Búsqueda tabú para el ruteo de vehículos

Juan Rodrigo Jaramillo Posada
Centro de Manejo Logístico Avanzado, Universidad del Estado en Albany. Estados Unidos
Correo electrónico: juan.jaramillo@asurams.edu

Recibido: 24/4/2012 / Aprobado: 6/6/2012

Resumen: El diseño de rutas eficientes para vehículos comerciales es de vital importancia en los sectores de transporte y logística. El ruteo de vehículos pertenece a la familia de problemas NP-Difícil, lo que obliga al diseño de algoritmos heurísticos para su solución. El presente estudio introduce una novedosa versión de la búsqueda tabú que hace uso de una lista tridimensional y aplica penalizaciones con incremento lineal a soluciones no viables durante la búsqueda. El algoritmo fue evaluado utilizando un reconocido conjunto de casos, y presentó buenos resultados.

Palabras clave: Ruteo de vehículos / optimización combinatoria / métodos heurísticos / búsqueda tabú

A novel tabu search algorithm for the capacitated vehicle routing problem

Abstract: The design of efficient routes for commercial vehicles is critical in the logistics and transportation sectors. The Vehicle Routing Problem is NP-Hard. Consequently, approximation algorithms are required to solve real-life size problem instances. This work introduces a tabu search algorithm that uses a tridimensional tabu list and a linear increasing penalty for handling infeasible solutions. The algorithm was tested using a well known set of problem instances, showing strong and encouraging results.

Key words: Vehicle routing problem / Combinatorial optimization / Heuristic methods / Tabu search

1. INTRODUCCIÓN

El diseño de rutas eficientes para vehículos comerciales es de vital importancia en los sectores de transporte y logística. Toda compañía que controle una flota de vehículos para servir a sus clientes se enfrenta con el problema del diseño de rutas con relativa frecuencia. En la práctica, las rutas se diseñan con distintos objetivos en mente. El objetivo más común es la minimización de la distancia recorrida por los vehículos. Un segundo objetivo es la minimización del tiempo utilizado por los vehículos. En la práctica se asume que distancia y tiempo están correlacionados y la optimización de uno lleva a la optimización del otro. Debido a su importancia, el ruteo de vehículos es uno de los problemas más estudiados en la literatura especializada. Máxime cuando es un problema táctico que se debe resolver con demasiada frecuencia.

El problema fue definido inicialmente por Dantzig y Ramser (1959) y es el resultado de la combinación de dos problemas en el área de la optimización combinatoria, el problema de empaquetamiento (*bin packing problem*) y el problema del agente viajero (*travelling salesman problem*). El problema de empaquetamiento asigna los clientes a cada vehículo cuidando de no exceder la capacidad de carga, mientras que el problema del agente viajero diseña las rutas de cada vehículo. Garey y Johnson (1979) demostraron que el problema de ruteo de vehículos pertenece al grupo de NP-Difícil. Esto último implica que solamente casos limitados (menos de 4 vehículos y menos de 20 clientes) pueden ser resueltos óptimamente utilizando niveles de tiempo computacional aceptables. Por esa razón, el desarrollo de algoritmos heurísticos con capacidad de generar soluciones de calidad en tiempo aceptable se hace prioritario.

Existe una gran variedad de versiones del problema de ruteo de vehículos debido a las diferentes funciones objetivo y a las diversas restricciones incorporadas a este. La versión más popular es la que considera la capacidad de los vehículos que se van a utilizar. En esta versión los destinos son visitados solamente una vez con el objeto de entregar o recoger mercancía y la función objetivo es la minimización de la distancia recorrida por los vehículos. Una segunda versión incluye la opción de recoger y entregar mercancía al mismo tiempo. Otra versión incluye restricciones en los tiempos de entrega de mercancía. Recientemente, Jaramillo (2010, 2011) presentó una versión del problema cuya función objetivo es minimizar las emisiones de CO2. Marinakis y Migdalas (2007) y Parragh et al. (2008) ofrecen una revisión muy completa de las

diferentes versiones del problema de ruteo de vehículos y las técnicas más utilizadas para resolverlas. En la segunda sección de este artículo se presenta una descripción detallada del problema incluyendo un modelo matemático y un ejemplo ilustrativo.

Debido a la dificultad computacional del problema, algoritmos heurísticos y meta-heurísticos son necesarios para resolver versiones comerciales del problema. Entre los algoritmos metaheurísticos se destaca la búsqueda tabú propuesta por Glover (1986). La búsqueda tabú es uno de los algoritmos más utilizados en optimización combinatoria debido a su efectividad y el bajo número de parámetros a tener en cuenta. El presente trabajo expone una novedosa versión de la búsqueda tabú que se caracteriza por su efectividad y facilidad para implementarse. El algoritmo propuesto se basa en una lista tabú tridimensional y en un mecanismo para penalizar soluciones inviables en la búsqueda. En la tercera sección de este trabajo se describe en forma detallada la búsqueda tabú. En la cuarta sección se discuten los resultados obtenidos al evaluar el algoritmo utilizando un reconocido grupo de problemas de la literatura especializada. Finalmente, en la quinta sección se presentan las conclusiones y las áreas de interés para futuras investigaciones.

2. DESCRIPCIÓN DEL PROBLEMA

2.1 Modelo matemático

El ruteo de vehículos considerado en este trabajo contempla un grupo de destinos (clientes) a los cuales se les debe entregar mercancía utilizando un grupo de vehículos con capacidad de carga limitada. En particular, el diseño de rutas eficientes para vehículos con capacidad limitada consiste en asignar destinos (clientes) a cada uno de los vehículos sin sobrepasar su capacidad de carga y diseñar cada una de las rutas de forma que la distancia recorrida por todos los vehículos sea la mínima posible. La formulación del problema que se presenta a continuación se basa en Kara et al. (2004).

Índices:

i, j Destinos $i, j = 1,\ldots, L$; el destino 1 representa el sitio de partida/llegada para todos los vehículos (base), los demás sitios representan las ubicaciones de los clientes. L es el número total de destinos a visitar.

Parámetros:

d_{ij} Distancia entre los destinos i y j.

q_i Demanda del cliente ubicado en el destino i en unidades de masa.

Q Capacidad de cada vehículo en unidades de masa.

K Número de vehículos disponibles.

Variables:

x_{ij} 1 si un vehículo visita el destino j inmediatamente después de visitar el destino i.
0 en caso contrario.

u_i Variable arbitraria real.

Función objetivo:

$$\min \sum_{i=1}^{L} \sum_{j=1}^{L} d_{ij} x_j \tag{1}$$

Restricciones:

$$\sum_{j=2}^{L} x_{1j} = K \tag{2}$$

$$\sum_{i=2}^{L} x_{i1} = K \tag{3}$$

$$\sum_{i=1}^{L} x_{ij} = 1 \qquad j = 2,...,L; i \neq j \tag{4}$$

$$\sum_{j=1}^{L} x_{ij} = 1 \qquad i = 2,...,L; j \neq i \tag{5}$$

$$Q - \left(Q - \max_{j \neq i} \{q_j\} - q_i \right) x_{1i} - \sum_{j=2}^{L} q_j x_{ij} \geq u_i \qquad i = 2,...,L \tag{6}$$

$$q_i \leq u_i \leq Q \tag{7}$$

$$x_{ij} \in [0,1] \quad (8)$$

$$u_i \geq 0 \quad (9)$$

La función objetivo (1) minimiza la distancia recorrida por todos los vehículos en conjunto. El grupo de restricciones (2) asegura que exactamente *K* rutas (una para cada vehículo) partan de la base (destino 1). El conjunto de restricciones (3) asegura que cada una de las rutas termine en la base. Las restricciones definidas en (4) controlan que solamente un vehículo llegue a cada destino. De forma similar, el grupo de restricciones (5) asegura que solamente parta un vehículo de cada destino. Los grupos de restricciones (6) y (7) aseguran que la capacidad máxima de los vehículos no sea sobrepasada y que no se generen subrutas; es una ruta que no incluye la base (destino 1). El grupo de restricciones (6) es una extensión del grupo Miller-Tucker-Zemlin para eliminar ciclos internos (ciclos que no incluyen el sitio de salida/llegada de los vehículos) propuesto en Kara et al. (2004). Finalmente, los grupos de restricciones presentados en (8) y (9) definen la naturaleza de las variables.

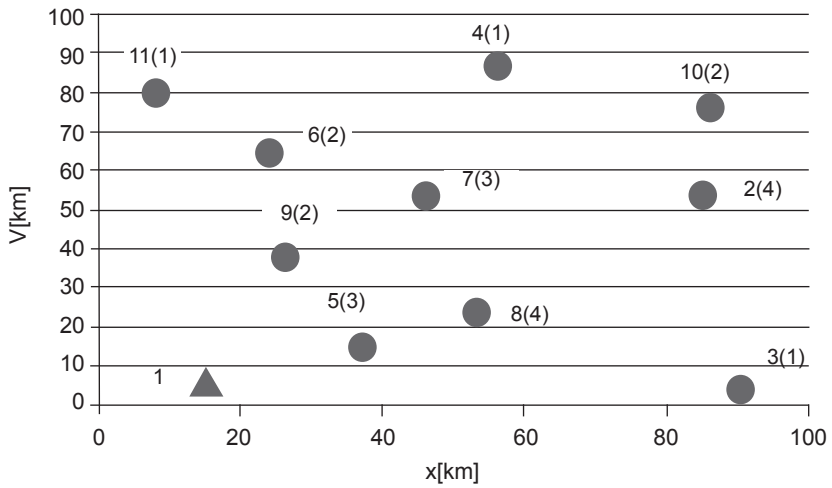
2.2 Caso ilustrativo

El caso presentado en esta sección tiene por objeto ilustrar el problema utilizando una compañía que debe entregar pedidos a 10 clientes utilizando 2 vehículos, cada uno con una capacidad de 12 toneladas. La tabla 1 contiene la información necesaria para resolver el caso. La primeras tres columnas contienen la identificación y las coordenadas en kilómetros que definen la ubicación de cada uno de los sitios a visitar. La cuarta columna (q_i) contiene el peso del pedido para cada cliente. Las siguientes 11 columnas muestran la distancia entre los diferentes destinos en kilómetros. Las distancias se obtuvieron a partir de las coordenadas y se redondearon. La figura 1 ilustra la ubicación de cada uno de los sitios y su respectiva demanda entre paréntesis. El sitio de partida y llegada de los vehículos es representado por un triángulo.

Tabla 1
Información del caso ilustrativo

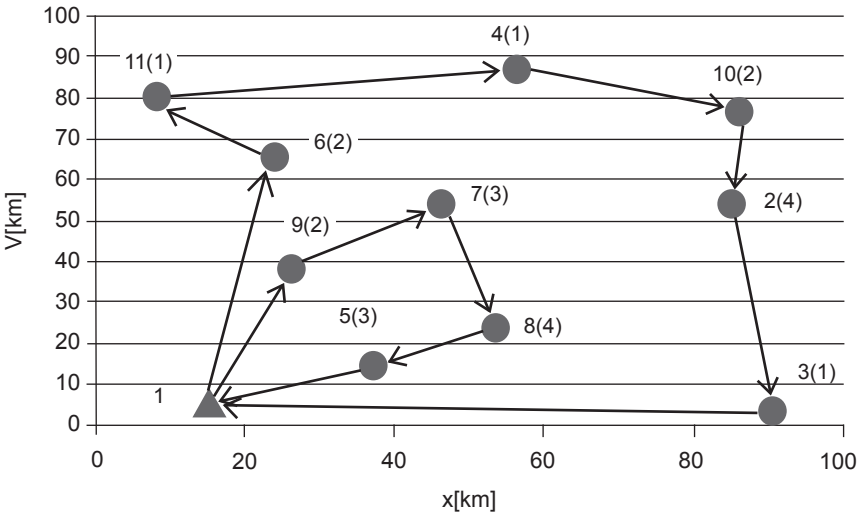
L	Coordenadas		q_j	d_{ij}										
	x	***y***		**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
1	15	6	0	0	84	75	90	23	59	57	42	33	100	74
2	85	54	2	84	0	50	43	61	61	39	43	61	23	81
3	90	4	1	75	50	0	89	54	89	66	42	72	73	111
4	56	87	3	90	43	89	0	74	38	34	63	57	31	48
5	37	15	3	23	61	54	74	0	51	40	18	25	79	71
6	24	65	2	59	61	89	38	51	0	24	50	27	63	21
7	46	54	3	57	39	66	34	40	24	0	30	25	46	46
8	53	24	4	42	43	42	63	18	50	30	0	30	62	71
9	26	38	2	33	61	72	57	25	27	25	30	0	71	45
10	86	77	2	100	23	73	31	79	63	46	62	71	0	78
11	8	80	1	74	81	111	48	71	21	46	71	45	78	0

Figura 1
Ubicación de los sitios a visitar



La figura 2 muestra la solución óptima del ejemplo. Dicha solución se obtuvo resolviendo el modelo matemático presentado en la sección 2.1. El modelo se programó en OPL y se resolvió utilizando la versión académica de IBM ILOG CPLEX Optimization Studio 12.2. El valor de la función objetivo de la solución óptima es de 431 kilómetros.

Figura 2
Solución óptima



Elaboración propia.

3. BÚSQUEDA TABÚ

Como se mencionó, debido a la dificultad para encontrar rutas óptimas para casos con un número significativo de destinos, es necesario el diseño de algoritmos heurísticos que permitan la construcción de rutas eficientes en tiempos aceptables. En su mayoría, los algoritmos heurísticos tienen los siguientes componentes: representación de la solución, búsqueda local y un mecanismo que permita escapar de óptimos locales. En las siguientes líneas se describen los componentes de la búsqueda tabú.

3.1 Representación de la solución

El algoritmo comienza con la generación de una solución inicial. La primera solución se crea utilizando un algoritmo constructor. El algoritmo constructor utilizado en este caso consiste en asignar los sitios de forma aleatoria para cada vehículo. Por ejemplo, una solución para el caso ilustrado en la sección anterior puede ser asignar los sitios 6, 7, 4, 2, y 3 al primer vehículo y los sitios 9, 11, 10, 8 y 5 al segundo vehículo, en ese orden. En general, las soluciones se pueden representar utilizando un conjunto de vectores $\boldsymbol{S}$, donde cada vector representa una ruta ($\boldsymbol{S} = \{[s_1, s_{i'}, \ldots s_j, s_1]_1\ [s_1, s_{n.}, \ldots, s_m, s_1]_2 \ldots\ [s_1, s_o, \ldots, s_p, s_1]_K\}$). Cada ruta comienza y termina en el destino 1, el cual es la base de los vehículos. La solución generada con el algoritmo constructor se puede representar como $\boldsymbol{S_0} = \{[1, 6, 7, 4, 2, 3, 1]_1\ [1, 9, 11, 10, 8, 5, 1]_2\}$. Como todas las rutas comienzan y terminan en el sitio 1, el sitio 1 se puede remover con el fin de hacer la representación de la solución más amigable, obteniéndose $\boldsymbol{S_0} = \{[6, 7, 4, 2, 3]_1\ [9, 11, 10, 8, 5]_2\}$. Una vez creada la primera solución, el siguiente paso consiste en calcular la carga inicial y la distancia recorrida por cada vehículo. La distancia total recorrida por el vehículo 1 es de 285 kilómetros (59 + 24 + 34 + 43 + 50 + 75) y la carga inicial es de 11 toneladas (2 + 3 + 3 + 2 +1). De forma similar, la distancia recorrida por el vehículo 2 es de 259 kilómetros, con una carga inicial de 12 toneladas. La función objetivo es la distancia total recorrida por los 2 vehículos, la cual es 544 kilómetros.

Es importante mencionar que el algoritmo constructor aleatorio puede generar soluciones que no son viables (soluciones en las que uno o más vehículos son sobrecargados). Sin embargo, las soluciones no viables son aceptables como punto de inicio o como soluciones intermedias en la búsqueda. En efecto, la experiencia ha mostrado que los algoritmos que vetan soluciones no viables durante el proceso de búsqueda son menos efectivos. Normalmente, la función objetivo de una solución no viable es penalizada proporcionalmente a su inviabilidad.

3.2 Búsqueda local

La búsqueda local es el mecanismo que permite explorar otras soluciones de forma controlada. Los algoritmos de búsqueda local evalúan las soluciones en el vecindario de la solución inicial. El vecindario se define como el grupo de soluciones que se pueden obtener como consecuencia de pequeños cambios (movidas) en la solución inicial. La búsqueda lo-

cal utilizada en este caso emplea 2 diferentes movidas denominadas 1^{opt} y 2^{opt}. 1^{opt} selecciona cada destino en la ruta y lo inserta en otra parte de la misma ruta o de una ruta diferente.

Un ejemplo de 1^{opt} es remover el destino 10 de la ruta del vehículo 2 y asignarlo a la ruta del vehículo 1 entre los sitios 4 y 2. La nueva solución es $\boldsymbol{S_1}$ = {[6, 7, 4, **10**, 2, 3]$_1$ [9, 11, 8, 5]$_2$}. El nuevo valor de la función objetivo es 536 kilómetros y los pesos iniciales de los vehículos son 13 y 10 toneladas, respectivamente. La nueva solución no es viable, pues el peso del vehículo 1 excede su capacidad máxima (12 toneladas); sin embargo, dicha solución es aceptable como solución intermedia. La movida tipo 2^{opt} intercambia dos sitios de manera simultánea en la misma ruta o en rutas diferentes. Por ejemplo, el intercambio de los sitios 7 y 11 en $\boldsymbol{S_1}$ genera la solución $\boldsymbol{S_2}$ = {[6, **11**, 4, 10, 2, 3]$_1$ [9, **7**, 8, 5]$_2$} con una función objetivo de 436 millas y con cargas iniciales de 11 y 12 toneladas, respectivamente, la cual es la solución óptima del caso utilizado como ejemplo. Es importante resaltar que la solución óptima se obtuvo a partir de una solución inviable.

En términos generales, cada iteración de la búsqueda local evalúa todas las soluciones posibles del vecindario y selecciona la que genera el menor valor en la función objetivo. La solución seleccionada se convierte en la nueva solución $\boldsymbol{S_0}$. La búsqueda local utilizada por el algoritmo se denomina 1^{opt}, 2^{opt} descendente. Es descendente porque selecciona la solución con la función objetivo de menor valor en el vecindario. El proceso se repite hasta que la búsqueda local no encuentra una solución que supere la existente al término de una iteración. Cuando eso pasa, se dice que la búsqueda local está atrapada en óptimo local.

3.3 Mecanismos de la búsqueda tabú

La búsqueda tabú le agrega diferentes mecanismos a la búsqueda local para permitirle escapar de los óptimos locales, evitar la repetición de soluciones evaluadas previamente, diversificar la búsqueda de nuevas soluciones y utilizar soluciones no viables como pasos intermedios.

El mecanismo central en la búsqueda tabú es la lista tabú. La lista tabú prohíbe la repetición de ciertas movidas por un número determinado de iteraciones para evitar la repetición de soluciones ya evaluadas. La novedosa lista tabú tridimensional utilizada en el presente algoritmo prohíbe la inserción de un destino entre 2 destinos durante cierto número de iteraciones (*tenor*). Por ejemplo, para la movida 1^{opt}

que permite generar S_1 a partir de S_0 la lista tabú prohíbe que el sitio 10 se inserte entre los sitios 11 y 8 en las siguientes *tenor* iteraciones. Al implementarse la lista tabú, la búsqueda local solo considera las movidas que no sean tabú. El valor de *tenor* es muy importante en la eficacia del algoritmo. Por un lado, valores muy bajos llevan al algoritmo a caer en ciclos, repitiendo la misma secuencia de soluciones. De otro lado, valores muy altos limitan la habilidad de la búsqueda al restringir demasiado las soluciones aceptables.

Un segundo mecanismo se aplica cuando una movida tabú genera una solución que supera la mejor solución encontrada hasta el momento. El mecanismo desestima la lista y la movida es aceptada. En ese caso, la acción de aceptar una movida tabú se denomina acción de aspiración.

Como se mencionó con anterioridad, el uso de soluciones no viables es importante en el proceso de búsqueda. La búsqueda tabú presentada en este algoritmo incluye un mecanismo para penalizar soluciones no viables. El mecanismo penaliza la función objetivo de las soluciones no viables con base en el exceso de capacidad de los vehículos. Es decir, al valor de la función objetivo se le adicionan el exceso de carga de los vehículos multiplicado por un factor predefinido. Al principio de la búsqueda el valor del factor para penalizar es pequeño, para permitir la diversidad en la búsqueda. El valor de la penalización se incrementa de forma lineal a medida que la búsqueda avanza. Es importante anotar que las penalizaciones altas al comienzo de la búsqueda limitan la diversidad de soluciones del algoritmo y las penalizaciones muy bajas llevan al algoritmo a estancarse en soluciones no viables.

El último parámetro del algoritmo es el encargado de finalizar la búsqueda y se denomina criterio de finalización. En este caso el criterio de finalización se definió como cierto número de iteraciones en los cuales no se obtiene una solución que supere la mejor encontrada hasta el momento. Finalmente, como la solución inicial de la búsqueda tabú es generada de forma aleatoria, el algoritmo se debe ejecutar varias veces.

4. RESULTADOS

La evaluación del algoritmo se hizo utilizando el conjunto de problemas creado por Augerat et al., denominado "Augerat set A". La ventaja de utilizar este grupo de problemas es que las soluciones óptimas se cono-

cen y permiten medir el desempeño del algoritmo. Los problemas se encuentran disponibles para descarga en el sitio web <http://www.coin-or.org/SYMPHONY/branchandcut/VRP/data/index.htm.old>. El conjunto consta de 26 problemas, el más pequeño con 32 destinos y 5 vehículos y el más complejo con 82 destinos y 10 vehículos. Para la programación del algoritmo se utilizó Visual C++ 2007 y la evaluación se hizo en un equipo con un procesador i5 de 2.4 GHz, con 8 GB de memoria y Windows 7 de 64 bits.

Los parámetros de la búsqueda tabú se sintonizaron por medio de la experimentación. En el diseño de experimentos se consideraron 4 niveles para el *tenor* y 3 niveles para el incremento en el factor de penalización para soluciones no viables. Los valores de los parámetros se definieron en función del número de destinos asociados a cada caso. Los niveles considerados para el *tenor* se definieron como 1**Destinos*, 5**Destinos*, 10**Destinos* y 20**Destinos*. Los resultados obtenidos sugieren que el mejor valor para el tenor es 5**Destinos*. Con valores menores, o mayores, la calidad de las soluciones se deteriora. Como se mencionó anteriormente, los tenores bajos permiten la generación de ciclos y los valores altos restringen la búsqueda.

El segundo parámetro a considerar es el factor utilizado para penalizar las soluciones. El factor inicial es de 1. Es decir, que por cada tonelada de exceso en cada vehículo, la función objetivo se penaliza agregándole una unidad de distancia. El factor se incrementa ligeramente en cada iteración. Esto permite la selección de soluciones no viables al principio de la búsqueda. El aumento del factor obliga al algoritmo a concentrar la búsqueda en soluciones viables a medida que la búsqueda avanza. El diseño de experimentos consideró 3 niveles: 0.00125/*Destinos*, 0.0025/*Destinos* y 0.0050/*Destinos*. El algoritmo demostró ser muy sensible a este parámetro. Valores ligeramente menores estancan el algoritmo en soluciones inviables y valores superiores generan resultados de baja calidad.

El tercer criterio a definir es el que finaliza la búsqueda. Los resultados del experimento indican que la búsqueda se debe suspender cuando se alcanzan 250**Destinos* iteraciones sin hallar una solución que supere la mejor encontrada hasta el momento. Finalmente, es importante anotar que las soluciones iniciales se generaron con el algoritmo constructor aleatorio. Por lo tanto, para cada combinación (penalización – *tenor*) se generaron 10 soluciones como punto de partida para la búsqueda.

La tabla 2 resume los resultados de la evaluación del algoritmo. La primera columna contiene la identificación de cada caso. El valor después de la *n* indica el número de destinos y el valor después de la *k* indica el número de vehículos. Por ejemplo, el caso *A-n*38-*k*5 tiene 38 destinos que deben ser visitados con 5 vehículos. La segunda columna indica las unidades de carga a distribuir. La tercera columna contiene la relación entre la carga a distribuir (*CT*) y la capacidad total disponible (cada vehículo tiene una capacidad de 100 unidades de carga, *C*). Es importante anotar que la dificultad de los casos es función del número de destinos, el número de vehículos y el valor de la relación carga/capacidad. A mayor número de destinos y de vehículos, mayor el número de soluciones posibles y por lo tanto mayor la dificultad del caso. De manera similar, entre más cercano a 1 es el valor de la relación carga/capacidad, mayor es la dificultad en la asignación de destinos (el problema de empaquetamiento se hace más difícil). La cuarta columna contiene la función objetivo de la solución óptima para cada caso. Es importante mencionar que resolver cada caso óptimamente requiere varios miles de horas de computación en equipos avanzados. Las siguientes columnas presentan un resumen de los resultados obtenidos para los 3 niveles de penalización: 0.00125/*Destinos*, 0.0025/*Destinos* y 0.0050/*Destinos*, utilizando un tenor de 5**Destinos*. Cada nivel de penalización cuenta con 3 columnas. La primera columna presenta la mejor función objetivo encontrada por la búsqueda tabú en los 10 intentos; la segunda columna contiene el número de veces que la búsqueda encontró dicho valor, y la tercera columna es el promedio obtenido en los 10 intentos del algoritmo en cada caso.

Los resultados muestran que la búsqueda tabú con tenor de 5**Destinos* y un incremento de 0.00125/*Destinos*, encontró la solución óptima para todos los casos y adicionalmente encontró la solución óptima el mayor número de veces (199 veces). Los otros 2 niveles encontraron la mejor solución a todos los problemas con excepción del caso *A-n61-k9*. De forma similar, la búsqueda tabú con 0.0025/*Destinos* encontró la mejor solución 189 veces y el algoritmo con 0.0050/*Destinos* encontró la solución óptima 187 veces. Es importante anotar que la efectividad del algoritmo se afecta de forma negativa con el incremento en la complejidad de los casos (número de veces que se encuentra la solución óptima). Finalmente, cuando el algoritmo no encontró la solución óptima, la solución final encontrada es muy cercana al valor óptimo, menos del 1% de diferencia.

Tabla 2
Resultados experimentales

Problema	Carga total	CT/C	Solución óptima	0.00125/Destinos			0.0025/Destinos			0.0050/Destinos		
				Mejor	Veces	Promedio	Mejor	Veces	Promedio	Mejor	Veces	Promedio
A-n32-k5	410	0.820	784	784	10	784.0	784	10	784.0	784	10	784.0
A-n33-k5	446	0.892	661	661	10	661.0	661	10	661.0	661	10	661.0
A-n33-k6	541	0.902	742	742	10	742.0	742	10	742.0	742	10	742.0
A-n34-k5	460	0.920	778	778	8	779.6	778	7	780.4	778	9	778.8
A-n36-k5	442	0.884	799	799	10	799.0	799	10	799.0	799	10	799.0
A-n37-k5	407	0.814	669	669	10	669.0	669	10	669.0	669	10	669.0
A-n37-k6	570	0.950	949	949	10	949.0	949	10	949.0	949	10	949.0
A-n38-k5	481	0.962	730	730	10	730.0	730	10	730.0	730	10	730.0
A-n39-k5	475	0.950	822	822	10	822.0	822	8	822.2	822	8	822.2
A-n39-k6	526	0.877	831	831	10	831.0	831	10	831.0	831	10	831.0
A-n44-k6	570	0.950	937	937	10	937.0	937	10	937.0	937	10	937.0
A-n45-k6	593	0.988	944	944	2	951.6	944	4	956.4	944	3	956.2
A-n45-k7	634	0.906	1146	1146	10	1146.0	1146	10	1146.0	1146	10	1146.0
A-n46-k7	603	0.861	914	914	10	914.0	914	10	914.0	914	10	914.0
A-n48-k7	626	0.894	1073	1073	10	1073.0	1073	10	1073.0	1073	10	1073.0
A-n53-k7	664	0.949	1010	1010	10	1010.0	1010	10	1010.0	1010	9	1010.2
A-n54-k7	669	0.956	1167	1167	10	1167.0	1167	8	1170.0	1167	5	1174.3
A-n55-k9	839	0.932	1073	1073	10	1073.0	1073	10	1073.0	1073	10	1073.0
A-n60-k9	829	0.921	1354	1354	6	1357.0	1354	2	1360.2	1361	2	1362.4
A-n61-k9	885	0.983	1034	1034	1	1034.9	1035	0	1035.1	1035	0	1035.0
A-n62-k8	733	0.916	1288	1288	5	1289.6	1288	5	1292.3	1288	2	1295.3
A-n63-k9	873	0.970	1616	1616	6	1617.7	1616	4	1617.2	1616	5	1618.8
A-n64-k9	848	0.942	1401	1401	2	1403.1	1401	1	1405.0	1401	2	1403.9
A-n65-k9	877	0.974	1174	1177	1	1178.8	1174	5	1176.1	1174	5	1176.3
A-n69-k9	845	0.939	1159	1159	6	1161.2	1159	1	1163.7	1159	6	1161.0
A-n80-k10	942	0.942	1763	1763	2	1769.0	1764	4	1771.3	1763	1	1770.6

5. CONCLUSIONES Y RECOMENDACIONES

El estudio presenta una búsqueda tabú novedosa y efectiva para el ruteo de vehículos. El algoritmo cuenta con una lista tabú tridimensional y un mecanismo para penalizar soluciones inviables durante el proceso de **búsqueda.** Adicionalmente, el algoritmo es de fácil implementación, pues solo se requiere sintonizar 2 parámetros, el tenor y el valor de la penalización. La búsqueda tabú se evaluó utilizando un reconocido grupo de casos. Los resultados muestran que el algoritmo encontró la solución óptima en todos los casos. Al momento de implementar el algoritmo se recomienda utilizar un valor para el *tenor* de 5**Destinos* y un incremento en la penalización de 0.0025/*Destinos* por iteración. Finalmente, el presente trabajo se puede expandir agregándole mecanismos de diversificación a la búsqueda tabú o empleándola en conjunto con otros métodos, como las colonias de hormigas.

REFERENCIAS

1. Dantzig, G. B. & Ramser, R. H. (1959). The truck dispatching problem. *Management Science, 6,* 80-91.
2. Garey, M. R. & Johnson, D. S. (1979). *Computers and intractability: A guide tothe theory of NP-completeness*. Nueva York: W. H. Freeman & Co.
3. Glover, F. (1986). Future paths for inter programming and links to artificial intelligence. *Computers and Operations Research, 1*(3), 533-549.
4. Kara, I., Laporte, G., & Bektas, T. (2004). A note on the lifted Miller-Tucker-Zemlin subtour elimination constraints for the capacitated vehicle routing problem. *European Journal of Operational Research, 58,* 793-795.
5. Jaramillo, J. R. (2010). *The single green vehicle routing problem*. Myrtle Beach, Carolina del Sur: SE InfORMS Annual Meeting.
6. Marinakis, Y. & Migdalas, A. (2007). Annotated bibliography in vehicle routing. *Operational Research*, *7*(1), 27-46.

7. Parragh, S. N., Doerner, K. F., & Hartl, R. F. (2008). A survey on pickup and delivery problems. Part I: Transportation between customers and depot. *J Betriebswirtschaft, 58*, 21-51.

8. Parragh, S. N., Doerner, K. F., & Hartl, R. F. (2008). A survey on pickup and delivery problems. Part II: Transportation between pickup and delivery locations. *J Betriebswirtschaft*, *58,* 81-117.