Búsqueda Tabú en Problemas de Rutas de Vehículos con Demanda Compartida

J. M. Belenguer and M. C. Martínez
Departamento de Estadística e I.O., Universitat de València, Spain

Febrero de 2000

Resumen

La dificultad inherente a los Problemas de Rutas de Vehículos con Capacidades invita a desarrollar algoritmos capaces de generar eficientemente buenas soluciones posibles para los mismos. En este trabajo abordamos la resolución heurística de los problemas de rutas de vehículos en los que se admite la posibilidad de abastecer la demanda de un mismo cliente utilizando uno o más vehículos. Para ello definimos una serie de movimientos especiales que nos han permitido desarrollar un algoritmo de búsqueda tabú, el cuál, gracias a la incorporación de estructuras de memoria a corto y largo plazo, guía la búsqueda dentro de la región de soluciones posibles hacia soluciones casi óptimas. Proporcionamos resultados computacionales relativos a instancias aleatorias y clásicas.

Key words and sentences: split delivery, vehicle routing, tabu search, heurísticos.

1 Introducción

El problema que nos ocupa en este trabajo, conocido como *Split Delivery Vehicle Routing Problem*, en adelante $SDVRP$, permite modelizar aquellos problemas de distribución en los que una empresa debe satisfacer, de la forma más económica posible, la demanda de servicio de sus clientes, teniendo en cuenta que dispone de un número fijo de vehículos de capacidad limitada y que la demanda de cada cliente puede servirse utilizando uno o varios vehículos. Como los clientes y el depósito, lugar en el que se encuentran los vehículos, están geográficamente localizados en puntos distintos, uno de los problemas reales que surgen cuando se intenta atender las demandas de los clientes es decidir qué clientes servir con cada uno de los vehículos, y en qué orden, de forma que se minimicen los costes de distribución. Esto suele ser equivalente a diseñar las rutas minimizando la distancia total recorrida por la flota.

Los Problemas de Rutas de Vehículos han despertado gran interés dentro del campo de la Investigación Operativa por dos motivos fundamentales: su extenso campo de aplicación y la dificultad que conlleva su resolución (pertenecen a la clase NP-*hard*). Sin embargo, siempre se ha prestado especial interés a aquellas aplicaciones en las que se asume que la demanda de cada cliente debe ser completamente abastecida por un único vehículo ($CVRP$). Prueba de ello son la gran cantidad de trabajos que podemos encontrar en la literatura intentado resolver estos problemas, tanto desde el punto de vista exacto como heurístico. Véanse por ejemplo los trabajos de Christofides, 1985, Laporte y Nobert, 1987, Osman, 1993, o Campos y Mota, 2000 entre otros. Teniendo en cuenta lo que acabamos de decir podríamos preguntarnos ¿qué es lo que justifica la necesidad de considerar el $SDVRP$ como un caso aparte? La respuesta a esta pregunta es múltiple. Por un lado, permitir compartir la demanda garantiza la posibilidad de construir siempre una solución para cualquier instancia del problema usando el mínimo número de vehículos. Esta circunstancia no acontece cuando se trabaja con el $CVRP$, en donde averiguar el número de vehículos necesario para realizar el servicio implica resolver un problema NP-*hard* (ver Garey y Johnson, 1979) i.e., un Bin-Packing asociado a las demandas de los clientes y a cajas de capacidad igual a la capacidad del vehículo. En segundo lugar, compartir la demanda permite encontrar, para ciertas instancias, soluciones óptimas en las que la distancia total recorrida es menor que si se resuelve el mismo problema dentro del contexto del $CVRP$, incluso utilizando, como ya veremos, el mismo número de vehículos.

Justificada la importancia del problema podríamos pensar que desde un punto de vista teórico podríamos resolverlo como un caso particular de $CVRP$: bastaría con replicar cada cliente tantas veces como indique su demanda y resolver un $CVRP$ con demandas unitarias (ver Federgruen y Simchi-Levi, 1985). El principal inconveniente de este enfoque es, evidentemente, que no resulta demasiado práctico, puesto que la réplica de los clientes incrementa considerablemente el tamaño de las instancias, dificultando así su resolución.

A pesar de las ventajas mencionadas, el $SDVRP$ sigue siendo un problema NP-$hard$ (Dror y Trudeau, 1989) poco estudiado hasta el momento, lo que justifica el interés en diseñar buenos algoritmos heurísticos que lo resuelvan. En la revisión de la bibliografía sobre el tema que hemos llevado a cabo no hemos encontrado ningún intento de resolver el problema utilizando técnicas de búsqueda tabú. El inicio de la investigación en esta línea es lo que ha motivado el trabajo que presentamos.

Para terminar esta sección revisamos los trabajos sobre el $SDVRP$ que, hasta donde nosotros conocemos, han sido publicados hasta el momento. La primera alusión al problema que encontramos se debe a Schrage, 1981, aunque hasta años más tarde no se publicó el primer trabajo sobre el tema, precisamente un algoritmo heurístico de descenso propuesto por Dror y Trudeau, 1989 y 1990. Con posterioridad, Dror et al., 1994, presentan una cota inferior para el $SDVRP$ basada en una formulación del problema como un modelo de programación lineal entera. Se puede completar la revisión bibliográfica a la que nos hemos referido con los trabajos de Frizzell y Giffin, 1992 y 1995, el ya mencionado Federgruen y Simchi-Levi, 1985, Sierksman y Tijssen, 1998, Mullaseril et al., 1997, y por último, con el estudio poliédrico del problema realizado por Martínez, 1995, Martínez y Mota, 1999 y Martínez et al., 2000, y la cota inferior desarrollada por Belenguer et al., 2000.

2 El problema y un esquema algorítmico para resolverlo

El *Split Delivery Vehicle Routing Problem* consiste en, dada una flota de K vehículos idénticos de capacidad Q, localizados en un depósito, y un conjunto de n clientes con demandas d_i, $0 < d_i \leq Q$, diseñar una ruta para cada vehículo de forma que la distancia total recorrida sea lo menor posible y se cumplan las siguientes restricciones:

- cada cliente debe estar asignado a al menos una ruta,

- la demanda de cada cliente debe abastecerse completamente entre los vehículos que lo visitan,
- cada ruta debe partir del depósito y regresar a él visitando a al menos un cliente, y por último,
- la demanda total servida por un vehículo no puede ser superior a su capacidad.

Belenguer et al., 2000, formulan el $SDVRP$ sobre un grafo no dirigido en el que los vértices representan a los clientes y al depósito (vértice 0), y las aristas, a los caminos entre cada par de clientes o entre cada cliente y el depósito. Esto permite modelizar el problema de la siguiente manera, definiendo las variables:

- x_{ij}^h : número de veces que el vehículo h utiliza la arista (i,j), $\forall (i,j) \in E$;
- $y_i^h = 1$ si el vehículo h visita al cliente i y 0 en otro caso, $\forall\, i \in V \backslash \{0\}$;
- d_i^h : parte de la demanda del cliente i servida por el vehículo h, $\forall\, i \in V \backslash \{0\}$.

$$Min \quad \sum_{h=1}^{K} \sum_{(i,j)\in E} c_{ij}\ x_{ij}^h$$

$$
\begin{array}{lllr}
s.t.: & \sum_{i=1}^{n} x_{0i}^h \geq 2 & h=1,\dots,K, \ \text{y par} & (1)\\
& \sum_{(i,j)\in\delta(i)} x_{ij}^h \geq 2y_i^h & h=1,\dots,K, \ \ i=1,\dots,n, \ \text{y par} & (2)\\
& x_{ij}^h \leq M y_i^h & \left\{ \begin{array}{l} \forall (i,j) \in E \\ h=1,\dots,K, \ \ i=1,\dots,n \\ M \text{ constante suficientemente grande} \end{array} \right. & (3)\\
& \sum_{(i,j)\in(S,\bar{S})} x_{ij}^h \geq 2y_u^h & \left\{ \begin{array}{l} \forall S \subseteq V\backslash\{0\} \\ 2 \leq |S| \leq n-1 \\ \forall u \in S \\ h=1,\dots,K \end{array} \right. & (4)\\
& d_i^h \leq d_i y_i^h & h=1,\dots,K, \ \ i=1,\dots,n & (5)\\
& \sum_{h=1}^{K} d_i^h = d_i & i=1,\dots,n & (6)\\
& \sum_{i=1}^{n} d_i^h \leq Q & h=1,\dots,K & (7)\\
& x_{ij}^h \geq 0 \text{ y enteros} & h=1,\dots,K, \ \ \forall (i,j) \in E & (8)\\
& y_i^h \in \{0,1\} & h=1,\dots,K, \ \ i=1,\dots,n & (9)\\
& d_i^h \geq 0 \text{ y enteros} & h=1,\dots,K, \ \ i=1,\dots,n & (10)
\end{array}
$$

en donde $K = \lceil \frac{\sum_{i=1}^{n} d_i}{Q} \rceil$ i.e., el menor entero igual o superior a $\frac{\sum_{i=1}^{n} d_i}{Q}$, y c_{ij} el coste asociado a la distancia entre i y j.

(1) garantiza que todas las rutas incluyen al depósito, mientras que (2) asegura que se visitan todos los clientes; las restricciones (3) y (4) garantizan la conexión de las rutas y previenen la formación de subtours. Por último, (5), (6) y (7) obligan a que la demanda de todos los clientes sea servida respetando la capacidad de los vehículos.

En este trabajo proponemos un esquema de búsqueda tabú que proporciona buenas soluciones para el problema, entendiendo que una solución es mejor que otra cuando la distancia total recorrida por las rutas que la forman es menor. Como el número de vehículos utilizados es fijo, puesto que trabajamos siempre con $K = \lceil \frac{\sum_{i=1}^{n} d_i}{Q} \rceil$ (el menor número de vehículos con el que se puede abastecer toda la demanda), su valor no se considera al evaluar la calidad de las soluciones obtenidas. Sin embargo, si que tendrá interés a la hora de compararlo con el número de vehículos que sería necesario utilizar si no se compartiese la demanda.

Dada s una solución del problema, los elementos básicos para implementar un algoritmo de tales características, ver Glover, 1992, pasan por definir un conjunto de soluciones, $N(s)$, obtenidas a partir de s mediante ciertos movimientos previamente definidos. También es preciso establecer un criterio para evaluar las soluciones vecinas a una dada, para después elegir la mejor de entre las que cumplen ciertas restricciones. Estas restricciones tienen que ver con el siguiente hecho: en un esquema de búsqueda tabú la forma de evitar quedar atrapados en un óptimo local consiste en admitir movimientos que conducen a soluciones peores que la mejor encontrada hasta un determinado momento. Para evitar el ciclado al que podría conducirnos esta estrategia es necesario definir criterios que prohíban ciertos movimientos durante un determinado número de iteraciones, y otros que permitan decidir cuando un movimiento vuelve a ser admisible.

El algoritmo que proponemos sigue básicamente el esquema que presentamos a continuación:

1.- Elegir una solución inicial, s, del $SDVRP$. Definir s^* como la mejor solución encontrada hasta el momento. Hacer $s^* = s$ e $iter := 0$.

2.- Mientras no se cumpla el criterio de parada 1:

BUCLE TABÚ

  2.1 Mientras no se cumpla el criterio de parada 2:
  - $iter := iter + 1$.
  - Generar $N' \subseteq N(s)$.
  - Elegir la mejor solución s' de N' de acuerdo con la función objetivo que mide la distancia total recorrida por las rutas, $Dist(s)$.
  - $s := s'$.
  - Si $Dist(s') < Dist(s^*)$, entonces $s^* = s'$.

  2.2 *Diversificar la búsqueda.* Generar a partir de s^* una nueva solución actual, s, aplicando el criterio de diversificación.

3.- Reoptimizar individualmente las rutas con el algoritmo de Lin y Kernighan, 1973. Si el proceso de reoptimización conduce a una solución mejor, volver al paso 2, incrementando convenientemente el máximo de iteraciones del criterio de parada 1.

El primer criterio de parada consiste en abandonar la búsqueda cuando no es posible llevar a término la diversificación o cuando, aún siendo posible, se ha alcanzado un número máximo de iteraciones previamente establecido. El segundo criterio establece el abandono del bucle cuando $N' = \emptyset$, o cuando se lleva más de un cierto número de iteraciones consecutivas sin mejorar la mejor solución encontrada hasta el momento.

3 Elementos que definen la búsqueda tabú

3.1 Obtención de soluciones iniciales

Como dijimos en la introducción, una de las principales ventajas de admitir el servicio compartido reside en garantizar la existencia de soluciones que utilizan el menor número de vehículos posible, es decir, $K = \lceil \frac{\sum_{i=1}^{n} d_i}{Q} \rceil$. El procedimiento que presentamos a continuación genera, a partir de una ordenación arbitraria de los clientes, una solución del $SDVRP$ que cumple dicha propiedad.

Algoritmo Constructivo

1. Considerar una ordenación arbitraria de los clientes. Inicializar la primera ruta visitando el primer cliente de la ordenación.

2.- Mientras no se supere la capacidad del vehículo asociado a la ruta actual, completar la ruta visitando y sirviendo clientes según la ordenación establecida.

3.- Si el último cliente visitado ya está completamente servido, ir al paso 4, en otro caso ir al paso 5.

4.- Si todavía quedan vértices por servir, inicializar una nueva ruta comenzado con el siguiente cliente no servido de la ordenación e ir al paso 2. En otro caso STOP.

5.- Inicializar una nueva ruta visitando el último vértice de la ruta anterior y terminar de servir su demanda. Si todavía quedan vértices por servir, ir al paso 2, en otro caso STOP.

Observar que todos los vehículos, excepto quizás el último, sirven exactamente Q unidades de demanda y además, el número máximo de clientes con demanda compartida es $K - 1$. Este tipo de soluciones satisfacen uno de los criterios de optimalidad que deben cumplir las soluciones óptimas del $SDVRP$ i.e., la de no existir dos rutas distintas con más de un vértice en común (ver Dror y Trudeau, 1989). Sin embargo, es evidente que la calidad de las soluciones obtenidas no tiene porque ser buena y dependerá mucho de la calidad de la secuencia inicial.

En nuestro caso hemos considerado distintas soluciones iniciales que han sido obtenidas a partir de las siguientes ordenaciones:

- la natural dada por los clientes $\{1, 2, \ldots, n\}$,
- la obtenida a partir del tour del TSP resultante de aplicar el algoritmo de Lin y Kernighan, 1973, a la secuencia inicial,
- la que se obtiene cuando se visitan en sentido contrario, los clientes de la ordenación que proporciona la mejor solución inicial, $\{i_1, i_2, \ldots, i_n\}$, i.e., en el orden $\{i_n, i_{n-1}, \ldots, i_2, i_1\}$,
- las que se obtienen inicializando la ordenación usada en i_2, i_3, y así sucesivamente hasta i_n.

3.2 Definición de movimientos y evaluación del coste asociado

En primer lugar definiremos los movimientos que permiten generar el entorno, $N(s)$, asociado a una solución s dada. Dichos movimientos están basados en los λ-intercambios definidos por Osman, 1993, para el $CVRP$. Como se observará, compartir la demanda incrementa considerablemente el número de movimientos que, al menos en principio, es posible realizar a partir de una solución, y consecuentemente incrementará el esfuerzo computacional necesario para evaluarlos. Este es el motivo que nos ha llevado a considerar exclusivamente movimientos que involucran a uno o, a dos clientes como máximo, y nos referiremos a ellos como *0-1* o *1-1 intercambios*, respectivamente. En ocasiones también hablaremos de *movimientos 0-1* o de *movimientos 1-1.*

Dada una solución del $SDVRP$ y una ruta r_h, denotaremos por $cap(h)$ la capacidad libre de la ruta, esto es, $cap(h) = max\{0, Q - \sum_{i=1}^{n} d_i^h\}$.

0-1 intercambios

Un *0-1 intercambio* consiste en eliminar un cliente i (y por tanto su demanda, d_i^h) de una ruta r_h, e insertarlo en la mejor posición posible en otra ruta distinta, r_l. Dependiendo de si r_h servía total o parcialmente la demanda de i antes del intercambio, y de si r_l visitaba o no a i, el tipo de movimiento a realizar puede ser más o menos complejo.

Diremos que un *movimiento 0-1* que elimina a i de r_h y lo inserta en r_l es *admisible* si d_i^h no es mayor que $cap(l)$ antes del intercambio. Observar que, como se ilustra en el primero de los movimientos de la figura 1, después de un intercambio 0-1 un cliente con demanda compartida puede pasar a ser abastecido por un único vehículo.

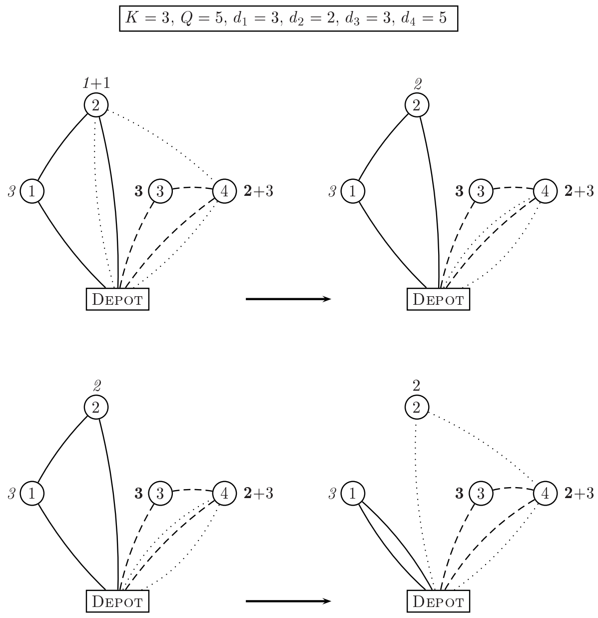

Figura 1: Intercambios 0-1

1 -1 intercambios

Los *intercambios 1-1* involucran a dos rutas distintas, r_h y r_l, y a dos clientes distintos, i y j, uno de cada una de ellas. Es evidente que la casuística a la que nos enfrentamos en esta ocasión es bastante más compleja que la que se da con los movimientos 0-1.

En un *movimiento 1-1* el vehículo asociado a la ruta r_h deja de servir δ_i unidades de demanda del cliente i para que las sirva el vehículo que realiza la ruta r_l. Análogamente, δ_j unidades de demanda del cliente j pasan de r_l a r_h. δ_i y δ_j se definen de la siguiente manera, si

$$\begin{array}{c} d_i^h \leq cap(l) + d_j^l \\ d_j^l \leq cap(h) + d_i^h \end{array} \quad \longrightarrow \quad \begin{array}{c} \delta_i := d_i^h \\ \delta_j := d_j^l \end{array}$$

en otro caso,

$$\delta_i = \delta_j := min\{d_i^h, d_j^l\}.$$

Si antes de realizar el movimiento los clientes i y j no pertenecían a las rutas de destino, r_l y r_h, respectivamente, al realizar el intercambio los insertamos en la mejor posición posible. En la figura 2 se ilustra un movimiento 1-1.

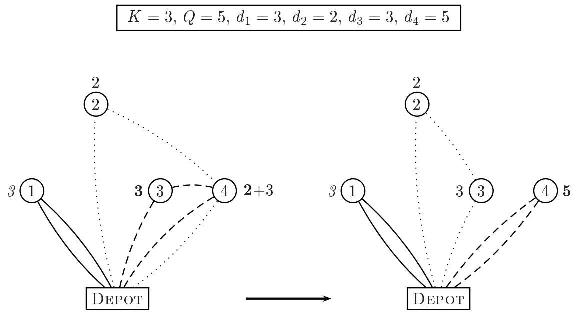

Figura 2: 1-1 intercambio de los clientes 3 y 4.

Evaluación del coste de un movimiento

Dado un cliente i y una ruta $r_h = \{i_0, i_1, i_2, \ldots, i_{n_h}\}$, en donde $i_0 = i_{n_h} = 0$, la siguiente función define el coste de inserción de i en r_h,

$$CostIns_i(h) = \begin{cases} 0 & \text{si } i \in r_h \\ min\{c_{i_j,i} + c_{i,i_{j+1}} - c_{i_j,i_{j+1}}, \;\; j = 0, \ldots, n_h - 1\} & \text{si } i \notin r_h \end{cases}$$

Si i_1 e i_2 son, respectivamente, el antecesor y el predecesor de un cliente i en una ruta r_h, el coste de eliminar a i de r_h se define de la forma habitual i.e.,

$$CostDel_i(h) = c_{i_1,i_2} - c_{i_1,i} - c_{i,i_2}$$

Como consecuencia, y teniendo en cuenta que la calidad de una solución del $SDVRP$ viene dada por la distancia total recorrida por todas las rutas, a la hora de evaluar un movimiento es necesario calcular la diferencia global de distancias entre las soluciones antes y después del intercambio. Si $Dist_s(r_h)$ indica la distancia recorrida por el vehículo h en s, el coste de un movimiento en el que intervienen r_h y r_l es:

$$(Dist_{s'}(r_h) - Dist_s(r_h)) + (Dist_{s'}(r_l) - Dist_s(r_l)),$$

en donde las distancias de las rutas en s' se calcularán teniendo en cuenta los costes de inserción y/o eliminación de los clientes implicados en el intercambio. Observar que un movimiento mejorará la solución actual si su coste es negativo.

La reoptimización individual de las rutas que se lleva a cabo en la última fase del algoritmo podría realizarse cada vez que cambiamos de solución. Sin embargo, hemos comprobado que tal acción no es eficiente desde el punto de vista computacional, y, como cabía esperar, tener buenas soluciones en cada una de las iteraciones del algoritmo no garantiza necesariamente que la mejor solución encontrada al final del proceso sea mejor. Este es uno de los motivos por los que a la hora de evaluar el coste de un movimiento solamente nos fijamos en la diferencia de las distancias recorridas por las dos rutas involucradas en el mismo, antes y después del intercambio, y no en la diferencia de distancias de las rutas reoptimizadas.

3.3 Exploración del entorno

En esta sección definimos lo que entendemos por *movimiento admisible*, describiremos también las estrategias utilizadas para la elección de movimientos, y la forma en la que conseguimos diversificar la búsqueda para entrar en regiones no exploradas. Por último, hablaremos de como intensificar la búsqueda dentro de zonas prometedoras. Concluimos así la descripción de los elementos que nos han permitido implementar el esquema algorítmico que hemos presentado en la Sección 2.

Movimientos admisibles y criterio de aspiración

La filosofía tabú escapa de los óptimos locales permitiendo realizar movimientos que conducen a soluciones peores que la que se tiene en un momento dado. Para evitar que un movimiento que no mejora nos devuelva a una solución ya visitada, prohibimos realizar intercambios que involucran a clientes que ya han intervenido en alguna de las $LongTabu$ últimas iteraciones del algoritmo. Por lo tanto, decimos que un movimiento es *admisible* cuando no está prohibido.

Sin embargo, parece poco recomendable dejar de realizar un movimiento que, aunque prohibido, nos conduzca a una solución mejor que la mejor solución encontrada, por lo que para evitar tales situaciones consideramos el siguiente *criterio de aspiración*:

> "un movimiento prohibido deja de serlo si al realizarlo se mejora la mejor solución encontrada hasta ese momento".

Estrategias de elección de movimientos

Dada una solución s del $SDVRP$ es posible aplicar diferentes estrategias para explorar su entorno, $N(s)$. Nosotros hemos optado por contemplar las dos posibilidades siguientes: explorar en primer lugar el subentorno definido por los intercambios 0-1, y solamente cuando no hay ningún movimiento 0-1 admisible exploramos el resto, es decir, el subentorno definido por los intercambios 1-1, o al contrario, primero nos centramos en los 1-1 y después en los 0-1.

Como ya henos comentado, compartir la demanda incrementa considerablemente el número de movimientos a evaluar en cada iteración del algoritmo, por lo que hemos decidido implementar el bucle tabú de forma que en el momento en que se detecta un intercambio de mejora (esto es, con coste asociado negativo), el movimiento se realiza. Solamente exploramos completamente $N(s)$ cuando no hay ningún movimiento conducente a una solución mejor que la actual y, en este caso, realizamos el menos malo de los intercambios no prohibidos.

Intensificación y diversificación

Después de un determinado número de iteraciones, o cuando no hay movimientos admisibles, abandonamos el bucle tabú e intentamos diversificar la búsqueda hacia nuevas regiones. Observar que las características de las soluciones iniciales, junto con los movimientos que hemos definido, conducen en muchas ocasiones a soluciones en las que ningún vértice es visitado por más de un vehículo. Esto significa que la búsqueda nos ha conducido a la región de soluciones posibles del $CVRP$ y, salvo que ideemos alguna forma de salir de ella, los intercambios que estamos considerando no nos van a permitir abandonarla. Esta situación no tendría que ser mala sino fuera porque el conjunto de soluciones del $SDVRP$ contiene estrictamente al conjunto de soluciones del $CVRP$. En el caso en que las soluciones atractivas para nuestro problema sean soluciones con vértices de demanda compartida, habremos caído en una zona en la que la búsqueda de buenas soluciones no tiene ningún futuro. Llegado este punto es necesario recurrir a la *diversificación* de la búsqueda, aunque también podemos aplicarla en situaciones distintas, como de hecho hacemos cada vez que abandonamos el bucle tabú.

La fase de diversificación se inicia recuperando la mejor solución conocida hasta el momento junto con su estructura de memoria a largo plazo, la cuál almacena el número de veces que cada cliente ha intervenido en algún intercambio a lo largo de todo el proceso. A continuación, se realiza un movimiento en el que se obliga a que dos clientes, de rutas distintas, y cuyas demandas son servidas por un único vehículo, compartan parte de sus respectivas demandas. Para ello seleccionamos, de entre los vértices que cumplen los requisitos anteriores, los dos que menos veces han intervenido en algún movimiento, digamos i y j, e intercambiamos una demanda de $\frac{1}{2}\min\{d_i, d_j\}$ unidades. A partir de la nueva solución se aplica de nuevo el paso 2.1 del algoritmo tabú descrito en la sección anterior, con la diferencia añadida de que prohibimos, durante un cierto número de iteraciones, realizar movimientos en los que intervengan alguno de los dos clientes que han dado lugar a la diversificación. El objetivo es evitar el regreso a la solución original. Además, cada vez que dos clientes participan en una diversificación, se penalizan convenientemente para evitar que vuelvan a ser elegidos en las diversificaciones inmediatamente posteriores.

Aunque no se ha implementando una fase de intensificación específica, disponemos de dos procedimientos que pueden entenderse como una intensificación en la búsqueda en el entorno de la mejor solución. En primer lugar, cuando abandonamos la fase de diversificación por haber superado un número máximo de iteraciones previamente fijado, intentamos mejorar la solución de manera que vértices con demanda compartida dejen de serlo realizando movimientos 0-1 con coste cero. A continuación, como dijimos en el punto 3 del esquema del algoritmo, se reoptimiza cada una de las rutas que componen la mejor solución encontrada hasta el momento, mediante el algoritmo de Lin y Kernighan.

4 Resultados Computacionales

Hemos implementado el algoritmo utilizando el lenguaje de programación C y la versión 5.0 del compilador Visual C++. Los resultados que presentamos se han obtenido probando el algoritmo sobre dos colecciones de problemas tests en un ordenador PC con un procesador Pentium II a 450 Mhz.

Las instancias de la primera colección corresponden a las instancias del $CVRP$ de la $TSPLIB$, Reinelt, 1991, en donde pueden consultarse los detalles relativos a las demandas de los clientes y a sus coordenadas. Las de la segunda colección las hemos generado aleatoriamente siguiendo las mismas pautas que Dror y Trudeau, 1989. Las coordenadas son las de los problemas *eil51*, *eil76* y *eil101*, respectivamente, la capacidad del vehículo es 160 en todas ellas, y las demandas se han generado aleatoriamente siguiendo una distribución uniforme en intervalos cuyos extremos dependen de la capacidad del vehículo. Para ambos conjuntos de instancias, los números que aparecen en los nombres de las instancias indican el número de vértices del problema, que como vemos varía de 22 a 101.

Los parámetros que hemos contemplado en el algoritmo son los siguientes:

$LongTabu$:	longitud de la lista tabú,
$DivTabu$:	número de iteraciones en las que se prohibe utilizar los vértices que han intervenido en una diversificación,
$estrategia$:	estrategia de exploración del entorno: 1 = primero los 0-1, después los 1-1 2 = primero los 1-1, después los 0-1,
$iter\_sin\_mejora$:	iteraciones acumuladas desde la última vez que se mejoró el valor de la función objetivo,

los cuáles han sido ajustados estudiando, para distintas combinaciones de los mismos, el comportamiento de las desviaciones, respecto de las mejores soluciones conocidas, de las soluciones alcanzadas con cada una de dichas combinaciones.

Antes de presentar los resultados obtenidos con el algoritmo vamos a describir brevemente cómo se han ajustado los parámetros del mismo.

En primer lugar fijamos los dos parámetros, $iter\_sin\_mejora = 300$ y $estrategia = 2$, y analizamos distintas combinaciones de los dos parámetros restantes, $DivTabu$ y $LongTabu$, obteniendo los siguientes resultados:

		$LongTabu$		
		5	7	10
	0	0.97	1.10	1.26
$DivTabu$	50	0.95	0.99	1.28
	100	0.97	1.04	1.35

Claramente se observa que, para valores fijos de $DivTabu$, los resultados mejoran a medida que disminuye la longitud de la lista tabú ($LongTabu$). En cambio, cuando se fija este parámetro, no parece que exista una pauta en el comportamiento de las desviaciones medias; solamente en el caso de $LongTabu = 10$ se observa que los resultados mejoran al disminuir el valor de $DivTabu$.

En las primeras experiencias computacionales se observó que la solución que se obtenía tras una diversificación podía llegar a ser mucho peor que la solución inicial del algoritmo, lo cuál obligaba a realizar un gran número de iteraciones antes de alcanzar una solución de un coste aceptable. Esto nos llevó a implementar una variante del algoritmo en la que la diversificación, para un par de vértices determinado, se realizaba únicamente cuando el coste de la solución a la que nos conducía era inferior a $cost\_ini + 0.1(cost\_ini - best\_cost)$, eligiéndose otra pareja de vértices, en otro caso. Los resultados fueron los siguientes:

		$LongTabu$		
		5	7	10
	0	1.23	1.04	1.22
$DivTabu$	50	1.16	1.20	1.38
	100	1.09	1.02	1.31

Sorprendentemente, los resultados obtenidos fueron ligeramente peores que los obtenidos con la diversificación propuesta originalmente, por lo que decidimos abandonar esta posibilidad.

Las pruebas realizadas parecían indicar que el mejor valor de $LongTabu$ era 5. Sin embargo, para dicho valor, diferentes valores de $DivTabu$ producían resultados similares. Decidimos entonces realizar un estudio más detallado incorporando al análisis el número de vértices de las instancias, n. Los resultados obtenidos revelaron, como se recoge en la siguiente tabla, que dependiendo del tamaño de la instancia conviene trabajar con distintas combinaciones de los parámetros $LongTabu$ y $DivTabu$, combinaciones éstas que decidimos mantener en el resto de experiencias computacionales.

	$n < 50$	$n = 51$	$n = 76$	$n = 101$
$LongTabu$	5	7	5	7
$DivTabu$	100	100	0	0

A continuación consideramos la influencia que podía tener en las soluciones alcanzadas la estrategia de exploración del entorno, observando que los mejores resultados se alcanzan cuando utilizamos la estrategia 1 y mantenemos el valor del parámetro $iter\_sin\_mejora = 200$.

		$iter\_sin\_mejora$		
		200	300	400
	1	0.77	0.95	0.91
$estrategia$	2	0.82	0.81	0.99

De nuevo, el análisis del comportamiento del algoritmo en función del tamaño de la instancia reveló que para instancias de tamaño 101, la estrategia 2 funcionaba mejor.

Los resultados computacionales que presentamos en las tablas 1 y 2 corresponden, por tanto, a los siguientes valores de los parámetros:

	$n < 50$	$n = 51$	$n = 76$	$n = 101$
$LongTabu$	5	7	5	7
$DivTabu$	100	100	0	0
$estrategia$	1	1	1	2

con $iter\_sin\_mejora = 200$ en todos los casos y estableciendo un número máximo de 10000 iteraciones en el criterio de parada 1.

Para dar una idea de la dificultad de las instancias consideradas, las primeras columnas muestran el número de vehículos de cada una de ellas, la capacidad de los mismos, y el ratio entre la demanda y la capacidad totales de los mismos. Además, con objeto de medir la calidad de las soluciones obtenidas, hemos comparado sus costes ($Cost$) con los de las mejores soluciones conocidas hasta el momento ($Best$), y con las cotas inferiores (LB) proporcionadas en Belenguer et al., 2000. En $Best$ presentamos el coste de la mejor solución encontrada durante las diferentes pruebas realizadas con nuestro algoritmo, excepto para la instancia S101D1 que fue obtenida con el metaheurístico que proponen Campos y Mota, 2000, para el $CVRP$.

Incluimos también dos columnas con los valores de los gap's asociados:

$$GAP_1 = \frac{Cost - Best}{Best} \times 100 \qquad GAP_2 = \frac{Cost - LB}{LB} \times 100$$

esto es, las desviaciones de la solución obtenida, respecto de la mejor solución conocida, y respecto de la cota inferior, respectivamente. Las últimas columnas muestran, respectivamente, el tiempo de CPU total del algoritmo y el número total de iteraciones, $iter$.

Name	K	Q	$Ratio$	$Cost$	$Best$	LB	GAP_1	GAP_2	CPU	$iter$
eil22	4	6000	.94	†375	†375	375.00	0.00	0.00	2.82	10171
eil23	3	4500	.75	†569	†569	569.00	0.00	0.00	2.48	10186
eil30	3	4500	.94	†510	†510	508.00	0.00	0.39	5.49	10194
eil33	4	8000	.92	†835	†835	833.00	0.00	0.24	7.36	10121
eil51	5	160	.97	532	†521	511.57	2.11	3.99	22.96	10154
eilA76	10	140	.97	837	825	782.70	1.45	6.94	59.49	10055
eilB76	14	100	.97	1008	1002	937.47	0.60	7.52	112.98	20103
eilC76	8	180	.95	737	734	706.01	0.41	4.39	112.21	20144
eilD76	7	220	.89	682	682	659.43	0.00	3.42	121.33	20061
eilA101	8	200	.91	820	817	793.48	0.37	3.34	281.76	20139
eilB101	14	112	.93	1068	1066	1005.85	0.19	6.18	248.04	20045

Tabla 1: Computational Results of the Tabu Algorithm on some $TSPLIB$ instances.
† solución óptima.

Obsérvese que en la tabla 2 hemos incluido una columna adicional (K') indicando el mínimo número de vehículos con el que se han encontrado soluciones cuando está prohibido servir la demanda de un mismo cliente utilizando más de un vehículo.

Estos resultados nos llevan a una desviación media de 0.70 respecto de la mejor solución conocida y de 4.38 respecto de la cota inferior.

Los tiempos de CPU, para instancias con el mismo número de vértices y que han efectuado un número similar de iteraciones, son muy parecidos, incrementándose éste proporcionalmente al número de iteraciones realizadas.

Name	K	K'	*Ratio*	*Cost*	*Best*	*LB*	GAP_1	GAP_2	*CPU*	*iter*
S51D1	3	3	.84	459	458	454.00	0.22	1.10	22.35	10067
S51D2	9	9	.98	708	706	676.63	0.28	4.64	70.14	30177
S51D3	15	15	.95	952	942	905.22	1.06	5.17	20.15	10075
S51D4	27	30	.99	1572	1551	1520.67	1.35	3.38	36.14	10010
S51D5	23	26	.99	1329	1329	1272.86	0.00	4.41	108.65	40074
S51D6	41	50	.98	2187	2171	2113.03	0.74	3.50	37.24	10194
S76D1	4	4	.96	603	594	584.87	1.52	3.10	83.76	10177
S76D2	15	16	.99	1097	1084	1020.32	1.20	7.52	70.19	10118
S76D3	23	23	.96	1452	1434	1346.29	1.26	7.85	133.96	20003
S76D4	37	40	.97	2102	2087	2011.64	0.72	4.49	74.48	10249
S101D1	5	5	.98	732	716	700.56	2.23	4.49	193.45	10033
S101D2	20	20	.96	1384	1376	1270.97	0.58	8.89	463.95	40053
S101D3	31	32	.98	1880	1869	1739.66	0.59	8.07	216.40	20028
S101D5	48	54	.99	2800	2786	2630.43	0.50	6.45	155.60	10005

Tabla 2: Computational Results of the Tabu Algorithm on some random instances

5 Conclusiones

En este artículo hemos presentado un algoritmo heurístico que incorpora las ideas de la búsqueda tabú para encontrar, rápidamente, buenas soluciones posibles para el $SDVRP$.

La desviación media de las soluciones alcanzadas, con respecto a las mejores soluciones conocidas, puede considerarse satisfactoria, ya que es de tan solo un 0.7% y no supera nunca el 2.23%.

No podemos decir lo mismo del gap respecto de la cota inferior, que es algo superior, ya que llega a ser de un 4.38% por término medio. Sin embargo, nos queda la duda de si este porcentaje depende de la calidad del heurístico o de la cota inferior, ya que son pocas las instancias en las que se conoce la solución óptima.

No obstante, creemos que la integración en un algoritmo de Branch and Cut de las cotas proporcionadas por el heurístico puede ayudar a despejar esta duda ya que cabe esperar que ello mejore la calidad de la cota inferior.

Referencias

[1] J. M. BELENGUER, M. C. MARTÍNEZ AND E. MOTA, "A Lower Bound for the Split Delivery Vehicle Routing Problem", to appear in *Operations Research* (2000).

[2] V. CAMPOS AND E. MOTA, "Heuristic Procedures for the Capacitated Vehicle Routing Problem", to appear in *Computational Optimization and Applications* (2000).

[3] N. CHRISTOFIDES, "Vehicle Routing", in: *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*, E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan and D. B. Shmoys eds, Wiley, Chichester, 431-448 (1985).

[4] N. CHRISTOFIDES AND S. EILON, "An Algorithm for the Vehicle-Dispatching Problems", *Operations Research Quarterly* **20**, 309-318 (1969).

[5] M. DROR, G. LAPORTE AND P. TRUDEAU, "Vehicle Routing with Split Deliveries", *Discrete Applied Mathematics* **50**, 239-254 (1994).

[6] M. DROR AND P. TRUDEAU, "Savings by Split Delivery Routing", *Transportation Science* **23**, 141-145 (1989).

[7] M. DROR AND P. TRUDEAU, "Split Delivery Routing", *Naval Research Logistics* **37**, 383-402 (1990).

[8] A. FEDERGRUEN AND D. SIMCHI-LEVI, "Analysis of Vehicle Routing and Inventory-Routing Problems", in: *Network Routing*, M. O. Ball, T. L. Magnanti, C.L. Monma and G. L. Nemhauser eds. Handbooks in OR&MS, 8. North-Holland, Amsterdam, 297-373 (1995).

[9] P. W. FRIZZELL AND J. W. GIFFIN, "The Bounded Split Delivery Vehicle Routing Problem with Grid Networks Distances", *Asia-Pacific Journal of Operational Research* **9**, 101-116 (1992).

[10] P. W. FRIZZELL AND J. W. GIFFIN, "The Split Delivery Vehicle Scheduling Problem with Time Windows and Grid Network Distance", *Computers & Operations Research* **22**, 655-667 (1995).

[11] M. R. GAREY AND D. S. JOHNSON, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman, San Francisco (1979).

[12] F. GLOVER, "A User's Guide to Tabu Search", *Annals of Operations Research* **41**, 3-28 (1993).

[13] G. LAPORTE AND Y. NOBERT, "Exact Algorithms for the Vehicle Routing Problem", *Annals of Discrete Mathematics* **31**, 147-184 (1987).

[14] S. LIN AND B. W. KERNIGHAN, "An Effective Heuristic Algorithm for the Traveling Salesman Problem", *Operations Research* **21**, 498-516 (1973).

[15] M. C. MARTÍNEZ, "Rutas de Vehículos con Demanda Compartida", Ph.D. Thesis, University of València (1995).

[16] M. C. MARTÍNEZ AND E. MOTA, "Del Poliedro del Agente Viajero Gráfico al de Rutas de Vehículos con Demanda Compartida", Technical Report Nr. 1-99, Dpto. Estadística e Investigación Operativa, University of Valencia (1999).

[17] M. C. MARTÍNEZ, E. MOTA AND G. RINALDI, "The Split Delivery Vehicle Routing Polyhedron: new families of facet defining inequalities", submited (2000).

[18] P. A. MULLASERIL, M. DROR, AND J. LEUNG, "Split-Delivery Routeing Heuristics in Livestock Feed Distribution", *Journal of the Operational Research Society* **48**, 107-116 (1997).

[19] I. H. OSMAN, “Metastrategy Simulated Annealing and Tabu Search Algorithms for the Vehicle Routing Problem”, *Annals of Operations Research* **41**, 421-451 (1993).

[20] G. REINELT, “TSPLIB - A Traveling Salesman Problem Library”, *ORSA Journal of Computing* **3**, 376-384 (1991). WWW-access:
http://www.iwr.uni-heidelberg.de/iwr/comopt/soft/TSPLIB/TSPLIB.html

[21] L. SCHRAGE, “Formulation and Structure of More Complex/Realistic Routing and Scheduling Problems”, *Networks* **11**, 229-232 (1981).

[22] G. SIERKSMA AND G. A. TIJSSEN, “Routing Helicopters for Crew Exchanges on Off-shore Locations”, *Annals of Operations Research* **76**, 261-286 (1998).