14.865,88	**0,30**	14.873,62	**0,55**	14.868,12	**0,80**	14.881,78
0,10	14.865,40	**0,35**	14.865,40	**0,60**	14.873,98	**0,85**	14.881,78
0,15	14.871,22	**0,40**	14.869,42	**0,65**	14.877,44	**0,90**	14.881,78
0,20	14.867,10	**0,45**	14.869,08	**0,70**	14.878,84	**0,95**	14.881,78
0,25	14.868,74	**0,50**	14.872,64	**0,75**	14.877,04		

TABLA 2
20 puntos de descarga y 20 de carga

α	C. medio	α	C. medio	α	C. medio	α	C. medio
0,05	7.671,4	**0,30**	7.661,12	**0,55**	7.660,44	**0,80**	7.660,86
0,10	7.688,92	**0,35**	7.659,50	**0,60**	7.659,80	**0,85**	7.661,32
0,15	7.663,98	**0,40**	7.661,10	**0,65**	7.660,22	**0,90**	7.660,84
0,20	7.663,66	**0,45**	7.663,68	**0,70**	7.660,88	**0,95**	7.659,92
0,25	7.664,36	**0,50**	7.657,22	**0,75**	7.659,22		

Para los problemas con 40 puntos de descarga los mejores resultados se obtienen para a=0,1 y 0,35; para los problemas con 20 puntos de descarga y carga los mejores resultados corresponden a $\alpha = 0,5$. Por tanto, éstos son los valores que se usarán ($\alpha = 0,35$ y 0,5 respectivamente) en el siguiente apartado.

7. RESULTADOS COMPUTACIONALES

Para contrastar la 'calidad' de la solución obtenida por el Metaheurístico GRASP diseñados en este trabajo, así como los Metaheurísticos Temple Simulado y Búsqueda Tabú diseñados en el trabajo anterior (Pacheco y Delgado, 1998), se ha diseñado un *algoritmo compuesto* consistente en:

- Obtención de una solución inicial por una adaptación del algoritmo de Fisher & Jaikumar (1981) a este modelo; más concretamente, se ejecuta el algoritmo diseñado en la sección 3, Fase constructiva, pero con las siguientes consideraciones: la Resolución del Problema de Asignación es determinística, i.e., se elige en cada paso el i correspondiente a *bmax;* se repite todo el proceso varias veces, tantas como puntos de vista tenga el problema, comenzando en cada una de ellas la construcción de los ángulos *beta* con un ángulo *alfa* diferente, y se determina como solución inicial la mejor de las obtenidas en estas repeticiones.
- Aplicación a esta solución inicial de un proceso de búsqueda local, con una estrategia de mayor descenso, tal como se describe en la sección 4, pero sin limitar el tamaño máximo de las cadenas a recolocar. De esta forma los vecindarios resultantes son mayores.

Asimismo se han simulado 10 problemas con 20 puntos de descarga y 20 de recogida, y otros 10 problemas con 40 puntos de descarga. Los datos de cada problema se definen como en el apartado anterior.

El Metaheurístico Temple Simulado usa como solución inicial la dada por una adaptación para este modelo del algoritmo de *inserción más cercana.* (Una colección exhaustiva de estos métodos constructivos para el TSP se puede encontrar en el trabajo de Golden y otros, 1980). Los algoritmos se han programado en PASCAL, utilizando los compiladores BORLAND PASCAL 7.0.

A continuación se muestran los resultados obtenidos (Coste y desviación porcentual de la mejor solución):

TABLA 1
20 puntos de descarga y 20 de carga

	Compuesto	Búsqueda Tabú	Temple Simulado	GRASP
n.1	8.142 1,33	8.216 2,28	8.133 1,24	8.033* **0**
n.2	8.168 0,65	8.174 0,73	8.390 3,39	8.115* **0**
n.3	8.287 2,46	8.178 1,11	8.199 1,37	8.088* **0**
n.4	6.890 0,80	7.067 3,39	7.068 3,41	6.835* **0**
n.5	9.539 1,12	9.434 0,01	9.473 0,42	9.433* **0**
n.6	6.849 0,23	7.063 3,37	7.018 2,71	6.833* **0**
n.7	8.340 1,32	8.478 3,00	8.437 2,50	8.231* **0**
n.8	9.466 1,95	9.363 0,84	9.379 1,01	9.285* **0**
n.9	8.118 0,72	8.167 1,25	8.400 4,22	8.060* **0**
n.10	9.435 1,56	9.420 1,40	9.605 3,39	9.290* **0**

* indica la mejor solución de cada problema.

TABLA 2
40 puntos de descarga

	Compuesto	Búsqueda Tabú	Temple Simulado	GRASP
n.1	15.998 0,46	15.925* **0**	16.062 0,86	15.994 0,43
n.2	15.793 1,05	15.679 0,73	15.807 3,39	15.629* **0**
n.3	14.692 2,46	14.324 1,11	14.658 1,37	14.599* **0**
n.4	15.684 0,37	15.652 0,17	15.626* **0**	15.633 0,04
n.5	13.358 0,65	13.302 0,23	13.321 0,37	13.272 **0**
n.6	16.137 0,19	16.133 0,17	16.299 1,20	16.106* **0**
n.7	15.783 0,79	15.667 0,04	15.660* **0**	15.723 0,40
n.8	13.099* **0**	13.305 1,57	13.310 1,61	13.103 0,03
n.9	14.482 0,06	14.568 0,65	14.658 1,28	14.473* **0**
n.10	16.407 1,30	16.197* **0**	16.392 1,20	16.197* **0**

* indica la mejor solución de cada problema.

Según se pueden apreciar en estos resultados GRASP ofrece mejores resultados en todos los problemas de la Tabla 1 y en 6 de los 10 problemas de la Tabla 2, separándose en el peor de los casos apenas un 0,43% de la mejor solución. Además, aunque en estos resultados no se indica el tiempo de computación utilizado, hay que indicar que el tiempo empleado por GRASP era menor que las de los otros metaheurísticos y algo mayor que el algoritmo compuesto.

8. REFERENCIAS Y BIBLIOGRAFIA

BODIN, L. D. y GOLDEN, B. L. (1981): «Classification in Vehicle Routing and Scheduling», *Networks,* vol. 11, n.° 2, 97-108.

CLARKE, G. y WRIGHT, J. W. (1964): «Scheduling of Vehicles from a Central Depot to a Number of Delivery Points», *Oper.Res.*, 12, 568-581.

CUADRAS, J. M. (1991): *Métodos de Análisis Multivariante*, Ed. PPU (Publicaciones y Promociones Universitarias), Barcelona.

DESROCHERS, M., LENSTRA, J. K., SAVELSBERGH, M. W. P. y SOUMIS, F. (1988): «Vehicle Routing with Time Windows: Optimization and Approximation», en *Vehicle Routing: Methods and Studies* (Studies in Management Sciences and Systems, vol. 16), eds.: GOLDEN, B. L. y ASSAD, A. A., Nort-Holland, 65-84.

FEO, T. A. y RESENDE, M. G. C. (1989): «A Probabilistic heuristic for a computationally difficult Set Covering Problem», *Operations Research Letters,* 8, 67-71.

— (1995): «Greedy Randomized Adaptative Search Procedures», *Journal of Global Optimization,* 2, 1-27.

FISHER, M. L. y JAIKUMAR, R. (1981): «A Generalized Assignment Heuristic for Vehicle Routing», *Networks,* vol. 11, n.° 2, 109-124.

GOLDEN, B., BODIN, L., DOYLE, T. y STEWART, W. Jr. (1980): «Approximate Traveling Salesman Algorithms», *Operations Researseh,* vol. 28, n.° 3, parte II, 694-711.

HAOUARI, M., DEJAX, P. y DESROCHERS, M. (1990): «Les Problems de Tournées avec Contraintes des Fenêtres de Temps: L'Etat de l'Art». *Recherche Operationnelle/Operations Research,* vol. 24, n.° 3, 217-244.

KONTORAVDIS, G. y BARD, J. F. (1995): «A GRASP for the Vehicle Routing Problem with Time Windows», *ORSA Journal on Computing,* 7: 10-23.

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

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

LIN, S. y KERNIGHAN, B. W. (1973). «An Effective Heuristic Algorithm for the Traveling Salesman Problem», *Operations Research,* vol. 20, 498-516.

MARTELLO, S. y TOTH, P. (1990): *Knapsack Problems*, John Wiley & Sons, Chichester.

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

OR, I. (1976): *Traveling Salesman Type Combinatorial Problems and their Relations to the Logistics of Blood Banking,* Ph. 7hesis, Dpt. of Industrial Engineering and Management Sciences, Northwestern Univ.

PACHECO, J. (1994): *Problemas de Rutas con Ventanas de Tiempo*, Tesis Doctoral leída en el Dpto de Estadística e I. Optva. de la Facultad de Matemáticas de la U. Complutense de Madrid, mayo 1994.

— (1997): *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 1997.

PACHECO, J. y DELGADO, C. (1995): *El Problema del Viajante con Carga y Descarga y Ventanas de Tiempo: Algoritmos heurísticos.* XXII Congreso Nacional de Estadística e Investigación Operativa, Sevilla, noviembre 1995.

— (1996): *Adaptación del Algoritmo de Or al VRPTW con Carga y Descarga simultánea,* X Reunión ASEPELT-ESPAÑA, Albacete, junio 1996.

— (1997a): *Metaheurístico para el VRPTW con carga y descarga simultánea,* XXII Congreso Nacional de Estadística e Investigación operativa, Valencia, abril 1997.

— (1997b): «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.

— (1998): *Diseño de Metaheurísticos híbridos para Problemas de Rutas con Flota Heterogénea,* XII Reunión ASEPELT-ESPAÑA, Córdoba, junio 1998.

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

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

ROSING, K. E., REVELLE, C. S., ROLLAND, E., SCHILLING, D. A. y CURRENT, J. R. (1998): «Heuristic Concentration and Tabu Search: A head to head comparison», *European Journal of Operational Research,* 104, 93-99.