Metaheurísticas aplicadas al ruteo de vehículos. Un caso de estudio. Parte 2: algoritmo genético, comparación con una solución heurística

Metaheuristics applied to vehicle routing. A case study. Part 2: genetic algorithm, compared to a heuristics solution

Guillermo González Vargas[1] y Felipe González Aristizábal[2]

RESUMEN

Este artículo presenta la solución a un problema de ruteo de vehículos a través de dos técnicas diferentes; en primera instancia se aplica un algoritmo genético y se realizan diferentes experimentos, posteriormente se utiliza la metodología de clusterizar primero y rutear después a través de las heurísticas de barrido y búsqueda local, respectivamente. Los resultados de las diferentes técnicas son comparados.

Palabras clave: ruteo de vehículos, algoritmo genético, barrido, búsqueda local.

ABSTRACT

This paper presents a solution to a vehicle-routing problem by using two different techniques. In the first instance, a genetic algorithm was applied (different experiments were done); later on the *cluster first–route second* methodology was used via heuristic sweep and local search, respectively. The results of the different techniques were then compared.

Keywords: vehicle-routing, genetic algorithm, sweep, local search.

Recibido: abril 18 de 2006
Aceptado: septiembre 20 de 2006

Introducción

El presente artículo es el segundo de una serie de tres que fue iniciada en el número anterior de la presente revista, en este se presentan dos metodologías de solución diferentes al problema de ruteo de vehículos (VRP) que surge de la necesidad que una empresa manufacturera presenta para decidir la localización de una bodega de producto terminado desde la cual satisfacer la demanda de sus clientes (la descripción del problema y su formulación matemática se encuentran en el primer artículo).

El artículo comienza con una breve descripción bibliográfica de los conceptos de heurística y metaheurística. En las siguientes líneas los autores presentan el concepto de algoritmo genético, metaheurística ampliamente utilizada en la solución de este tipo de problemas y se ilustra la metodología implementada describiendo de manera detallada los operadores utilizados; adicionalmente se exhíben los resultados obtenidos en los diferentes experimentos realizados. Posteriormente se aborda el problema limitando el espacio de soluciones a través de la utilización de la técnica de clusterizar primero – rutear después (*cluster firts – routen second*) (Olivera, 2004), aplicando para ello dos heurísticas: el barrido para clusterizar y la búsqueda local para rutear; igual que en el caso anterior, se registran los resultados.

Finalmente, los autores comparan los resultados logrados por medio de las diferentes técnicas y con base en ello se exponen algunas conclusiones.

Heurísticas y metaheurísticas

El problema de ruteo de vehículos que compete a esta serie de artículos es de naturaleza combinatoria, tal tipo de problemas es formulado generalmente como programas de optimización entera, los cuales a su vez pueden

[1] Ingeniero industrial, Universidad Nacional de Colombia. Aspirante a M.Sc. en Ingeniería Industrial, Universidad de los Andes, Colombia. gugonza@uniandes.edu.co

[2] Ingeniero industrial, Universidad Nacional de Colombia. Aspirante a M.Sc. en Ingeniería Industrial, Universidad de los Andes, Colombia. fegonza@uniandes.edu.co

ser reformulados o acotados a través de la exploración de conjuntos de soluciones factibles (Crainic y Toulouse, 2003). Esta exploración se realiza por medio de la enumeración implícita, para lo cual han surgido diferentes métodos tales como el *Branch and Bound, Branch and cut, Branch and price;* sin embargo, en muchos problemas del "mundo real" dichos métodos fallan por diferentes motivos, entre los que se pueden nombrar los espacios de búsqueda demasiado amplios, o bien, demasiado complejos debido a las restricciones de los problemas (Burke *et al.*, 2003). Para evitar inconvenientes como los mencionados es común utilizar procedimientos heurísticos, con los cuales se sacrifica la garantía de encontrar una solución óptima a cambio de velocidad y alguna certeza de encontrar cierto nivel de calidad en la solución hallada (Burke *et al.*, 2003), por lo que, según Crainic y Toulouse (2003) las heurísticas son una alternativa muy usada en la solución de problemas de tipo combinatorio.

Según Melián *et al*. (2003) el término "heurística" proviene de la palabra griega *eureka*, utilizada por Arquímedes, y ha sido utilizado para agrupar un conjunto de algoritmos de búsqueda basados en estructuras repetitivas, con los cuales se dirigen procesos de decisión (Burke *et al.,* 2003). En términos generales, una heurística es el procedimiento repetitivo que identifica una solución factible, con la cual se espera acercarse a la solución óptima. En el presente artículo se implementa lo que Torres (2005) denomina heurísticas de dos fases, específicamente la metodología de clusterizar primero y rutear después (*cluster firts–routen second*), que consiste en definir qué nodos del grafo serán abastecidos por determinado vehículo o depósito (Olivera, 2004) para posteriormente determinar la ruta a seguir bajo el esquema de un problema de agente viajero (*TSP,* por sus siglas en inglés *Traveling Salesman Problem*), que puede ser solucionado según Torres (2005) mediante heurísticas de inserción, vecino más próximo, búsqueda local o algoritmo de Clark y Wright (1964).

Aunque autores como Lin (1965) han demostrado las bondades de las heurísticas en problemas de agente viajero, otros (Martí y Moreno, 2003) han sido enfáticos en afirmar que en muchas ocasiones las heurísticas suministran soluciones localmente óptimas que pueden estar muy alejadas (en términos de valor objetivo) de las soluciones óptimas globales, es decir, estas no dejan de tener una gran limitación para superar óptimos locales (Crainic y Toulouse, 2003). Frente a este inconveniente se han desarrollado las denominadas "metaheurísticas", que son estrategias maestras que permiten resolver de manera inteligente un problema (Melián *et al.*, 2003). El término metaheurísticas se obtiene de anteponer a *heurística* el sujeto *meta*, que significa más allá o a un nivel superior, lo que concuerda con la descripción que Crainic y Toulouse (2003) hacen de estas técnicas al afirmar que modifican otras heurísticas para producir mejores soluciones que las encontradas de la manera clásica.

Al igual que las heurísticas, las metaheurísticas no aseguran una exploración sistemática de todo el espacio de soluciones; sin embargo, exploran aquellas regiones en las que se cree es posible encontrar buenas soluciones con base en cierto criterio. Según Crainic y Toulouse (2003), metaheurísticas bien diseñadas pueden evitar problemas de ciclaje (secuencias repetitivas de soluciones) y detención en óptimos locales, con lo cual se supone se deben encontrar mejores soluciones en comparación de las heurísticas clásicas. Lourenço *et al*. (2003) han clasificado las metaheurísticas en dos grandes conjuntos; el primero de ellos, denominado metaheurísticas de vecindario (*Nighborhood based metaheuristics*), contiene técnicas como recocido simulado y búsqueda tabú, las cuales se basan en exploración de soluciones vecinas a través de la heurística de búsqueda local; y el segundo grupo, denominado metaheurísticas multiarranque (*Multi-start based metaheuristics*) entre las que se encuentran los algoritmos de colonias de hormigas y los algoritmos evolutivos (entre ellos los algoritmos genéticos, técnica aplicada en este estudio), genera de manera iterativa diferentes soluciones.

Solución metaheurística: algoritmo genético (GVR)

Como se ilustró en el artículo anterior, una de las técnicas metaheurísticas más ampliamente utilizada en los problemas de ruteo de vehículos son los algoritmos genéticos (GA, por sus siglas en inglés), técnica que ha evolucionado desde el concepto alemán de *Evolutionsstrategie* o el norteamericano de *evolutionary programming* concebido en la época de los sesenta hasta llegar a vincularse con la optimización en 1975 gracias al *Genetic Algoritm* de John Holland (Reeves, 2003). Las soluciones generadas mediante algoritmos genéticos se obtienen al utilizar los conceptos de cruce y la mutación de las ciencias biológicas (LeBlanc *et al.*, 1999) y las reglas naturales de autorreparación y adaptación de los seres vivos (Goldberg, 1989). Según Reeves (2003), el uso de algoritmos genéticos en optimización es muy popular y frecuentemente presenta resultados exitosos en aplicaciones reales, lo que se respalda claramente en el comentario de Goldberg (1989), quien afirma que teórica y empíricamente ha sido probada la posibilidad de lograr búsquedas robustas en espacios complejos con la aplicación de GA.

Según Goldberg (1989), los algoritmos genéticos difieren de la mayoría de procedimientos de búsqueda y optimización normales en cuatro aspectos básicos:

1. Los algoritmos genéticos trabajan con una codificación de parámetros (o genotipo) y no con los parámetros en sí mismos (fenotipo), por esta razón en los GA cada solución (individuo de la población) está representada por un vector denominado cromosoma, en el que cada uno de sus componentes (gen) representa un parámetro de la solución.
2. Los algoritmos genéticos realizan la búsqueda a partir de una población de soluciones y no desde una sola solución, lo cual, según LeBlanc *et al*. (1999), asegura la exploración de una mayor porción del espacio de soluciones y evita la caída en óptimos locales.

3. Los algoritmos genéticos utilizan la información de la evaluación de la función objetivo (*fitness*) para guiar la búsqueda, no conocimiento auxiliar.
4. Los algoritmos genéticos utilizan reglas de transición probabilística y no determinística.

Estructura y codificación del algoritmo genético

Con base en la información recopilada acerca de los algoritmos genéticos y las ventajas que ellos ofrecen en la solución de problemas de tipo combinatorio, se propone la utilización de esta técnica para solucionar a cada uno de los subproblemas de tipo CVRP que surgen de dividir el problema original del que trata este caso, tal como se describió en el primer artículo; por tal motivo, se implementa una estructura general de algoritmo genético como la que se describe a continuación (los operadores son los usados en C++):

```
Escoja una población inicial

While   (no se satisfaga la condición de terminación)
        {
        Evalúe cada individuo
        Ordene la población
        While (no se hayan obtenido suficientes hijos)
                {
                Seleccione los cromosomas padres
                Realice el cruce
                If (se satisface la condición de mutación)
                        Realice la mutación
                If (se realiza la reparación
                }
        }
```

La representación de cada individuo que compone la población, las estrategias de cruce y mutación, y los diferentes parámetros del algoritmo, difieren según los autores e implementadores de las rutinas, por tanto es importante que el lector conozca aquellas condiciones utilizadas en el presente caso.

Codificación de individuos

Los individuos se codifican con un cromosoma (vector) de 52 genes (componentes), cada uno de los cuales debe contener uno de los nodos que componen el grafo (se excluye el nodo que identifica el origen), en ningún caso pueden existir genes repetidos, ya que esto indicaría que un nodo está siendo visitado más de una vez (Figura 1). La ruta que sigue cada vehículo consiste en partir del origen y visitar consecutivamente los nodos que se indican en el cromosoma hasta llegar a uno cuya demanda, aunada con la de todos los nodos visitados anteriormente por el vehículo, supere su capacidad; cuando esto sucede el vehículo se dirige al origen y se inicia la ruta del vehículo siguiente.

Z_1	Z_2	Z_3	Z_4	Z_5	Z_6	…	…	…	Zi	…	…	…	Z_{47}	Z_{48}	Z_{49}	Z_{50}	Z_{51}	Z_{52}

Zi={1,2,3,…,52}; Donde {1,2,3,…,52} representan a cada uno de los nodos del grafo.

Figura 1. Representación del cromosoma

Generación de la población inicial

La población inicial es un conjunto de individuos generados de manera aleatoria de manera que cada uno de sus cromosomas presenta la estructura de la Figura 1. El tamaño de la población a utilizar en la evolución del algoritmo es un parámetro del algoritmo y se ajusta por ensayo y error según los resultados obtenidos.

Condición de terminación

El algoritmo propuesto se detiene después de realizar un determinado número de generaciones (ciclos); al igual que el tamaño de la población, es un parámetro del algoritmo y se ajusta por ensayo y error de tal manera que no se limite la evolución del algoritmo.

Cálculo de fitness (evaluación del individuo)

El costo de cada cromosoma equivale a la suma de los costos de las rutas recorridas por cada uno de los vehículos. Para cada vehículo, el costo de su ruta se calcula con la suma de las distancias recorridas, iniciando con la distancia entre el origen y el primer nodo; posteriormente se suman las distancias entre los pares de nodos consecutivos que aparecen en el cromosoma hasta llegar a aquel que no pueda ser satisfecho con la capacidad del vehículo, en este momento se suma la distancia entre el último nodo visitado y el origen y se inicia el mismo procedimiento para el vehículo siguiente; este proceso se realiza hasta agotar los nodos a visitar. En caso de encontrar individuos que requieran más de seis vehículos para satisfacer la demanda de todos los nodos el algoritmo está programado para penalizarlo incrementando significativamente el costo de su ruta, de tal manera que se reduzca su probabilidad de reproducirse.

Ordenación de la población

Una vez que la población cuenta con su tamaño indicado, el algoritmo realiza una ordenación de los individuos, colocando en primer lugar aquel que posea el mejor *fitness*, es decir aquella solución que presenta la menor distancia total recorrida para satisfacer la demanda de todos los nodos.

Estrategia de selección de padres

El algoritmo propuesto utiliza una estrategia elitista para seleccionar a los individuos que mediante su cruce generan los hijos que conforman la población de la siguiente generación. La estrategia elitista consiste en definir una elite de individuos que por sus características superiores (menor distancia total recorrida) son los encargados de generar, mediante cruces de miembros de la elite, una proporción de individuos de la siguiente generación. Para definir esta estrategia es necesario determinar que porcentaje de la población será considerada elite y qué porcentaje de los hijos será generada por ella, estos porcentajes son parámetros que deben ser definidos por ensayo y error según la evolución del algoritmo. Para generar los hijos restantes hasta completar la población de la siguiente generación se cruzan indistintamente todos los individuos.

Estrategia de cruce

El cruce es una operación realizada sobre dos cromosomas de una generación, los cuales generan dos individuos nuevos (hijos u *Offsprings*) por su combinación. En el algoritmo propuesto por los autores se utiliza el cruce uniforme (Zhou *et al.*, 2002), consistente en la generación de un vector binario de la misma longitud de los cromosomas denominado máscara de cruce, con base en el cual se combinan los genes de los padres. Los componentes del vector máscara se generan de manera aleatoria, lo que garantiza que cada cromosoma hijo contiene una cantidad aleatoria de información de cada uno de los padres. La Figura 2 ilustra el cruce uniforme.

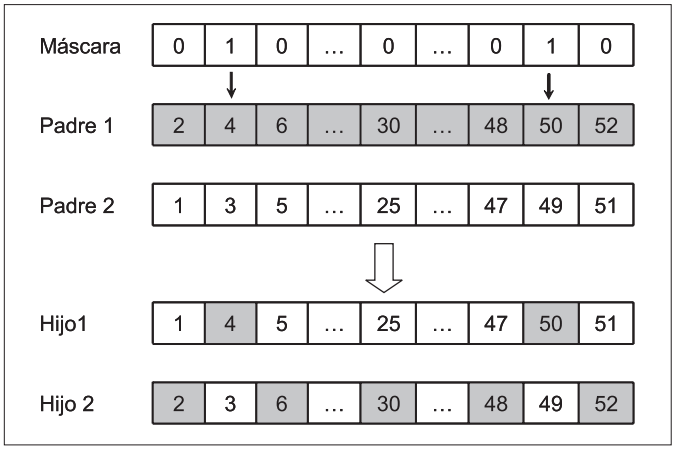


Figura 2. Un ejemplo de cruce uniforme

Estrategia de mutación

La mutación es un operador que produce cambios espontáneos de manera aleatoria en algunos individuos de cada población (Zhou *et al.*, 2002); en el algoritmo propuesto se utiliza como procedimiento de mutación un intercambio entre dos posiciones del cromosoma, es decir, en individuos seleccionados de manera aleatoria (según probabilidad de mutación) se seleccionan aleatoriamente dos posiciones y se intercambian los valores de sus genes (nodos). La Figura 3 muestra un ejemplo del procedimiento de mutación. La probabilidad de mutación es un parámetro del algoritmo y se ajusta por ensayo y error según los resultados obtenidos.

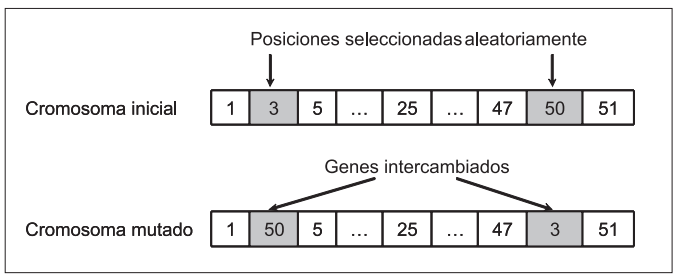


Figura 3. Ejemplo de mutación

Estrategia de reparación

Al generar cromosomas a partir de la combinación de los genes de los padres existe la posibilidad de repetir nodos en los hijos, condición que implica la repetición de la visita a un nodo y por ende la infactibilidad del individuo, ante esta situación es necesario proporcionar al algoritmo una estrategia para reparar individuos que tengan este tipo de falencias. La estrategia de reparación propuesta es la de "primero repetido – primero faltante", es decir, el primer nodo que se encuentre repetido en el cromosoma es reemplazado por el primero que se halla ausente (en orden lexicográfico).

Implementación

Una vez programado el algoritmo descrito el usuario ejecuta el programa ingresando los parámetros iniciales: tamaño de la población, el porcentaje de individuos que componen la elite, porcentaje de hijos generados por la elite, probabilidad de mutación y la cantidad de generaciones a realizar. Estos parámetros son ajustados por el usuario para manipular la evolución del algoritmo y de esta forma mejorar los resultados, tal y como se describirá en las siguientes líneas. El algoritmo genético descrito hasta el momento fue implementado en C++ y ejecutado en una PC Pentium IV de 2.4 Ghz con 512 MB de RAM y 120 Gb de disco duro.

Experimentos

Con el propósito de ajustar los parámetros de tal manera que el algoritmo evolucione de la mejor forma posible y por ende se obtengan buenos resultados, se realizan diferentes experimentos, en cada uno de los cuales se modifican uno o varios parámetros y se observa el comportamiento del algoritmo. Los beneficios de un algoritmo se miden según el valor promedio de los resultados logrados al realizar diferentes réplicas y su desviación típica (con base en la prueba estadística Anova es posible determinar si un juego de parámetros presenta resultados estadísticamente superiores a los obtenidos con valores diferentes), ya que esto es una medida de la posibilidad de obtener buenos resultados al ejecutar de manera constante el algoritmo. Aunado a lo anterior, los autores han decidido tener en cuenta adicionalmente el valor mínimo obtenido con cada juego de parámetros, ya que en última instancia este es el valor con el que se debe tomar la decisión de localización.

Para el caso en estudio se realizaron diferentes experimentos aplicados a los tres nodos candidatos a alojar el centro de distribución de la empresa. Para cada nodo (9, 28, 49) se realizaron siete experimentos diferentes (Tabla 1); la cantidad de réplicas indica el número de veces que se ejecutó el procedimiento, esto con el fin de realizar comparaciones estadísticamente válidas de los resultados conseguidos.

Los cuatro primeros experimentos tienen en común la cantidad de generaciones permitidas y el tamaño de la población. En ellos se modificó el porcentaje de individuos elite, el porcentaje de hijos generados por la elite y la probabilidad de mutación. Para comparar los resultados de estos experimentos se realizó un total de 50 réplicas para cada uno. Con base en los resultados obtenidos (Tabla 2), y teniendo en cuenta que los mejores experimentos (para los tres nodos) fueron el 1 (resultado promedio más bajo) y el 4 (resultado

mínimo más bajo[3]), se decidió continuar la experimentación con sus parámetros, efectuándose los experimentos 5 y 6, los cuales mantienen una probabilidad de mutación del 60%, un porcentaje de individuos elite igual al 40% de la población total, y difieren en la cantidad de hijos generados por la elite que en un caso es del 60% de la población total y en el otro es del 80%. En estos experimentos se permitió una mayor evolución al algoritmo, pasando de 100 a 1.000 generaciones (Tabla 1). Debido a que el tiempo computacional se incrementa al elevar la cantidad de generaciones se decidió reducir el número de réplicas por experimento, en este caso se realizaron 30.

Tabla 1. Experimentos

	Experimentos						
	1	*2*	*3*	*4*	*5*	*6*	*7*
Generaciones	100	100	100	100	1000	1000	500
% de individuos "élite"	40	60	20	40	40	40	40
% hijos generados por la "élite"	80	80	80	60	80	60	80
Probabilidad de mutación	60	80	40	60	60	60	60
Población	50	50	50	50	50	50	100
Cantidad de replicas	50	50	50	50	30	30	20

Tabla 2. Resultados por experimento

		1	2	3	4	5	6	7
Nodo 9	Mínimo	1869,6	2535,9	2060,4	1954,2	1699,8	1756,2	1682,1
	Promedio	2314,4	2857,2	2492,4	2453,2	1875,6	1942,2	1867,5
	Desviación típica	165,9	128,1	274,0	156,9	109,8	100,8	90,3
Nodo 28	Mínimo	2193,2	2720,7	2197,3	2187,2	1906,4	1950,4	1839,2
	Promedio	2527,6	3168,9	2680,7	2751,4	2096,5	2169,7	2091,6
	Desviación típica	185,5	152,0	222,1	225,6	140,6	112,4	120,1
Nodo 49	Mínimo	1913,0	2423,3	2047,0	1994,8	1614,3	1663,7	1492,5
	Promedio	2262,2	2836,5	2390,7	2415,3	1778,3	1918,3	1765,0
	Desviación típica	140,4	139,9	184,7	205,7	145,7	113,7	89,7

Con base en los resultados de los dos últimos experimentos (Tabla 2) se corroboró que para el caso en estudio lo mejor es mantener la cantidad de hijos generados por la elite en un 60% de la población total, ya que el experimento 5 obtuvo el menor resultado promedio, al mismo tiempo que el menor valor mínimo; es importante dar a conocer al lector que al graficar la evolución de los algoritmos se encontró que estos convergen (en todos los casos) al mejor resultado entre las generaciones 300 y 400, por lo cual se hace innecesaria la utilización de las 1.000 generaciones propuestas. Con base en las observaciones anteriores, se decidió realizar un último experimento en el cual se buscó explorar en cada generación un mayor número de soluciones, para lo que se incrementó el tamaño de la población, pasando de 50 a 100 individuos por generación, lo que dio lugar al experimento 7, en él se redujo la cantidad de generaciones a 500.

Para comparar los promedios de los experimentos, se realizó la prueba Anova a través del paquete estadístico Minitab 14.[4] Los resultados para los nodos 9, 28 y 49 se encuentran en las figuras 4, 5 y 6, respectivamente. En ellas se puede apreciar que para los tres nodos hay una diferencia estadísticamente significativa entre los resultados de los experimentos 1 y 2 con relación a los experimentos 3 y 4. En el caso del experimento 1 los resultados son significativamente menores, mientras que en el experimento 2 los resultados son mayores y entre los experimentos 3 y 4 no se distingue ninguna diferencia estadística. En cualquier caso los experimentos 5, 6 y 7 presentan resultados significativamente menores respecto de los cuatro primeros experimentos; sin embargo, en los tres últimos no se observan diferencias distinguibles estadísticamente (salvo en el nodo 48, donde el experimento 7 presenta resultados menores a los encontrados en el 6).

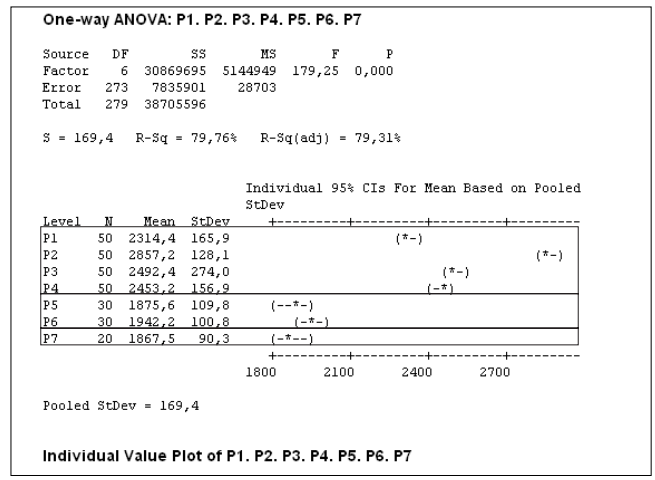


Figura 4. Análisis de varianza para los siete experimentos realizados al nodo 9

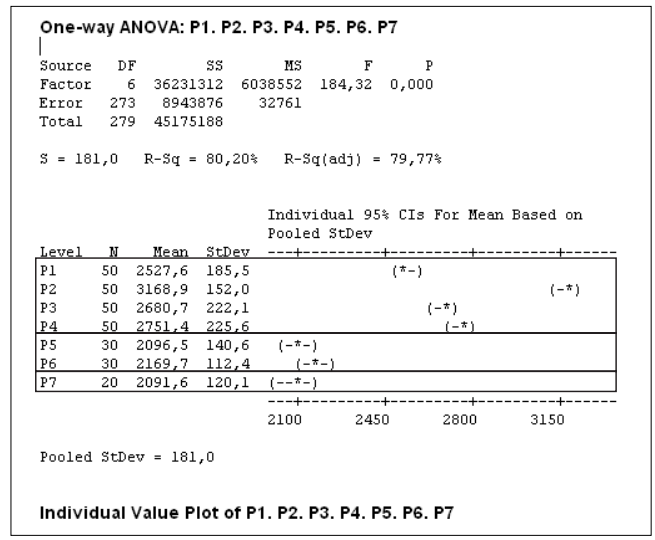


Figura 5. Análisis de varianza para los siete experimentos realizados al nodo 28

[3] Es importante notar que el experimento de resultado mínimo para los tres nodos fue el número 1, por lo tanto se seleccionó el siguiente experimento de menor resultado.

[4] Minitab® Release 14.13. 1972-2004 Minitab Inc. All rights reserved.

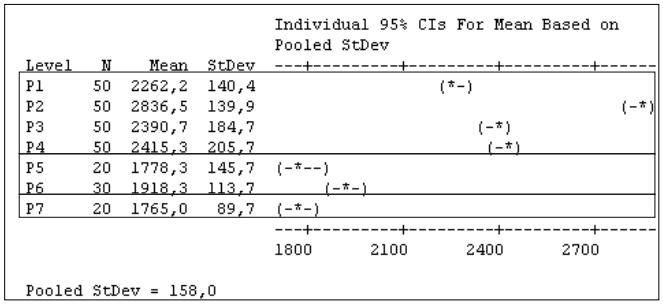


Figura 6. Análisis de varianza para los siete experimentos realizados al nodo 49

La evolución de los algoritmos en el experimento 7, para los nodos 9, 28 y 49, candidatos a convertirse en centros de distribución, es la que se muestra en las figuras 7, 8 y 9, respectivamente. Teniendo en cuenta los resultados logrados en el último experimento, se obtuvo una distancia total a recorrer de 1.682,1 km. partiendo del nodo 9, 1.839,2 km partiendo del nodo 28 y 1.492,5 km. partiendo del nodo 49, con lo cual la empresa debería localizar su punto de distribución en el nodo 49 y recorrer mensualmente un total de 1492,5 Km. para satisfacer la demanda de los 53 nodos.

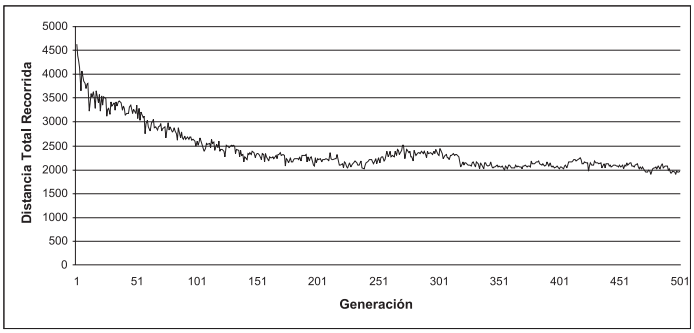


Figura 7. Evolución del experimento 7 para el nodo 9

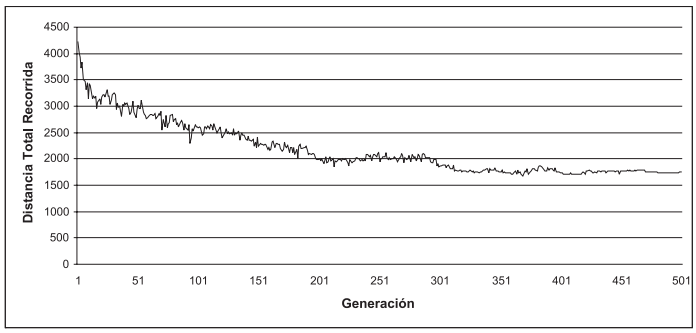


Figura 8. Evolución del experimento 7 para el nodo 28

Solución heurística: clusterizar primero, rutear después

Como se mencionó en el primer artículo, en los problemas de ruteo de vehículos con múltiples depósitos o vehículos es posible reducir la complejidad del problema acotando el universo de soluciones a través de lo que Medaglia (2005) llama *set covering*, o lo que otros autores conocen como clusterizar o asignar primero, rutear después *(cluster firts – routen second)* (Olivera, 2004). Esta técnica también fue aplicada al caso de estudio del presente artículo, utilizando heurísticas para desarrollar las dos etapas, esto con el propósito de comparar los resultados obtenidos a través de la aplicación de heurísticas y metaheurísticas como el algoritmo genético anteriormente expuesto.

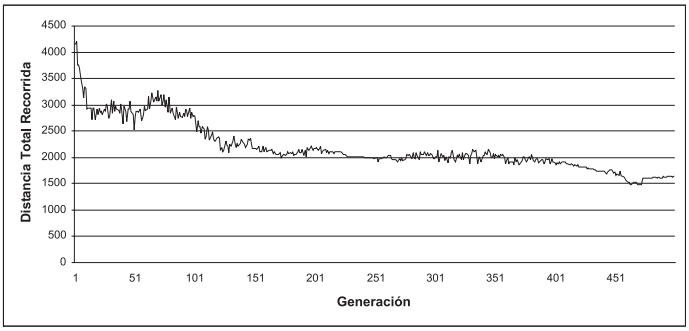


Figura 9. Evolución del experimento 7 para el nodo 49

Clusterización: barrido o *sweep*

Después de evaluar las posibilidades para creación de *clusters* expuestas por Olivera (2004) y Ahuja *et al*. (1993), y buscando utilizar una técnica que involucre la restricción de capacidad de los vehículos, se decidió aplicar la técnica de barrido o *sweep*. Esta técnica propone establecer un punto de origen en el depósito y desde allí realizar un barrido para abarcar toda la área geográfica del problema, determinando así cada uno de los *clusters*. Este proceso sigue los siguientes pasos (anónimo, s/f):[5]

1. Localizar el depósito y todos los clientes en una gráfica.
2. Centrar el punto de origen del barrido en el depósito y trazar una línea recta
3. Realizar el barrido rotando la línea en sentido contrario a las manecillas del reloj.
4. Detener la inclusión de un cliente en el barrido cuando el vehículo haya excedido su capacidad.
5. Reiniciar el barrido hasta asignar todos los clientes a un *cluster*.

Para el caso de estudio, la clusterización se realizó ubicando como ejes centrales las tres posibles localizaciones para la planta, es decir, determinando las coordenadas polares $(\theta,\rho)=(0,0)$ en los nodos 9, 28 y 49, obteniendo las figuras 10, 11 y 12, respectivamente.

Es importante resaltar que algunos *clusters* resultan conformados por un solo nodo, dada su gran demanda, por lo que es necesaria la capacidad completa de un vehículo para abastecerlo. También vale la pena mencionar que al clusterizar desde el nodo 49, solo resultan necesarios cinco vehículos (Figura 12).

[5] Disponible en: www.cs.tcd.ie/courses/baict/bass/4ict5/Networks2004.pdf. Recuperado en marzo de 2005

Ruteo: búsqueda local

Una vez definido el conjunto de clientes a atender (*cluster*) se procede a realizar la asignación de la mejor ruta, este subproblema generalmente subyace en un caso de problema de agente viajero (TSP) y puede resultar tan pequeño como para ser solucionado mediante una técnica de optimización como la programación lineal (Ahuja *et al.*, 1993). No obstante, en este caso de estudio la mayoría de los problemas TSP resultantes tienen dimensiones grandes, por lo que se utilizó la heurística de búsqueda local para definir la secuencia de nodos a visitar y así atender la demanda de cada *cluster*.

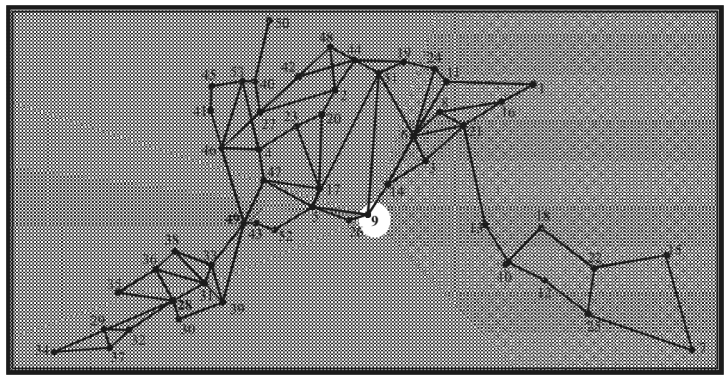


Figura 10. Clusterización a partir del nodo 9

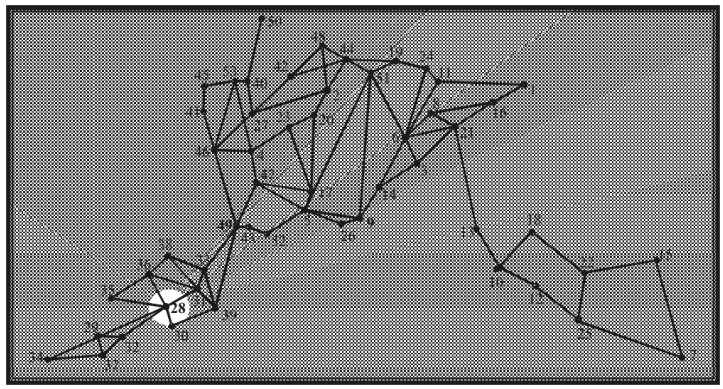


Figura 11. Clusterización a partir del nodo 28

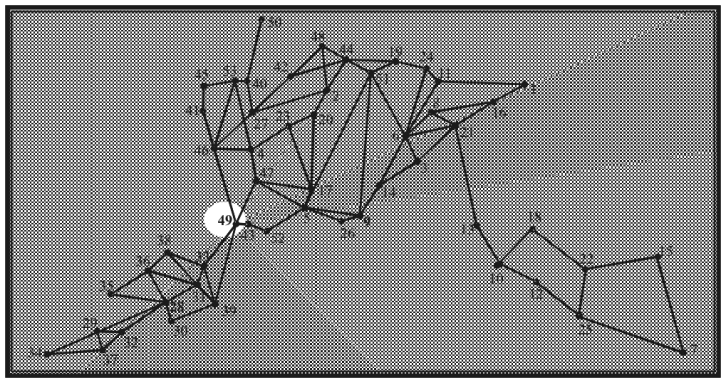


Figura 12. Clusterización a partir del nodo 49

La búsqueda local es una heurística en la que se explora el espacio de soluciones a través de búsquedas de vecindario, esto es, partiendo de una solución inicial (posiblemente generada de manera aleatoria); la búsqueda local se mueve a un "vecino" que representa una mejor solución que la actual de acuerdo a la función objetivo. La heurística se detiene cuando ninguna de las soluciones existentes en el vecindario explorado representan mejoras en la función objetivo, o bien, cuando se considera que el resultado converge (Voudouris y Tsang, 2003). La estructura general de un algoritmo de búsqueda local puede ser ilustrada en lenguaje C++ de la siguiente manera:

```
Escoja una solución inicial y conviértala en solución actual
While (no se satisfaga la condición de terminación)
        {
        Explore todos los vecinos
        If (Mejor vecino es mejor igual que la solución actual)
                {
                Haga Solución actual=Mejor vecino
                If (mejor vecino es igual que la solución actual)
                    Incremente en contador de convergencia
                Else
                    Reinicie el contador de convergencia
                }
        }
```

Codificación de la solución

Las soluciones se codifican con vector n componentes, donde n indica la cantidad de nodos pertenecientes al *cluster* ruteado; el orden en el que aparecen los nodos en el vector indica la secuencia de la ruta, la cual inicia y finaliza en el origen. La solución inicial se genera de manera aleatoria.

Vecinos

Un vecino es una solución candidata que se genera a partir de la modificación de la solución actual; para el caso de estudio la modificación realizada consistió en el intercambio de las posiciones de dos componentes de la solución actual. En la búsqueda local implementada se exploró de manera exhaustiva todo el vecindario de cada solución actual, realizando todos los intercambios posibles entre pares de componentes del vector. A manera de ejemplo, la Figura 13 muestra el vecindario de un vector de cuatro posiciones.

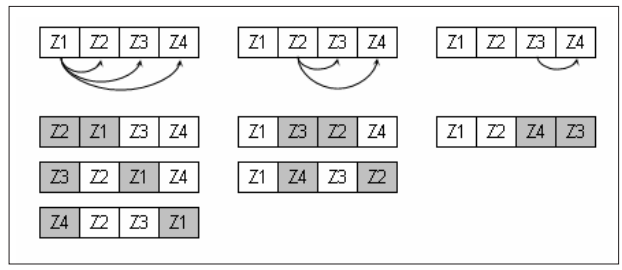


Figura 13. Ejemplo de vecindario

Condición de terminación

En general toda búsqueda local detiene sus iteraciones al encontrar una vecindad en la cual todos los vecinos repre-

sentan soluciones en las que no se mejora la función objetivo. El algoritmo propuesto en este artículo utiliza tal criterio como condición de terminación; sin embargo, se adicionó una segunda condición teniendo en cuenta que al algoritmo propuesto se le permitió cambiar de solución al encontrar un vecino con valor de la función objetivo igual a la actual. Esta segunda condición consiste en limitar la cantidad de veces que se acepta el mismo valor en la función objetivo, es decir, cuando se considera que la solución converge hacia un óptimo (local o global).

Resultados

Una vez realizada la clusterización y programado el algoritmo para efectuar la búsqueda local, se procedió a utilizar esta heurística para rutear los *clusters* conformados por más de un nodo, para este efecto se realizaron 100 réplicas en cada *cluster* y se obtuvieron los resultados de la Tabla 3. Sumando las distancias mínimas requeridas para satisfacer la demanda de cada *cluster*[6] se encontró que para satisfacer la demanda de todos los nodos es necesario recorrer 1.258,59 km partiendo del nodo 9, 1.772,07 km partiendo del nodo 28 y 1.262,26 km partiendo del nodo 49, con lo cual la empresa debería localizar su punto de distribución en el nodo 9 y recorrer mensualmente un total de 1.258,59 km para satisfacer la demanda de los 53 nodos.

Tabla 3. Resultados de la búsqueda local por cluster y nodo de origen

		Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5	Cluster 6
Nodo 9	Promedio	588,28	434,81	N/A*	N/A*	207,03	271,98
	Desv. típica	184,11	116,17	N/A*	N/A*	49,69	36,25
	Mínimo	410,8	287,79	36,2	85,34	188,12	250,33
Nodo 28	Promedio	458,57	67,86	390,25	385,64	304,01	1451,01
	Desv. típica	137	0	113,63	71,04	0	546,07
	Mínimo	299,77	67,86	199,05	282,56	304,01	618,82
Nodo 49	Promedio	524	N/A*	309,71	553,3	197,23	
	Desv. típica	176,19	N/A*	73,64	159,1	43,5	
	Mínimo	419,88	85.34	244,28	348,99	163,77	

******Cluster* **compuesto por un solo nodo. Búsqueda local no aplica.**

Comparación de resultados y conclusiones

Para el caso estudiado en este artículo la aplicación de la heurística resulta ser más efectiva que la solución metaheurística implementada, ya que se observa que en todos los casos los resultados obtenidos a través del algoritmo genético son mayores que los logrados por medio de la combinación del barrido para clusterizar y la búsqueda local para rutear. Los resultados del algoritmo genético son mayores en un 34% para el ruteo a partir del nodo 9, 4% a partir del nodo 28 y 18% a partir del nodo 49.

Sin embargo, al analizar en detalle el comportamiento de la técnica de clusterizar primero y rutear después se detecta que la calidad de los resultados no depende específicamente del ruteo realizado por la búsqueda local, ya que se observa que los resultados de esta son altamente variables (ver desviación típica en la Tabla 3), lo que da pie a pensar que el mínimo valor obtenido es fruto de la aleatoriedad y no de una búsqueda esquemática que haga evolucionar satisfactoriamente el resultado y por ende se trata de óptimos locales que pueden ser mejorados. Por el contrario, al estudiar la clusterización se aprecia que los nodos agrupados en cada *cluster* se convierten en conjuntos que facilitan la distribución de los productos, y en esto radica la efectividad de la técnica.

Teniendo en cuenta lo anterior, queda abierta la posibilidad de desarrollar una solución metaheurística más efectiva, basada en el principio de clusterizar primero y rutear después. Este desarrollo se ilustra en el tercer artículo de la serie.

Bibliografía

Ahuja, R., Magnanti, T. and Orlin, J., Network flows: theory, algorithms, and applications., Englewood Cliffs, New Jersey: Prentice Hall, 1993.

Anónimo S/F., Network., Disponible en: www.cs.tcd.ie/courses/baict/bass/4ict5/ Networks2004.pdf. Consultado en Marzo de 2004.

Burke, E., Kendall, G., Newall, J., Hart, E., Ross, P. and Schulenburg, S., Hyper-heuristics: an emerging direction in modern search technology., En: Glover, F. y Kochenberger, G. (Eds.). Handbook of metaheuristics. Kluwer academic publisher, 2003.

Clarke, G. and Wright W., Scheduling of vehicles from a central depot to a number of delivery points., Operations Research, 12, 1964, pp.568-581.

Crainic, T. and Toulouse, M., Parallel strategies for meta-heuristics., En: Glover, F. y Kochenberger, G. (Eds.), Handbook of metaheuristics, Kluwer academic publisher, 2003.

Goldberg, D.E., Genetic algorithms in search, optimization and machine learning., Addison- Wesley, 1989.

LeBlanc, L., Shtub, A., and Anandalingam, A., Formulating and solving production planinning problems., En: European Journal of Operational Research, No. 112, 1999.

Lin, S., Computer solutions of traveling salesman problem., Bell Systems Tech. J., 44, 1965.

Lourenço, H., Martin, O. and Stützle, T., Iterated local search., En: Glover, F. and Kochenberger, G. (Eds.), Handbook of metaheuristics, Kluwer academic publisher, 2003.

Martí, R. y Moreno, M., Métodos Multiarranque. Inteligencia Artificial., Revista Iberoamericana de Inteligencia Artificial. No.19, 2003, pp. 49-60.

Medaglia, A., Combinatoria para Logística., Coloquio en Optimización Combinatoria Sesión Avanzada, Universidad de los Andes, marzo, 2005.

[6] En aquellos cluster en los que no se realizó el ruteo se sumó la distancia requerida para ir y volver del nodo origen al único nodo que compone el cluster.

Melián, B., Moreno, J. y Moreno, M., Metaheurísticas: una visión global. Inteligencia Artificial., Revista Iberoamericana de Inteligencia Artificial, No.19, 2003, pp. 7-28.

Olivera, A., Heurísticas para Problemas de Ruteo de Vehículos., Instituto de Computación, Facultad de Ingeniería. Universidad de la República, Montevideo, Uruguay. 2004, Disponible en: www.fing.edu.uy/inco/pedeciba/bibliote/reptec/ TR0408.pdf. Consultado en Febrero de 2005.

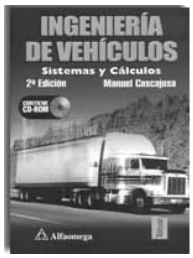
Reeves, C., Genetic algorithms., En: Glover, F. and Kochenberger, G. (Ed.), Handbook of metaheuristics, Kluwer academic publisher, 2003.

Torres, J. F., Problemas de Ruteo de Vehículos., En: Modelos de Sistemas Logísticos, Maestría en Ingeniería Industrial, Universidad de los Andes, 2005.

Voudouris, C. and Tsang, E., Guided local search., En: Glover, F. and Kochenberger, G. (Eds.), Handbook of metaheuristics, Kluwer academic publisher, 2003.

Zhou, G., Min, H. and Gen, M., The balanced allocation of customer to multiple distribution centers in the supply chain network: a genetic algorithm approach., En: Computer and Industrial Engineering, No. 43, 2002.

Alfaomega Colombiana S.A.



INGENIERÍA DE VEHÍCULOS
Sistemas y Cálculos

CASCAJOSA, Manuel

548 págs. Rústica, 17 x 23 cm
ISBN 970-15-0943-9
Coedición: Alfaomega-Tébar

NOVEDAD

Un texto dirigido a técnicos y estudiantes de mecánica automotriz y también para los profesionales vinculados al sector de la automoción que tendrán en sus manos un libro de consulta. Con un enfoque hacia el vehículo industrial este libro brinda un amplio panorama de la ingeniería de vehículos: pesos y dimensiones de los vehículos, adherencia, la caja de cambios, el grupo reductor y diferencial, el bastidor, la dirección, hasta la seguridad activa y pasiva en el automóvil
Explica el funcionamiento y tipos de suspensión: flexibilidad, ballestas, muelles helicoidales, barras de torsión, resortes de goma, suspensiones conjugadas y neumáticas

Adquiera nuestros textos en el punto de venta **Alfaomega Carrera 15 No 64a - 29** o en las principales librerías del país.

Afíliese a nuestro **CLUB DEL CONOCIMIENTO** a través de nuestra página web**,** y reciba descuentos en nuestro punto de venta, contenidos actualizados vía Internet, información de novedades, prioridad en productos promocionales y entregas a domicilio sin costo adicional.

Visite nuestra página Web:
www.alfaomega.com.co