Estudios de Economía Aplicada
Nº 7, 1997, Págs. 79-100

Problemas de rutas con ventanas de tiempo y carga y descarga simultanea: diseño de filtros en algoritmos de intercambio (caso de un vehículo)(+)

PACHECO BONROSTRO , JOAQUÍN ANTONIO(*)
Universidad de Burgos
DELGADO SERNA, CRISTINA ROCIO(**)
S.A.A.T. (Sociedad Anónima de Aduanas y Transportes) Villafría- Burgos

Esta nueva versión incluye todas las correcciones sugeridas por el evaluador, las cuales nos han parecido oportunas y por las que le quedamos muy agradecidos.

RESUMEN

En este trabajo se propone un método para diseñar algoritmos para el *Problema de Rutas de Vehículos con Ventanas de Tiempo, y con Carga y Descarga simultánea (VRPTW mixto)*, basados en las conocidas técnicas de mejora o intercambio r-óptimos. Este método consiste en incorporar un mecanismo que *filtre* los intercambios que van a dar lugar a rutas factibles. El resultado es que el chequeo de los intercambios factibles de esta forma va a ser mucho más rápido que el realizarlo uno por uno, a la vista de los resultados computacionales. En este trabajo se adapta este método al algoritmo de Or, consiguiéndose ahorros porcentuales considerablemente.

Palabras clave: Problemas Ruta de Vehículo Mixtos con Ventanas de tiempo, Algoritmo de Or, Chequeo de Intercambios.

ABSTRACT

In this work on proposed a method to design heuristics for the VRPTW with Delevery and Pickup, based in the well-know improvement algorithms r-opt. This method consist of to incorporate a

(+) El algoritmo descrito en este trabajo se desarrolla dentro del proyecto S.I.L.T, desarrollado por *SAAT* y aprobado y cofinanciado por el *Centro de Desarrollo Tecnológico Industrial* con referencia 950107.

mechanisme to filter the interchanges that generate feasibles solutions. The resultat is a checking of the interchanges very faster to checking one for one, according computational results. In this work we adoptated our checking method to Or algorithm, and we obtain importants porcentuals.

KeyWords: Mixed Vehicle Routing Problem with Time Windows, Or Algorithm, Interchange checking.

Artículo recibido en diciembre de 1996. Revisado en enero de 1997.

1.-Introducción

El Problema de Rutas de Vehículos con Ventanas de tiempo y con Carga y Descarga Simultánea o sencillamente VRPTW Mixto (Mixed VRPTW) 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 *pales* q(i), i = n1+1,..., n1+n2, y llevarlas al origen 1. Para cumplir estos requerimientos se dispone de una flota de vehículos homogénea con capacidad de carga de Q *palés* cada uno; 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.

El objetivo del problema es diseñar un conjunto de rutas de forma que:

- 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 Q;
- Cada punto i, i = 2,..,n1+n2, sea visitado exactamente una vez:
- Se respeten los intervalos de tiempo de visita;
- La distancia total recorrida sea mínima. (En algunas ocasiones se requiere como objetivo preferente minimizar el número de vehículos utilizados).

(En adelante denotaremos n = n1+n2, es decir el número de puntos que intervienen en el problema).

Existen muchos algoritmos de solución para el VRP y el VRPTW en la literatura. Se pueden encontrar recopilaciones de los principales en trabajos como los de Bodin y Golden (1.981), Desrochers y otros (1.988), Haouri y otros (1.990), y Laporte (1.992). En los últimos años han tomado importancia el desarrollo de algoritmos basados en procesos de *temple simulado* y *búsqueda tabú* especialmente a partir del trabajo de Gendreu y otros, (1.991); como así se refleja en el trabajo de Osman, (1.993), y mas recientemente en el de Campos y Mota (1.995).

Sin embargo, para el VRP (o VRPTW) Mixto existen pocas referencias: Pacheco, (1.994), y Pacheco y Delgado, (1995) y (1.996). Existen trabajos que tratan otro modelo que puede considerarse como un caso particular del VRPTW Mixto: el *VRP with Backhauls*, (VRPB), problema que tiene lugar cuando algún punto actúa a la vez de punto de descarga y carga. Entre estos destacamos el de Casco y otros, (1.988), Deif y otros, (1.984), Goetschalkx y otros, (1.986), y Golden y otros, (1.985).

En este trabajo se propone un algoritmo para el VRPTW Mixto para el caso de un solo vehículo, consistente en una extensión y adaptación a este problema del algoritmo de Or para el TSP, (1.976). El algoritmo de Or es una variante de los conocidos algoritmos r-óptimos desarrollados por Lin, (1.965) y Lin y Kernighan, (1.973), para el Problema del Viajante (TSP) simétrico. En el caso del algoritmo de Or se toma r = 3 y se reduce la búsqueda de los 3-intercámbios a aquellos que supongan una recolocación de cadenas[1] de 1, 2 o 3 elementos entre otros dos consecutivos en la solución actual. La ventaja de este método es que el número de operaciones que se realiza es de $\theta(n^2)$, frente al algoritmo 3-óptimo original que utiliza $\theta(n^3)$, (Or, (1.976)). Además, como se verá mas adelante, el algoritmo de Or se puede utilizar en problemas asimétricos. La eficacia de este algoritmo para el TSP ha sido contrastada en el trabajo de Nurmi, (1.991).

Nuestra adaptación consiste en incorporar un método que chequee la factibilidad de las soluciones a que den lugar los intercambios sin tener que realizar este chequeo uno a uno, intercambio a intercambio. Esto va a suponer

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

un ahorro considerable en el tiempo de computación y permite que el algoritmo pueda ser utilizado en problemas de gran tamaño en ordenadores personales.

El trabajo se estructura de la siguiente manera: en el segundo apartado se expone la idea básica, en el tercero se aborda el diseño del algoritmo para problemas con un vehículo, en el cuarto se muestran una serie de resultados computacionales, y en el quinto se establecen las conclusiones.

2.- Idea básica del algoritmo de Or

Or, (1.976), propone restringir la búsqueda a los 3-intercambios en los que cadenas de uno, dos o tres puntos consecutivos en la solución actual son recolocadas entre otros dos clientes (ver figura 1). Esto hace reducir el número de 3-intercambios a considerar a una cantidad de orden $\theta(n^2)$. Nótese que con estos intercambios no se cambia el sentido de los diferentes tramos.

En nuestro caso seguiremos la misma idea, pero no pondremos límite al tamaño de la cadena a recolocar (salvo el determinado por el tamaño del problema); además se debe chequear la factibilidad de cada posible recolocación respecto a las restricciones del problema

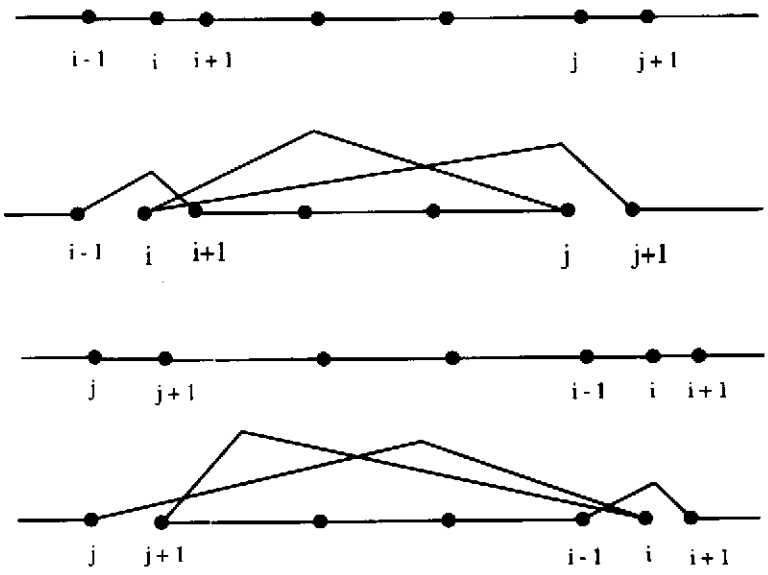


Figura 1.- Posible recolocación del elemento i hacia adelante y hacia atrás entre j y j+1.

En el caso general de recolocación de cadenas de k elementos consecutivos, comenzando en la posición i, entre dos puntos que ocupan las posiciones j y j+1, como muestra la figura 2, supone suprimir los arcos (i-1,i), (i+k-1,i+k), (j,j+1) e incorporar los arcos (i-1,i+k), (j,i) y (i+k-1,j+1).

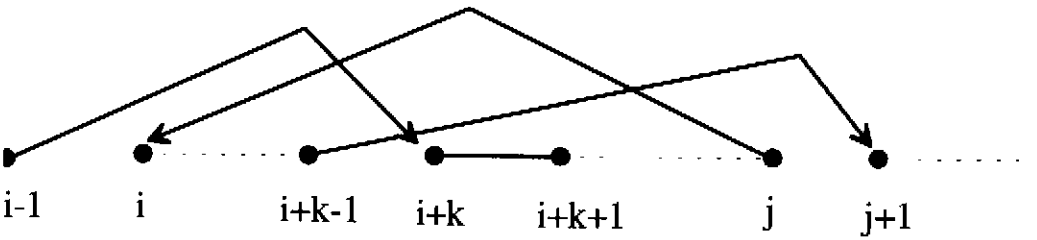


Figura 2.- Recolocación de una cadena de k elementos.

En el caso de un solo vehículo, que tratamos en este trabajo, cada solución se compone de n+1 elementos: por comodidad se añade al final de las soluciones el punto 1 que se interpreta como la vuelta al origen. Se define i como el elemento inicial de la cadena; k como el número de elementos que la componen y j el punto tras el cual se recoloca. Los valores que puede tomar i varían desde 2, hasta n-1. Para cada valor de i los valores que puede tomar k varían desde 1 a n-i. Para cada valor de i y cada valor de k, los posibles valores que puede tomar j varían desde i+k hasta n, (en nuestra adaptación solo se considera la recolocación hacia adelante).

El algoritmo básicamente actúa de la forma siguiente: a partir de una solución inicial examina todos los intercambios o recolocaciones posibles que sean factibles. Para cada uno de estos intercambios factibles calcula la variable *ganancia* definida como la suma de las distancias de los arcos que se suprimen

menos la suma de las distancias de los arcos que se añaden. Se realiza el intercambio correspondiente al mayor valor de ganancia, y se vuelve a repetir el proceso con la nueva ruta. El algoritmo termina cuando el mayor valor de ganancia no sea positivo.

El chequeo de la factibilidad de cada intercambio, es decir de la nueva solución a la que este da lugar, puede suponer un tiempo de computación excesivo. En los apartados siguientes se va a proponer un método que de forma rápida examine las recolocaciones factibles para cada cadena, es decir, para cada valor de i y de k determinar los valores de j factibles. Dicho método se basa en la definición de un conjunto de variables globales que se actualizan de forma recursiva utilizando un número constante de operaciones.

3.-Diseño del algoritmo (caso de un solo vehículo)

El chequeo de la factibilidad de los intercambio en el problema que se analiza supone examinar si estos intercambios pueden hacer que las rutas resultantes violen o no las restricciones de capacidad del vehículo en cada momento y los horarios de visita de cada punto.

A partir de ahora se define la variable r(s), s = 1,.., n+1, como el punto que ocupa la posición i-ésima en la ruta actual en cada iteración (siempre r(n+1)=r(1)=1); así mismo, para cada posible valor de i, k, j se definen las siguientes cadenas:

cadena_0 : 1 -r(2)-...-r(i-1)

cadena_1 : r(i)-r(i+1)-....-r(i+k-1)

cadena_2 : r(i+k)-r(i+k+1)-...-r(j)

cadena_3 : r(j+1)-r(j+2)-...-r(n)-1;

obsérvese que la recolocación en realidad supone un intercambio de posición entre *cadena_1* y *cadena_2*.

3.1- Factibilidad respecto a la carga

Para determinar que intercambios pueden *violar* la capacidad del vehículo definimos las siguientes variables:

esp_libre(s) : Espacio libre disponible en el vehículo después de visitar el punto r(s); para s = 2,...,n:

los valores de la variable *esp_libre* se pueden calcular fácilmente en cada iteración. Así mismo sea *cadena* = r(p)-r(p+1)-...-r(q), cualquier cadena de la solución actual se define

esp_modificado(*cadena*) :Espacio ocupado (si es positivo) o liberado (si es negativo) en el vehículo tras visitar los puntos de *cadena*;

se tiene que:

esp_modificado(*cadena*) = *esp_libre*(p-1)-*esp_libre*(q);

también se define

minimo_cadena(*cadena*) :Mínimo espacio libre disponible en el vehículo durante la visita a los puntos de *cadena*;

se tiene que

$$minimo_cadena(cadena) = \min_{s=p,\ldots,q} esp_libre(s)$$

y

minimo_cadena(*cadena*)=min{*minimo_cadena*(r(p)-..-r(q-1)), *esp_libre*(q)}.

con

minimo_cadena(r(p)) = *esp_libre*(p) como valor inicial.

La recolocación definida por los valores de i, k y j no afecta a *cadena_0* y *cadena_3* en cuanto a espacio disponible en los vehículos, solamente afecta a *cadena_1* y *cadena_2*. Se tienen las siguientes proposiciones:

Proposición 1

Si *esp_modificado(cadena_1)* + *minimo_cadena(cadena_2)* < 0 entonces la recolocación definida por i, j y k es infactible.

Demostración.- Vale observar que en este caso se violarían las restricciones de capacidad del vehículo en al menos un punto de *cadena_2*.

Corolario 1

Sea la cadena a recolocar definida por los valores de i y k; se define

$$s = \min \{r / r > i+k-1 \text{ y } esp_libre(r) + esp_modificado(cadena_1) < 0\}$$

entonces para los valores de j comprendidos entre s y n la recolocación definida por i, j, k es infactible.

Proposición 2

Si *esp_modificado(cadena_2)* > *minimo_cadena(cadena_1)* entonces la recolocación definida por i, j y k es infactible.

Demostración.- Vale observar que se violarían las restricciones de capacidad del vehículo en al menos un punto de *cadena_1*.

3.2- Factibilidad respecto a las ventanas de tiempo

En este caso hay que tener presente que la recolocación definida por i, j y k además de afectar a los tiempos llegada y salida de los puntos de *cadena_1* y *cadena_2*, también podría afectar a los tiempos de los puntos de *cadena_3*. Para todo s = 2,...,n+1 se van a definir las siguientes variables:

A(s) : Tiempo de llegada del vehículo a r(s),

D(s) :Tiempo de salida de r(s); se tiene que $D(s) = \max\{A(s), e_{r(s)}\}$;

M(s) :Margen de tiempo en la llegada a r(s), es decir, $M(s) = l_{r(s)} - A(s)$;

E(s) :Tiempo de espera del vehículo en r(s), es decir, $E(s) = \max\{e_{r(s)} - A(s), 0\}$;

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

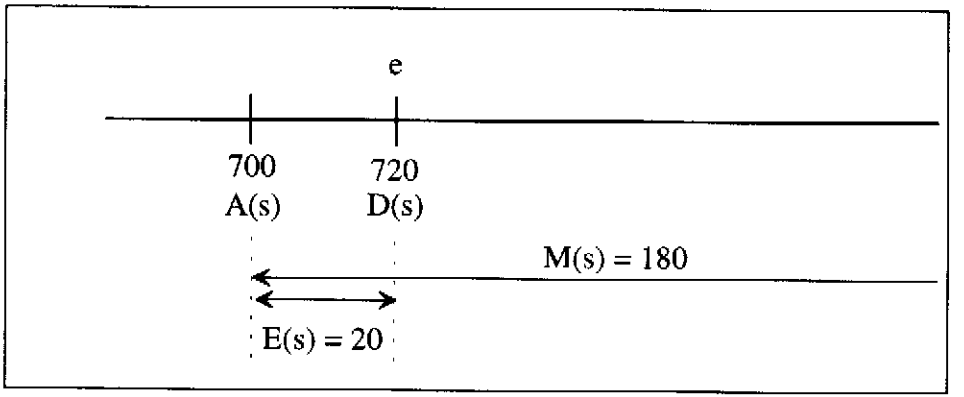


Figura 3.- Ejemplo de la llegada del vehículo a un punto r(s), con $e_{r(s)}$=720 y $l_{r(s)}$=780; si la llegada A(s), se produce en 700, entonces el tiempo de salida es D(s) = 720, el tiempo de espera, E(s), es 20, y el margen, M(s), es 80.

Proposición 3.

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

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

Demostración.- Por inducción en los valores de h:

- para h =0 es trivial: A*(p) = A(p) + x.
- sea cierto para h-1, entonces, $A^*(p+h-1) = A(p+h-1) + \max\{x - \sum_{s=p}^{p+h-2} E(s), 0\}$.

Observando la figura 3 anterior, y teniendo presente las relaciones entre A, D y E, es fácil de comprobar que un retardo en la llegada a un punto repercute en el tiempo de salida en dicho punto, si este retardo es mayor o igual que el tiempo de espera; en este caso el aumento en el tiempo de salida igual a la diferencia entre el retardo y el tiempo de espera; por tanto:

$$D^*(p+h-1) = D(p+h-1) + \max\{\max\{x - \sum_{s=p}^{p+h-2} E(s), 0\} - E(p+h-1), 0\}$$

$$= D(p+h-1) + \max\{x - \sum_{s=p}^{p+h-1} E(s), 0\}$$

luego: $A^*(p+h) = D^*(p+h-1) + t_{r(p+h-1)\, r(p+h)} = A(p+h) + \max\{x - \sum_{s=p}^{p+h-1} E(s), 0\}$.-

Corolario 2.-

Sea H(p,q), p = 1,...,n+1, y q = p,...,n+1, el máximo retardo en la llegada a r(p), es decir incremento en A(p), que no produce violación de las ventanas de tiempo en r(q), (es decir se mantiene $A^*(q) \le l_{r(q)}$); se tiene que:

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

Sea *cadena* = r(p)-r(p+1)-...-r(q) se define

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

tespera(*cadena*) = Tiempo total de espera acumulado en *cadena*;

obviamente *tespera*(*cadena*$=\sum_{s=p}^{q} E(s)$ y *tespera*(*cadena*)*tespera*(r(p)-..-r(q-1))+ E(q); por comodidad si q < p, (es decir si *cadena* no contiene elementos), se define *tespera*(*cadena*) = 0.

Proposición 4

Se tiene que los valores de *maxretraso* se pueden obtener de la forma siguiente
maxretraso(*cadena*) = $\min_{s=p...q}$ H(p,s)= $\min_{s=p...q}$ {M(s)+tespera(r(p)-..-r(s-1))}

Demostración.- Obvio a partir de Proposición 3 y Corolario 2 y la definición de *maxretraso*,-

Corolario 3

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

maxretraso(r(p)-...-r(q)=min{*maxretraso*(r(p)-...-r(q-1)),M(q)+tespera(r(p)-..-r(q-1)}

con

maxretraso(r(p)) = M(p) como valor inicial.-

Este método recursivo puede utilizarse para calcular y actualizar *maxretraso* de *cadena_1* y *cadena_2*, ya que según se incrementan los valores de i, k, y j, se añaden elementos al final de estas cadenas. No ocurre lo mismo en *cadena_3*, ya que a medida que se incorporan elementos en el último lugar de *cadena_2* se eliminan en el comienzo de *cadena_3*, (para cada i y k fijos j se incrementa desde i+k hasta n, y a medida que aumenta j aumenta *cadena2* y disminuye *cadena3*). Para *cadena3* se hace uso de las variables H(p,q) definidas en Corolario 2; obviamente:

maxretraso(*cadena_3*) = $\min_{q=j+1,..,n+1}$H(j+1,q)

también se define las variables

$s^*(p)$ = Valor verificando $\min_{q=p,\ldots,n+1} H(p,q) = H(p,s^*(p))$; para $p= i+k+1,\ldots,n+1$.

Proposición 5

Se tiene que

$$H(p,s^*(p))= \min_{q=p,\ldots,n+1}\{H(i+k+1,q)\}\text{-}tespera(r(i+k+1)\text{-}\ldots\text{-}r(p\text{-}1))$$

$$H(p,s^*(p))=H(i+k+1,s^*(p))\text{-}tespera(r(i+k+1)\text{-}\ldots\text{-}r(p\text{-}1)),\text{para } p=i+k+1,\ldots,n+1$$

Demostración.- Vale ver que $H(p,q)= H(i+k+1,q)$-*tespera*$(r(i+k+1)$-...-$r(p-1)$, para $q = p,..,n+1$,

y por tanto

$$\min_{q=p,\ldots,n+1} H(p,q)= \min_{q=p,\ldots,n+1}\{H(i+k+1,q)\text{-}tespera(r\ (i+k+1)\text{-}\ \ldots r(p\text{-}1)\}.$$

Corolario 4

Se pueden obtener el valor de $s^*(p)$ de forma recursiva de la forma siguiente:

$s^*(p) = p$ si $H(i+k+1,p) < H(i+k+1,s^*(p+1))$
$s^*(p) = s^*(p+1)$ en caso contrario;

con

$s^*(n+1) = n+1$ como valor inicial;

de esta forma:

$$maxretraso(cadena_3) = H(i+k+1,s^*(j+1)) - tespera(r(i+k+1)\text{-}\ldots\text{-}r(j)).$$

Así mismo sea *cadena* = r(p)-r(p+1)-...-r(q), cualquier cadena en la actual solución 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, (es decir, se mantiene $A(s) \geq e_{r(s)}$ para s = p,...,q); obviamente si en algún punto de *cadena* ya existe espera *maxadelanto*(*cadena*) es 0.

Proposición 6

Los valores de *máximoadelanto* puede obtenerse de la siguiente forma:

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

Corolario 5

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

$$maxadelanto(r(p)-...-r(q)) = \min\{maxadelanto(r(p)-...-r(q-1)), \max[0, A(q)-e_{r(q)}]\}.$$

con

$$maxadelanto(r(p)) = \max[0, A(p)-e_{r(p)}] \text{ como valor inicial.-}$$

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

$$adelanto_final = \min \{adelanto_inicial, maxadelanto\};$$

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

$$retraso_final = \max \{retraso_inicial - \sum_{s=p}^{q} E(s), 0\}.$$

De esta forma para cualquier recolocación se puede determinar secuencialmente:

- La alteración en el tiempo de llegada a *cadena_2*, la violación o no de las ventanas de tiempo en los puntos de esa cadena, y la alteración en el tiempo de salida;

- Para *cadena_1* se realiza el mismo proceso;

- Para *cadena_3* finalmente se determina el adelanto o atraso de llegada, y si esta alteración respeta o no las ventanas de tiempo.

3.3.- Estructura Algorítmica y Estudio de la Complejidad

El esquema algorítmico con un estudio de la factibilidad de las recolocación una a una es el siguiente:

Paso 0: Partir de una solución inicial
Paso 1: Para i = 2 hasta n-1:
Para k = 1 hasta n-i:
Para j = i+k hasta n:
Chequear la factibilidad del intercambio definido por i, j, k; Si es fa ctible calcular el valor de ganancia correspondiente (tal como se definió en el apartado 2) .

Paso 2: Si el valor máximo de *ganancia* es positivo, realizar el intercambio correspondiente y volver al paso 1; en caso contrario parar.

El esquema del algoritmo que se propone en este trabajo, basado en los anteriores resultados es el siguiente:

Paso 1: Determinar valores de A, D, M, E y *esp_libre*.
Para i = 2 hasta nt-2:
-Iniciar datos de *cadena_1*;
Para k = 1 hasta nt-i-1:
-Actualizar datos de *cadena_1*;

-Iniciar datos de *cadena_2*;
-Calcular datos de *cadena_3*;
Para j = i+k hasta nt-1:
-Actualizar datos *cadena_2*;
-Chequear restricciones de capacidad y tiempo en *cadena_1* y *cadena_2*;
-Determinar *maxretraso* y chequear restricciones de tiempo en *cadena_3*
-Si el intercambio definido por i j y k es factible calcular el valor de *ganancia* correspondiente.

El *Paso 0* y el *Paso 2* no cambian.

Los datos que se consideran para *cadena_1* y *cadena_2* son los valores correspondientes a las variables *esp_modificado*, *minimo_cadena*, *tespera*, *maxretraso* y *maxadelanto*. Según los resultados desarrollados anteriormente, la inicialización y actualización de estas variables así como el chequeo de la factibilidad en estas cadenas requieren un número contante de operaciones no dependiente del tamaño del problema.

Los datos que hay que calcular referentes a *cadena_3* son los valores de *ttespera*(r(i+k+1)-..-r(p)), H(i+k+1,p) y s*(p), para p = i+k+1,..,n+1. El cálculo de cada uno de estos valores así como el cálculo de *maxretraso* y el chequeo de la factibilidad respecto las ventanas de tiempo en *cadena_3* necesitan un número constante de operaciones.

Proposición 7.-

El número de operaciones requerido para en cada iteración (paso 1) en el primer esquema es del orden $\theta(n^4)$.

Demostración.- El número de intercambios a estudiar es:

$$\sum_{i=2}^{n-1}\left(\sum_{k=1}^{n-i}\left(\sum_{j=i+k}^{n} 1\right)\right)=\frac{1}{6}(n+1)^3-(n+1)^2+\frac{11}{6}(n+1)-1.$$

esto es fácil de comprobar recordando que:

$$\sum_{s=1}^{x} s = \frac{x(x+1)}{2}, \quad \text{y} \sum_{s=1}^{x} s^2 = \frac{1}{3}x^3 + \frac{1}{2}x^2 + \frac{1}{6}x.$$

Por otra parte el número de operaciones para examinar la factibilidad de cada uno de estos intercambios es $\theta(n)$, (guardar la solución resultante por ese intercambio en una variable auxiliar y examinar la factibilidad elemento a elemento). Por consiguiente examinar la factibilidad de todas las iteraciones requiere $\theta(n^4)$ operaciones.-

Proposición 8

El número de operaciones requerido para en cada iteración en el segundo esquema es del orden $\theta(n^3)$.

Demostración.- El número de operaciones a realizar es

$$F\cdot(n+1) + \sum_{i=2}^{n-1}\left(G + \sum_{k=1}^{n-i}\left(R + L(n+1-i-k) + \sum_{j=i+k}^{n} T\right)\right)$$

donde:

F = Es el número de operaciones requeridas para determinar los valores de A, D, M, E y *esp_libre* para cada elemento de la solución;
G = N° de operaciones requeridas para iniciar los datos de *cadena_1*;
R = N° de operaciones requeridas para actualizar datos de *cadena_1* e iniciar datos de *cadena_2*;
L = N° de operaciones requeridas para calcular *ttespera*(r(i+k+1)-..-r(p)), H(i+k+1,p) y s*(p), para cada valor de p = i+k+1,..,nt;
T = N° de operaciones requeridas para actualizar datos de *cadena_2*, calcular *maxretraso* de *cadena_3*, chequear la factibilidad y calcular el valor de *ganancia*.

Se tiene que la anterior expresión va a ser igual a

$$F\cdot(n+1)+G\cdot(n-2)+W\frac{(n-2)(n-1)}{2}+V\left(\frac{(n-2)^3}{6}+\frac{(n-2)^2}{4}+\frac{(n-2)}{12}\right)$$

donde V = L+T y W= R+(V/2); por consiguiente el orden de operaciones es $\theta(n^3)$.-

Como se ha comentado en la introducción, se ha realizado un conjunto de pruebas para comparar los tiempos de computación que utilizan ambos esquemas algorítmicos: se pone de manifiesto la reducción de tiempo del segundo sobre el primero, confirmándose los resultados teóricos de esta sección.

4.- Resultados computacionales

Como se ha comentado en el apartado 2, el chequeo de la factibilidad de cada intercambio uno a uno puede emplear mucho tiempo de computación. El objeto del apartado 3 es desarrollar un método para determinar de forma más *rápida* que intercambios son factibles y cuales no. Este método se basa, como se ha descrito anteriormente, en definir un conjunto de variables globales, cuyo carácter recursivo permite ahorrar cálculos y que determinan la factibilidad de los intercambios.

Para establecer la eficacia del método propuesto en el caso de un solo vehículo, se han simulado problemas de diferente tamaño, concretamente 20 para cada tamaño. Estos problemas se han resuelto utilizando el algoritmo con el método de chequeo propuesto y chequeando los intercambios uno a uno (tal como se describe en el apartado 3.3). Los datos de cada uno de los problemas 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).

- 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_1 igual a 480 y l_1 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. Se define $Q = \max\{\sum_{i=2}^{n1} q(i), \sum_{i=n1+i}^{n1+n2} q(i)\}$.

En todos los problemas se ha utilizado como solución inicial la dada por una adaptación propia para este modelo del algoritmo de inserción más cercana. (Una colección exhaustiva de estos métodos constructivos para el TSP se puede encontrar en el trabajo de Golden y otros, (1.980)). Los algoritmos se han programado en PASCAL, utilizando el compilador BORLAND PASCAL 7.0; el equipo usado es un ordenador tipo PC PENTIUM-100 Mhz.

Se han simulado problemas de los siguientes tamaños: 10, 20, 30, 40, 50, 60 y 70 puntos: A continuación se muestra un cuadro con los siguientes resultados medios: tiempo de computación en segundos utilizando el método de propuesto y chequeando los intercambios uno a uno, y reducción porcentual

Tamaño	Chequeo uno a uno	Método propuesto	Reducción Porcentual
10	0'022	0'006	72'72
20	0'445	0'077	82'69
30	4'731	0'592	87'48
40	14'276	1'464	89'74
50	45'024	4'016	91'08
60	109'297	8'375	92'33
70	305'940	20'359	93'34

Se ha de señalar que en todos los casos el tamaño de los problemas tratados se ha tenido que limitar a la vista de la evolución de los tiempos de computación en el *chequeo uno a uno* de las recolocaciones. Sin embargo, esta misma evolución permite concluir también que con el método propuesto se podrían tratar problemas de tamaño mucho mayor.

5.- Resumen y conclusiones

A la vista de los resultados anteriores se pueden extraer las siguientes conclusiones:

- Los métodos de intercambio r-óptimos tienen una amplía aceptación dentro de las técnicas heurísticas para problemas de rutas; en muchos trabajos se reflejan su eficacia, como el mencionado de Nurmi, (1.991) en el que se analizan diversos algoritmos para el TSP. Sin embargo a medida que aumenta la complejidad de los problemas y se debe chequear la factibilidad de cada intercambios, el tiempo de computación aumenta de forma importante. Se hace pues, necesario incorporar métodos alternativos al del chequeo intercambio a intercambio.

- Este trabajo propone una técnica para determinar los intercambios factibles, en el caso de la adaptación al modelo tratado del algoritmo de Or, basado en definiciones de variables globales que sean calculadas de forma recursiva, con el consiguiente ahorro de tiempo computacional. Una idea semejante a esta fue propuesta para el TSPTW por Savelsbergh, (1.985); en el modelo que se trataba en dicho trabajo se minimizaba el tiempo total empleado en la ruta, por lo que se simplificaban los cálculos al tratar simultáneamente el problema de la factibilidad y *deseabilidad*, en el sentido de ahorro de tiempo, de los intercambios. También para el TSPTW, Salomon y otros, (1.988), proponen un método para filtrar intercambios factibles basado en la definición de una matriz de precedencia que prevee en muchos casos la infactibilidad de los intercambios; sin embargo este método no siempre detecta todos los intercambios infactibles y por consiguiente es necesario un chequeo posterior. En cualquier caso, según los autores, se conseguía un ahorro medio del 69% para un conjunto de 20 problemas, en la adaptación de Or, y se llegaba a un ahorro del 85% en los algoritmos 3-óptimos.

- En este trabajo se han tratado un modelo más complejo que el TSPTW, al incorporarse carga y descarga simultánea, y las ahorros conseguidos permiten tratar problemas en ordenadores personales de tamaño muy superior a los que se podrían

resolver sin este método de filtrado; al menos esto permite concluir la evolución de los tiempos de computación en los diferentes casos. Concretamente, en nuestro caso, se han conseguido resolver problemas con 200 puntos, en menos de 115 segundos de tiempo medio.

Este es el primer algoritmo diseñado para el VRP Mixto, por consiguiente no pueden hacerse comparaciones que determinen la calidad de este; en cualquier caso al tratarse de un algoritmo de mejora siempre puede combinarse con otros heurísticos que se puedan diseñar o adaptar posteriormente. Un aspecto interesante, con relación a este punto, sería el estudio de la influencia de la solución inicial tanto en el tiempo de computación como en la solución final. Por otra parte el método diseñado para el chequeo de intercambios puede usarse para determinar el conjunto de soluciones vecinas de una dada y formar parte del diseño de Metaheurísticos basados en procesos de *temple simulado* y *búsqueda tabú*. Se tiene previsto tratar este punto, así como el caso de varios vehículos en trabajos posteriores.

Referencias y bibliografía

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

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

CASCO,D.O., GOLDEN,B.L. y WASIL,E.A. (1.988). Vehicle Routing with Backhauls: Models, Algorithms, y Case Studies. In *Vehicle Routing: Methods y Studies*, (Studies in Management Sciences y Systems, vol.16), eds: GOLDEN,B.L. y ASSAD,A.A., Nort-Holland, 7-46.

DEIF, I. y BODIN,L. (1.984). Extension of the Clark y Wrigth Algorithm for Solving the Vehicle Routing Problem with Backhauling. Proceedings of the *Babson Conference on Software Uses in Transportation y Logistics Management* (A.E.KIDDER, edit.), 75-96, Babson Park, Massachussets.

DESROCHERS,M., LENSTRA,J.K. SAVELSBERGH,M.W.P. y SOUMIS,F. (1.988). Vehicle Routing with Time Windows: Optimization y Approximation. In *Vehicle Routing: Methods y Studies*, (Studies in Management Sciences y Systems, vol.16), eds: GOLDEN,B.L. y ASSAD,A.A., Nort-Holland, 65-84.

FISHER,M.L. y JAIKUMAR,R. (1.981). A Generalized Assignment Heuristic for Vehicle Routing. *Networks*, vol.11, nº 2, 109-124.

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

GOETSCHALCKX,M. y HORSLEY, C. (1.986). The Vehicle Routing Problem with Backhauls. *Material Handling Researcsh Center*, Dept. of Industrial y Systems Engineering, Georgia Institute of Technology.

GOLDEN,B., BAKER,E., ALFARO,J. y SCHAFFER,J. (1.985). The Vehicle Routing Problem with Backhauling: Two Approaches. *Twenty-First Annul Meeting of S.E. TIMS* (R.D.HAMMESFAHR, edit.), Myrtle Beach, SC, 90-92.

GOLDEN,B., BODIN,L. DOYLE,T. y STEWART,W.Jr. (1.980). Approximate Traveling Salesman Algorithms. *Operations Researsch*, vol.28, nº 3, parte II, 694-711.

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

LAPORTE,G. (1.992). The Vehicle Routing Problem: An overview of exact y approximate algorithms. *European Journal of Operations Research*, 59, 345-358.
LIN,S. (1.965). Computer Solutions to the Traveling Salesman Problem. *Bell Syst.Tech.Jou.*, vol 44, 2245-2269.

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

NURMI,K. (1.991). Traveling Salesman Problem Tools for Microcomputers. *Computers & Ops.Res.* Vol. 18, nº 8, 741-749.

OR,I. (1.976). Traveling Salesman Type Combinatorial Problems y their Relations to the Logistics of Blood Banking. *Ph.Thesis, Dpt. of Industrial Enginnering y Management Sciences, Northwestern Univ.*

OSMAN,I.H. (1.993). Metastrategy Simulated Annealing y Tabu Search Algorithms for the Vehicle Routing Problem. *Annals of Operations Researsch*, vol. 41, pp. 421-451.

PACHECO,J. (1.994). 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 1.994.

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* celebrado en Albacete en Junio de 1.996.
SAVELSBERGH,M.W.P. (1.985). Local Search for Routing Problems with Time Windows. *Ann.Oper.Res.*, 4, 285-305.

SOLOMON,M., BAKER,E. y SCHAFFER,J. (1.988). Vehicle Routing y Scheduling Problems with Time Window Constraints: Efficient Implementations of Solution Improvement Procedures. In *Vehicle Routing: Methods y Studies*, (Studies in Management Sciences y Systems, vol.16), eds: GOLDEN,B.L. y ASSAD,A.A., Nort-Holland, 85-106.