Universidad Nacional Autónoma de México

Facultad de Ciencias

Un Algoritmo Genético para tratar el Problema de Rutas de Vehículos

T E S I S

QUE PARA OBTENER EL TÍTULO DE:

Licenciado en Ciencias de la Computación

PRESENTA:

Fernando Michel Tavera

TUTORA

Dra. María de Luz Gasca Soto

Ciudad de México 2018

1. Datos del alumno
Michel
Tavera
Fernando
26 50 43 40
Universidad Nacional Autónoma de México
Facultad de Ciencias
Ciencias de la Computación
412056325

2. Datos del tutor
Dra.
Gasca
Soto
María de Luz

2. Datos del sinodal 1
Dra.
Rodríguez
Vázquez
Katya

2. Datos del sinodal 2
Dr.
Galaviz
Casas
José de Jesús

2. Datos del sinodal 3
Dr.
Peláez
Valdéz
Canek

2. Datos del sinodal 4
M. en C.
Zerón
Martínez
Carlos

3. Datos del trabajo escrito
Un Algoritmo Genético para tratar el Problema de Rutas de Vehículos
62 p.
2018

Introducción

El Problema de Rutas de Vehículos consiste en realizar una serie de entregas desde un depósito central con una flota dada de vehículos, minimizando costos. Sin embargo, al ser un problema *NP*-difícil, resulta inviable abordarlo con algoritmos tradicionales, por ello en este trabajo se propone un método heurístico para tratar el problema.

A continuación se da una breve descripción del contenido y la organización de este trabajo.

En el Capítulo 1 se detalla el Problema de Rutas de Vehículos, dando una descripción general del mismo y después una definición matemática formal.

En el Capítulo 2 se presenta una descripción general del funcionamiento de la heurística conocida como Algoritmos Genéticos.

En el Capítulo 3 se describe la estructura de la implementación específica para el Algoritmo Genético utilizado, así como el pseudocódigo para el cuerpo principal del mismo.

En el Capítulo 4 se presentan algunas variantes consideradas para el Algoritmo Genético propuesto y se describe cuáles fueron utilizadas y cuáles rechazadas y por qué.

En el Capítulo 5 se muestran los resultados de pruebas experimentales con un conjunto de ejemplares públicamente disponibles.

Finalmente se detallan las conclusiones alcanzadas.

La lista de referencias bibliográficas se encuentra al final del documento.

Índice

1 El Problema de rutas de vehículos

El Problema de Rutas de Vehículos, mejor conocido por sus siglas en inglés VRP (*Vehicle Routing Problem*) consiste en encontrar la mejor manera de realizar una serie de entregas desde un depósito central con una flota dada de vehículos, de tal forma que el costo derivado del uso de dichos vehículos sea el menor posible.

En este capítulo se discuten los detalles del VRP.

1.1 Antecedentes del VRP

Según De Jaegere *et al.* [2], los primeros en publicar un modelo de Rutas de Vehículos fueron Dantzig y Ramser (1959), aunque ellos trataban específicamente con el problema de surtir combustible a gasolineras desde un depósito central, minimizando la distancia recorrida por las pipas que transportan el combustible.

Fueron Clarke y Wright (1964) quienes generalizaron el problema para tratar con entregas de bienes que pueden cambiar de ejemplar a ejemplar, con clientes que pueden pedir cantidades distintas de esos bienes y considerando algún costo que no necesariamente sea la distancia recorrida; el problema generalizado es conocido como VRP, por las siglas en inglés de *Vehicle Routing Problem*, o Problema de Rutas de Vehículos.

El VRP tiene importantes aplicaciones en la industria, ya que nos dice cómo optimizar la manera de transportar bienes, servicios o personas que requieran llevarse a varios lugares, de modo que el costo de ese transporte sea el mínimo posible.

Esto puede ser, por ejemplo, surtir tiendas de la forma más barata posible con algún producto que sea llevado desde la fábrica o un depósito intermedio; entregar el correo en el mínimo tiempo posible; trazar una ruta eficiente para un paseo turístico de modo que se evite tráfico y se pase por varios puntos de interés; entre muchas otras aplicaciones concretas.

Es debido a esa gran utilidad que el VRP ha sido ampliamente estudiado por más de medio siglo, muchas veces siendo adaptado a los requisitos específicos de problemas reales, para lo cual se han desarrollado numerosas variantes del VRP, cada una con restricciones específicas. Algunas de esas variantes se exponen brevemente a continuación.

1.2 Definición del VRP y variantes

El VRP como lo conocemos hoy en día consiste de una flota dada de vehículos, todos ellos con las mismas características, principalmente la misma capacidad de carga, a los cuales se les asigna una ruta de modo que cada vehículo parte de un almacén central, surte a determinados clientes y finalmente regresa al almacén para terminar su recorrido.

Las rutas deben asignarse de forma que cada cliente sea visitado una sola vez por un solo vehículo y satisfaciendo su demanda completamente, además de que cada vehículo debe recorrer una sola ruta y no debe sobrepasar la capacidad de carga. En ocasiones, también se limita la máxima distancia que puede recorrer un vehículo en su ruta.

El objetivo del problema es que la asignación de rutas se realice de tal forma que minimice el costo asociado a recorrerla, ya sea distancia, uso de combustible, tiempo o algún otro criterio, e incluso alguna combinación de varios o todos estos criterios.

También se suele tener por objetivo minimizar el número de vehículos empleados.

En esta tesis se trabajará con una flota de vehículos dada, de modo que se considera una cota máxima para el número de vehículos a utilizar, pero tomaremos el costo total como el único criterio para determinar qué tan buena es una asignación de rutas.

Esta variante es conocida simplemente por VRP, aunque a veces también se le llama CVRP, por las siglas de *Capacitated Vehicle Routing Problem* o Problema de Rutas de Vehículos con Capacidad fija.

Otras variantes comunmente estudiadas en la literatura [2] incluyen:

- El Problema de Rutas de Vehículos con Flota Heterogénea, denotado HFVRP por las siglas en inglés de *Heterogeneous Fleet VRP*, donde cada vehículo tiene asociada una capacidad que no necesariamente es igual a la de los otros. Esta variante también es llamada VRP con Flota Mixta.
- El Problema de Rutas de Vehículos con Ventanas de Tiempo, o VRPTW por las siglas de *VRP with Time Windows*, agrega la restricción de que cada cliente debe surtirse en cierto intervalo de tiempo. Si algún cliente es visitado fuera

de la ventana de tiempo correspondiente, puede incurrirse en una penalización a la calidad de la solución, o incluso rechazarse por completo esa asignación de rutas, dependiendo del ejemplar específico que se considere.

- El Problema de Rutas de Vehículos con Múltiples Depósitos, o MDVRP por las siglas de *Multi Depot VRP*, incluye varios depósitos distribuidos en la misma área geográfica que los clientes, por lo que cada vehículo debe asignarse a un depósito en particular para iniciar y terminar su ruta.

- Similar a la variante anterior, el Problema de Rutas de Vehículos con Recolección y Entrega, o VRPPD por las siglas de *VRP with Pickup and Delivery*, establece que algunos puntos de la ruta son para entrega y otros son para recolección, por lo que el mismo vehículo debe pasar por los dos puntos correspondientes y debe visitar el punto de recolección antes que el punto de entrega.

Como puede apreciarse, estas variantes son generalizaciones del VRP que agregan más aspectos a considerar en comparación con el problema original. Por ejemplo, puede pensarse que el CVRP es un caso particular del HFVRP en el cual todos los vehículos resultan tener la misma capacidad de carga, o que es un caso del VRPTW donde las ventanas de tiempo son lo suficientemente amplias para garantizar que la entrega se realiza dentro del intervalo asignado.

De ahí que el CVRP resulta ser la variante más sencilla de las mencionadas para ser abordada computacionalmente, pero eso no quiere decir que sea un problema simple, pues incluso esa versión aparentemente básica constituye un problema *NP*-difícil [7], por lo cual resulta necesario emplear métodos alternativos para tratar el problema, como se discute en el siguiente capítulo.

1.3 Definición formal del VRP

A continuación se presenta una definición matemática formal del VRP:

Parámetros:

- n - número de entregas por realizar.

- m - número de vehículos.

- $V = \{v_1, v_2, ..., v_m\}$ - conjunto de m vehículos, cada uno con la misma capacidad de carga máxima.

- M_D - capacidad de carga máxima para los vehículos.
- M_P - distancia máxima para una ruta.
- $C = \{c_1, c_2, ..., c_n\}$ - conjunto de n clientes.
- $D: \ C \to \mathbb{R}$ - función de demandas, donde $D(c_i)$ es la demanda del cliente c_i y se cumple $0 \ < \ D(c_i) \ \leq \ M_D, \ \forall i, \ i \ = \ 1, \ 2, \ ..., \ n$.
- c_0 - depósito.
- $P : C \cup \{c_0\} \times C \cup \{c_0\} \ \to \ \mathbb{R}$ - función de pesos, donde $P(c_i, \ c_j)$ es el costo de que un vehículo vaya de c_i a c_j y $P(c_i, c_i) \ = \ 0, \ \forall i, \ i \ = \ 0, \ 1, ..., \ n$.

Una solución al problema VRP es un conjunto $R = \{r_1, r_2, ..., r_m\}$ de rutas, donde cada r_i es la ruta para el vehículo v_i, tal que $r_i = [c_{i0}, c_{i1}, ..., c_{il(i)}]$, con $l(i)$ la longitud de la ruta r_i, y $c_{ij} \in \ C \cup \{c_0\}, \ \forall j, \ j \ = \ 0, \ 1, ..., \ l(i)$.

El peso p_i de una ruta r_i se define como la suma de las distancias entre elementos consecutivos de r_i; es decir,

$$p_i \ = \ \sum_{j=1}^{l(i)} P(c_{i(j-1)}, c_{ij}).$$

El peso total $T(R)$ de la solución es la suma de los pesos de todas las rutas; esto es,

$$T(R) = \sum_{i=1}^{m} p_i.$$

Una solución para el VRP es factible si satisface lo siguiente:

- la carga total de cada vehículo no sobrepasa la capacidad

$$\sum_{j=1}^{l(i)-1} D(c_{ij}) \ \leq M_D, \ \forall i, \ i \ = \ 1, \ 2, ..., \ m.$$

- la distancia total de cada ruta no sobrepasa la cota

$$p_i \leq M_P, \ \forall i, \ i \ = \ 1, \ 2, \ ..., \ m.$$

- todas las rutas inician y terminan en el depósito

$$c_{i0} = c_{il(i)} = c_0 \ \forall i,\ i\ =\ 1,\ 2, ...,\ m.$$

- cada entrega se realiza una única vez, por un solo vehículo

$$\forall i,\ i\ =\ 1,\ 2, ...,\ n\ \exists!\ r \in R \text{ tal que } c_i \in r.$$

Así, el objetivo es encontrar una solución factible R^* minimizando su peso total $T(R^*)$.

A continuación se destacan algunas notas sobre el modelo matemático empleado:

- Como se mencionó anteriormente, se emplea un número fijo de vehículos m. Esto no altera el problema ni las soluciones alcanzables, ya que m se toma solamente como una cota superior a la cantidad de rutas empleadas. Cualquier vehículo que no fuera necesario para satisfacer a todos los clientes será asignado a una ruta vacía de la forma $r\ =\ [c_0,\ c_0]$, la cual no aporta al peso total y, por lo tanto, no afecta la calidad de la solución.

- Se incluye el parámetro de distancia máxima, por considerarse que si el ejemplar específico no limita la distancia, bastará con asignar un valor suficientemente grande a ese parámetro para que cualquier ruta quede por debajo de la cota.

- Se supone que existe un camino entre cada par de clientes, es decir, el peso $P(c_i,\ c_j)$ tiene está definido para todos los valores de i y j. Esto puede justificarse al notar que si no hay un camino directo entre la posición geográfica de dos clientes, puede tomarse la ruta que va del primer cliente al depósito central y del depósito al segundo cliente, por lo que siempre existe al menos esa forma de llegar. Así, se puede garantizar que la función P está definida en todo su domino.

1.4 Ejemplo del VRP

Consideremos la gráfica de la Figura 1, la cual representa un ejemplar del VRP, a modo de ilustrar el problema.

Los vértices representan al depósito c_0 y a los clientes desde c_1 hasta c_5, cada cliente indica su demanda D y cada arista indica el peso P asociado al par de vértices incidentes, por ejemplo $P(c_1,\ c_2)\ =\ 3$, la capacidad de carga es 1.0 y no hay cota máxima para la distancia. Los colores de las aristas no tienen significado y solamente

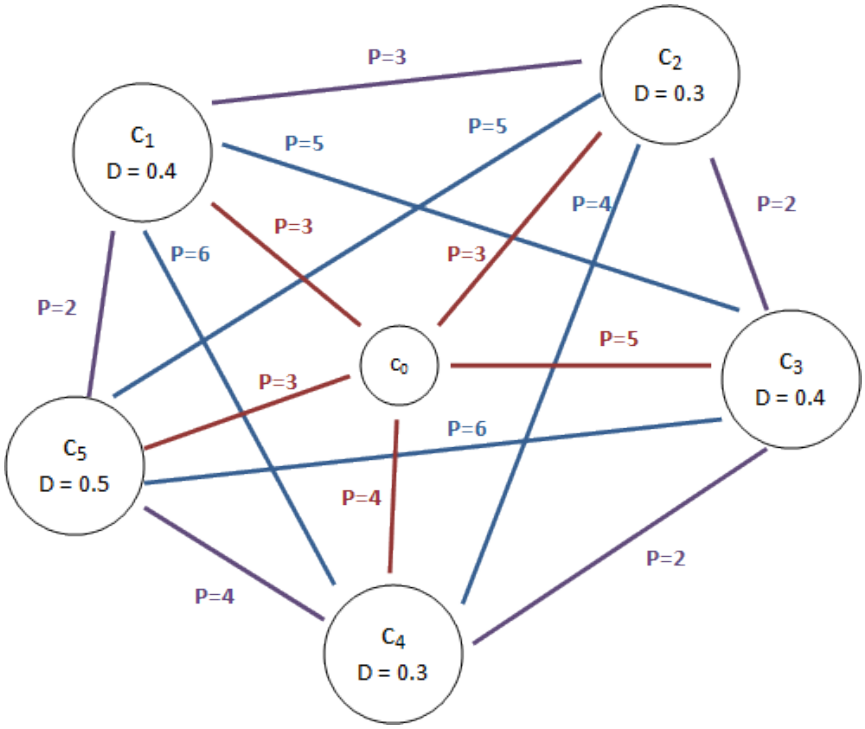

Figura 1: Ejemplar del VRP representado como una gráfica.

son para facilitar la lectura.

Así, podemos aplicar la definición formal antes mencionada a este ejemplar:

- El número de clientes es $n = 5$.
- La suma de todas las demandas es 1.9, por lo que claramente no existirá solución factible con un solo vehículo, así que se asignará el número máximo de vehículos como $m = 2$.
- Por lo tanto, el conjunto de vehículos será $V = \{v_1, v_2\}$.
- El conjunto de clientes es $C = \{c_1, c_2, c_3, c_4, c_5\}$.
- La función de demanda D está dada por los valores dentro de los vértices: $D(c_1) = 0.4$, $D(c_2) = 0.3$, $D(c_3) = 0.4$, $D(c_4) = 0.3$, $D(c_5) = 0.5$.
- Se tiene el depósito c_0, el cual no es un cliente y, por lo tanto, no tiene un valor de demanda.

- La función de pesos P está dada por los pesos marcados en las aristas. Nótese que en este caso los pesos son simétricos; es decir, $P(c_i,\ c_j) = P(c_j,\ c_i)$ para cualesquiera $i,\ j \in [0,\ 1, ...,\ n]$. Sin embargo ese no necesariamente es el caso para cualquier ejemplar, y el modelo matemático no lo requiere, como tampoco el método propuesto en este trabajo.

Una vez definido el ejemplar, se procede a buscar una solución factible, o sea un conjunto de rutas que satisfaga todas las condiciones del problema.
A continuación se muestran dos posibles soluciones a este ejemplar:

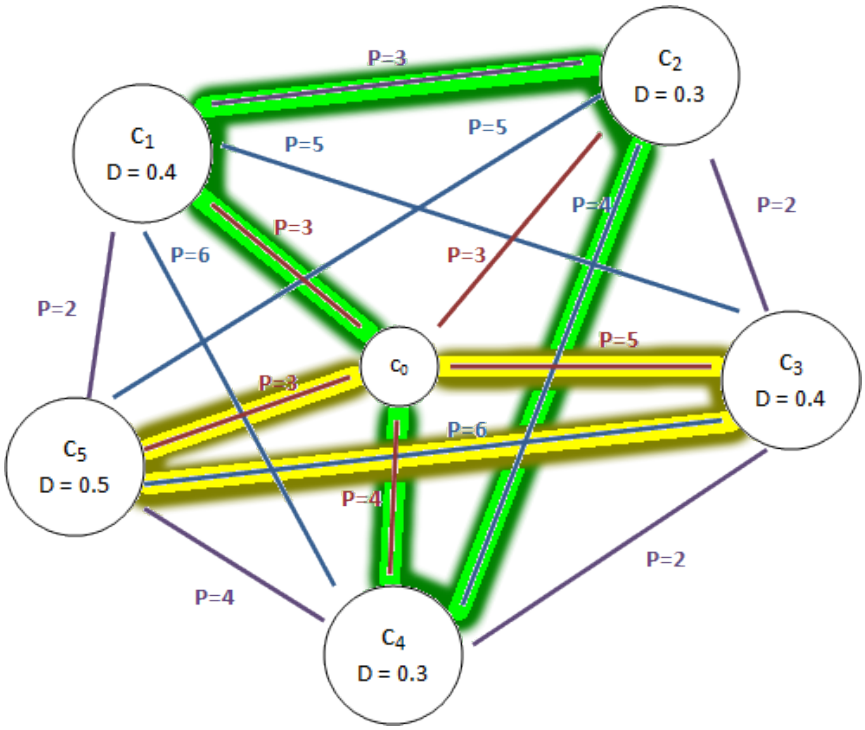

Figura 2: Solución factible para el ejemplar dado del VRP.

En la primera solución, presentada en la Figura 2, se asignó al vehículo v_1 el cliente c_1 y se continuó *en orden* con c_2. Como $D(c_1) + D(c_2) + D(c_3) > 1$, se sobrepasaría la capacidad de carga del vehículo si se agregara c_3 a la ruta, por lo cual lo *saltamos* y terminamos la ruta visitando c_4 y regresando al depósito c_0, con lo cual queda

$$r_1 = [c_0,\ c_1,\ c_2,\ c_4,\ c_0]$$

que es la ruta marcada en verde.

Los clientes no visitados por v_1 se asignan a v_2, con lo cual se obtiene la ruta

$$r_2 \ = \ [c_0, \ c_3, \ c_5, \ c_0]$$

marcada en amarillo.

Vemos entonces que la carga total para el vehículo v_1 es:

$$D(c_1) \ + \ D(c_2) \ + \ D(c_4) \ = \ 0.4 \ + \ 0.3 \ + \ 0.3 \ = \ 1.0$$

.
La carga total para v_2 es:

$$D(c_3) \ + \ D(c_5) \ = \ 0.4 \ + \ 0.5 \ = \ 0.9$$

por lo que ningún vehículo sobrepasa su capacidad de 1.0.

Además, cada entrega es realizada por un vehículo y solo uno, ambas rutas inician en c_0 y ambas terminan en c_0, así que la solución dada por $R \ = \ \{r_1, \ r_2\}$ es factible.

Por último, calculamos $T(R)$ sumando los pesos de todas las aristas marcadas:

$$\begin{aligned} T(R) \ &= \ P(c_0, \ c_1) \ + \ P(c_1, \ c_2) \ + \ P(c_2, c_4) \\ + \ P(c_4, \ c_0) + P(c_0, \ c_3) \ &+ \ P(c_3, \ c_5) \ + \ P(c_5, \ c_0) \\ = \ 3 \ + \ 3 \ + \ 4 \ + \ 4 \ + \ 5 \ + \ 4 \ + \ 3 \ &= \ 26. \end{aligned}$$

Otra posible solución, siguiendo las mismas convenciones para la representación, se presenta en la Figura 3.

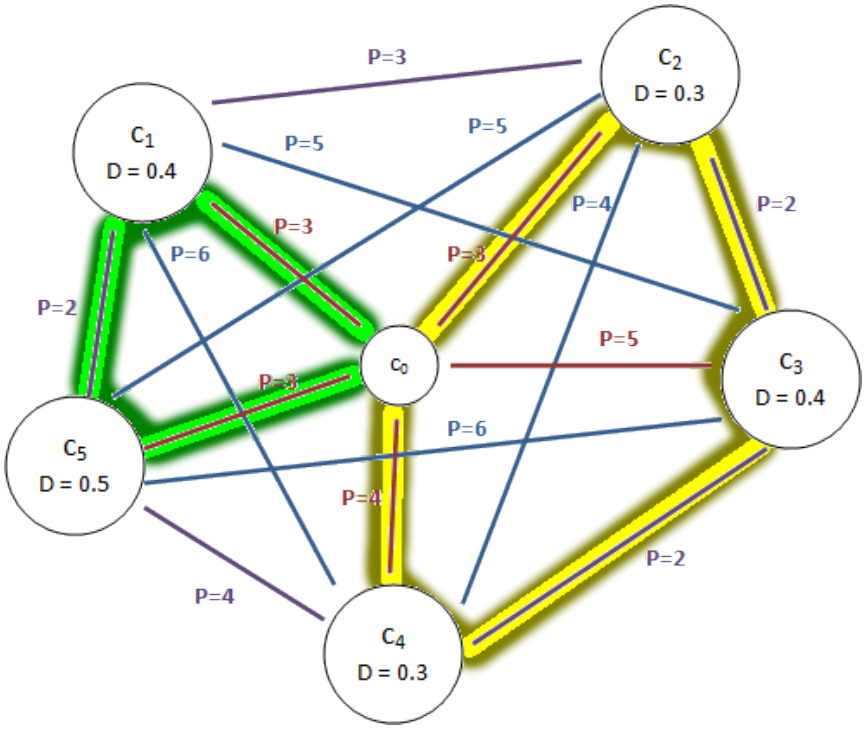

Figura 3: Otra opción de solución para el ejemplar dado.

Si el vehículo v_1 sigue la ruta r'_1 marcada en verde, y v_2 toma la ruta r'_2 marcada en amarillo, vemos que la carga total de v_1 es

$$D(c_1) + D(c_5) = 0.4 + 0.5 = 0.9.$$

La carga de v_2 es:

$$D(c_2) + D(c_3) + D(c_4) = 0.3 + 0.4 + 0.3 = 1.0.$$

Además, las rutas están correctamente formadas, por lo que la solución $R' = \{r'_1,\ r'_2\}$ es factible.

Vemos entonces que el peso total queda:

$$T(R') = P(c_0,\ c_1) + P(c_1,\ c_5) + P(c_5,\ c_0)$$
$$+ P(c_0,\ c_2) + P(c_2,\ c_3) + P(c_3,\ c_4) + P(c_4,\ c_0)$$
$$= 3 + 2 + 3 + 3 + 2 + 2 + 4 = 19.$$

Tenemos entonces que la solución dada por R' tiene un peso de 19, menor al peso de R, que es de 26, así que R' es una mejor solución.

Con este ejemplo es fácil apreciar que para buscar la solución R^* de costo mínimo, comparar todas las posibles combinaciones de rutas es impráctico. En realidad, el VRP es un problema *NP*-difícil [7], lo que quiere decir que no se dispone de métodos que permitan encontrar una solución óptima en un tiempo razonable para propósitos prácticos, como se explica en el siguiente capítulo.

2 Algoritmos Genéticos

Debido a que en la vida real raras veces se dispone del tiempo necesario para obtener soluciones a problemas *NP*-difíciles con el uso de algoritmos tradicionales, es necesario el uso de técnicas de optimización, las cuales permiten obtener una *buena solución* dentro de un tiempo razonable, aunque no se puede garantizar que esa *buena solución* sea la mejor posible.

Tales técnicas de optimización incluyen los algoritmos de aproximación, los métodos numéricos, las heurísticas, las meta-heurísticas, entre otras. En este trabajo se utilizará una heurística conocida como Algoritmo Genético o AG, la cual se describe durante el desarrollo de este capítulo.

2.1 Problemas *NP*-difíciles y heurísticas

Existen problemas para los cuales no se conocen métodos de solución que sean tanto óptimos como eficientes; es decir, sabemos calcular la solución exacta del problema, pero hacerlo requiere de una gran cantidad de recursos computacionales, principalmente tiempo, lo cual lleva a que no sea viable en términos prácticos abordar el problema directamente. Tales problemas incluyen los denominados *NP*-difíciles, categoría que contiene al VRP, como se mencionó anteriormente.

Más aún, actualmente no se sabe a ciencia cierta si encontrar un método eficiente de solución exacta para un problema NP-difícil es siquiera posible, interrogante que equivale en cierta forma al problema abierto conocido como "*P vs NP* ", uno de los llamados Problemas del Milenio.

Dicho esto, tampoco es viable evitar este tipo de problemas, ya que como se estableció en el Capítulo 1, los problemas *NP*-difíciles y el VRP en particular se presentan en numerosas situaciones reales de gran importancia para la industria y para la vida cotidiana.

Así, debemos recurrir a aproximar la solución a estos problemas, utilizando para ello métodos que permiten obtener respuestas de forma eficiente, aunque sin garantía de que la respuesta encontrada sea óptima. Tales técnicas de optimización incluyen los algoritmos de aproximación, los métodos numéricos, las heurísticas, las meta-heurísticas, entre otras.

Las meta-heurísticas en particular han sido ampliamente estudiadas y aplicadas para optimizar el VRP [2]. En la investigación para este trabajo, los mejores resultados encontrados casi siempre fueron obtenidos con una de dos técnicas: la primera denominada Búsqueda Tabú, o TS por las siglas en inglés de *Taboo Search*; y en menor grado la llamada ACO, por las siglas de *Ant colony Optimization*, en español Optimización por Colonia de Hormigas.

Sin embargo, en este trabajo se decidió utilizar otra heurística prometedora, aunque menos prevalente, conocida como Algoritmo Genético o AG. Esta decisión se debe precisamente a una menor presencia en la literatura estudiada, la cual lleva a suponer que es más probable aportar algo nuevo; además de una mayor familiaridad del autor con esta técnica, habiendo atendido e impartido cursos sobre el tema.

La descripción y el funcionamiento de un AG se abordan a continuación.

2.2 Antecedentes de los AG

Según Reeves [6], los Algoritmos Genéticos fueron desarollados originalmente en la Universidad de Michigan durante las décadas de los 1960s y los 1970s por J. H. Holland y sus colegas, con el objetivo de mejorar el desempeño de los métodos de optimización basados en búsquedas aleatorias, los cuales eran ampliamente utilizados en ese tiempo.

Esa mejora se logra introduciendo un mecanismo que dé enfoque o dirección a la búsqueda, obteniendo lo que a veces es llamado una Búsqueda Aleatoria Dirigida o Guiada.

Así, la idea que sigue el diseño de los AG es que, en muchas ocasiones, una forma de encontrar soluciones más eficientes a problemas es partir de varias soluciones ya existentes y combinarlas de alguna forma para obtener un mejor resultado.

Al repetir esa estrategia, combinando las mejores soluciones disponibles y luego volviendo a combinar las soluciones resultantes iterativamente, se logra obtener gradualmente soluciones cada vez mejores.

El argumento es que esa técnica funciona de manera similar al proceso de la crianza selectiva, ya sea de ganado, granos, perros de raza o alguna otra especie, donde se

seleccionan los ejemplares que poseen características deseables y se los cruza para promover la aparición de esas mismas características en la siguiente generación.

Los AG son entonces una heurística bio-inspirada, la cual genera varias posibles soluciones, usualmente de forma aleatoria e intenta combinarlas iterativamente para obtener mejores resultados cada vez, con base en los principios de la evolución biológica:

- En una población de individuos de una especie, cada individuo tiene características que lo hacen más o menos apto para sobrevivir en su medio ambiente.
- Los individuos más aptos tienen una mayor probabilidad de alcanzar la madurez y reproducirse.
- Cuando un individuo se reproduce, sus descendientes pueden heredar las características que lo hacen apto.
- Al transcurrir varias generaciones, la especie tiende a adaptarse a su entorno, mejorando en general la aptitud de los individuos que conforman la población.

El proceso por el cual se aplican estos conceptos para generar buenas soluciones a problemas es descrito a continuación.

2.3 Descripción general de un AG

Un AG trabaja como se describe a continuación [4] [6]:

Se generan varias posibles soluciones al problema dado, cada una de las cuales se representa como una secuencia de datos llamados genes. La secuencia completa se denomina cromosoma o individuo.

Esos individuos son evaluados según una función de aptitud, la cual asigna a una solución un valor numérico denominado el valor de adaptación o aptitud del individuo, el cual es mayor mientras mejor sea la solución.

En un AG clásico o puro cada gen es un bit y el individuo es la representación binaria de la posible solución, pero en ocasiones es más práctico permitir que un gen sea un número en cierto rango, una letra, o algún otro tipo de dato, siendo el cromosoma una cadena de valores permitidos; cada uno de esos valores permitidos para un gen es llamado alelo y cada gen del cromosoma puede tener un conjunto de

alelos propio, no necesariamente igual que el de otros genes.

El conjunto de las soluciones generadas es llamado una población, y el AG opera un número predeterminado de iteraciones, llamadas generaciones en este contexto, o hasta alcanzar algún criterio de detención definido por quien implemente el AG, construyendo en cada generación un nuevo conjunto de individuos a partir de la población anterior por medio de los llamados operadores genéticos, los cuales funcionan combinando los genes de algunos individuos, llamados padres, para producir miembros de la nueva generación, denominados hijos.

Un individuo en una generación tendrá mayor probabilidad de ser seleccionado como padre mientras mayor sea su aptitud, lo cual provoca que las generaciones posteriores tengan una tendencia a contener individuos con mayor aptitud; es decir, mejores soluciones.

Al finalizar la ejecución del AG, se regresa como respuesta al individuo mejor adaptado de todas las generaciones, el que tiene mayor aptitud, ya que dicho individuo representa la mejor solución encontrada.

La Figura 4 muestra un diagrama del funcionamiento general de un AG.

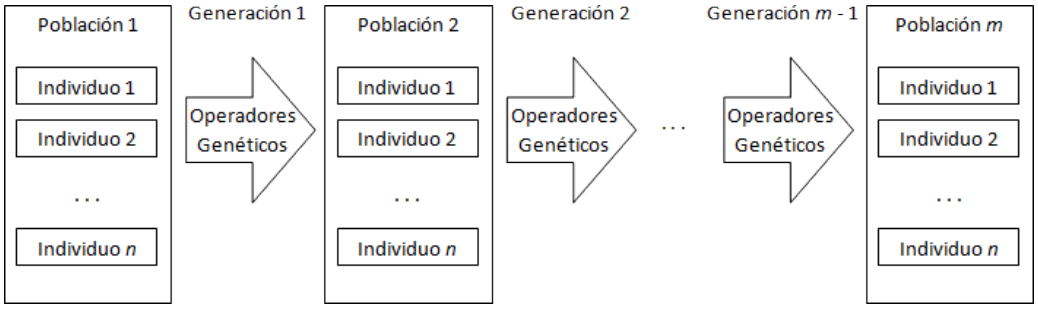

Figura 4: proceso iterativo para un AG con poblaciones de tamaño n

2.4 Operadores genéticos

Los individuos, definidos en la sección anterior, son combinados para formar la siguiente generación por medio de los operadores genéticos que se describen brevemente a continuación:

Selección:
El operador de selección tiene como objetivo elegir elementos de la población, de tal manera que mientras mayor sea el valor de aptitud de un individuo, mayor será la probabilidad de elegir a ese individuo. Usualmente se eligen dos individuos y frecuentemente se requiere que sean diferentes.

Este operador es sumamente importante, ya que es el análogo de la presión selectiva que el ambiente ejerce sobre los seres vivos, aunque algunos autores consideran a la selección como una parte de la lógica principal del AG y no como un operador genético.

Cruzamiento:
Este operador recibe los individuos padres elegidos por el proceso de selección y los combina para generar nuevos individuos hijos, los cuales forman parte de la nueva población, que será trabajada en la siguiente generación.

Se puede decir que el Cruzamiento es el operador genético más importante, ya que su función es generar las nuevas soluciones de forma tal que se fomente la transmisión de características deseables entre una generación y la siguiente.

Elitismo:
Este operador no es indispensable para un AG, pero se incluye en este trabajo debido a su gran utilidad para mejorar la calidad de las soluciones. [4]

El operador de elitismo consiste en garantizar que el individuo con el mayor valor de adaptación de una generación siempre pase a la siguiente población sin ningún cambio.

Esto resulta extremadamente útil, ya que garantiza que el individuo mejor adaptado de la población siempre tendrá un valor de adaptación al menos tan bueno como cualquier individuo de todas las generaciones anteriores.

En otras palabras, este operador da una garantía de la tendencia a mejorar las soluciones, a diferencia de un AG que no emplea elitismo, en el cual existe el riesgo de que los nuevos individuos generados por el cruzamiento sean todos peor adaptados que la generación anterior.

Mutación:

Un problema que puede darse al trabajar con AG es que los operadores antes descritos provocan una tendencia a que los individuos generados sean cada vez más parecidos, esto debido a que el mejor adaptado tiene mayor probabilidad de reproducirse, por lo cual puede suceder que se produzcan clones en el cruzamiento cuando los padres son parecidos. Esto llevaría a la denominada convergencia temprana de las posibles soluciones, donde tras algún número de generaciones ya no se generan nuevos individuos, sino que solo se replican los ya generados y, por lo tanto, no se exploran nuevas regiones del espacio de soluciones, traduciéndose en un riesgo de que el AG quede *atrapado* en un óptimo local de la función de aptitud.

Para evitar eso se implementa el operador de mutación, el cual genera cambios aleatorios en los individuos de cada nueva generación con el fin de introducir mayor diversidad entre las posibles soluciones, con lo cual se induce la exploración más general de todo el espacio de soluciones.

El operador de Mutación no es requerido estrictamente para que un AG sea considerado correcto, bastando para ello que los operadores de Selección y Cruzamiento realicen sus funciones correctamente. Sin embargo, en este trabajo se incluye dicho operador debido a su gran utilidad para extender el área explorada del espacio de búsqueda y evitar caer en óptimos locales.

Evaluación:
Para que el proceso de selección de los padres pueda llevarse a cabo, primero es necesario que se determine el valor de aptitud de cada individuo, para así poder compararlos.

Ese cálculo queda a cargo de una rutina de evaluación, la cual no suele considerarse un operador genético *per se*, pero se menciona aquí debido a su gran importancia.

Este proceso de evaluación es el que más varía de un AG a otro, ya que debe adaptarse al problema que se desea optimizar, por lo que casi siempre debe construirse a la medida para la implementación en cuestión, asegurando siempre que se le asigne un mejor valor de aptitud a una mejor solución.

A veces también es necesario garantizar que los valores de adaptación sean siempre positivos, o que ningún individuo tenga una aptitud de cero, dependiendo de cómo se implemente el operador de selección.

Así, el proceso para construir una nueva población entre generaciones consiste en aplicar repetidamente los operadores genéticos sobre la población dada hasta tener suficientes individuos para formar una nueva población del mismo tamaño, como muestra el diagrama de la Figura 5.

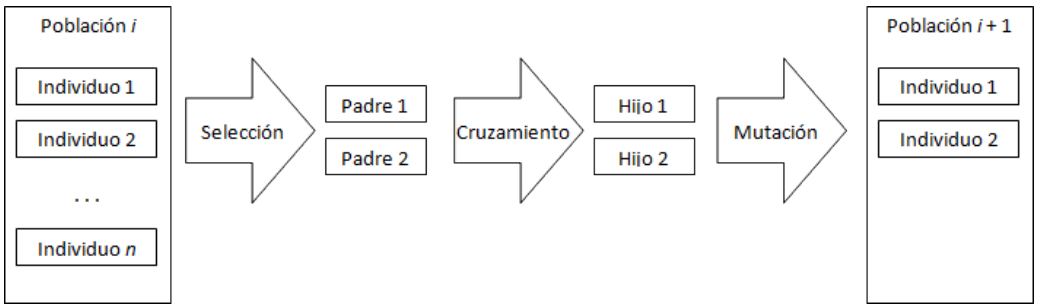

Figura 5: Diagrama de la generación de nuevas poblaciones

Cabe mencionar que en la literatura existen otros operadores adicionalmente a los aquí mencionados, además de que para todos los operadores presentados en esta sección existe una gran cantidad de variantes en su estrategia e implementación, pero este trabajo no pretende explorar todas esas variantes, por lo que solamente se describirá la forma en que se implementaron estos cinco procesos para utilizarlos en el AG propuesto.

2.5 Esquemas

Una teoría que intenta explicar *a posteriori* la eficacia de los AG es la de los llamados esquemas [6], la cual se describe brevemente a continuación:

Como se mencionó en la Sección 2.2, los AG operan sobre individuos constituidos por una cadena de genes, teniendo cada uno de esos genes un conjunto de posibles valores o alelos.

Si se eligen ciertos genes de la cadena y se fijan sus valores para determinados alelos, se tiene entonces un esquema.

La Figura 6 muestra dos individuos del lado izquierdo y cuatro esquemas del lado derecho; los asteriscos (*) representan un gen cuyo valor no es fijo en ese esquema.

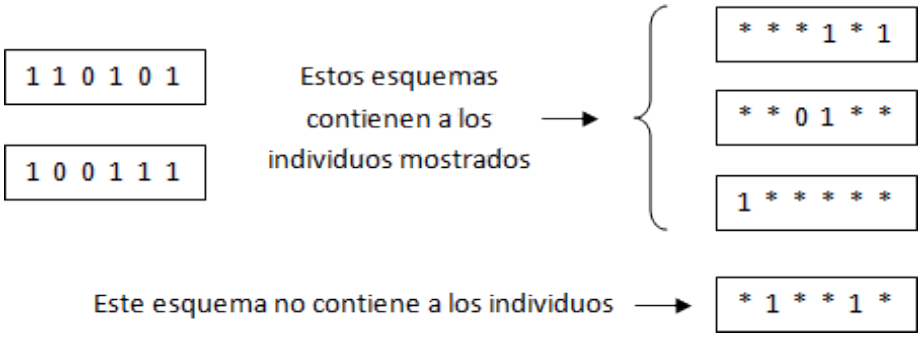

Figura 6: Diagrama de varios individuos y esquemas

Ambos individuos de la Figura 6 tienen todos los valores fijos iguales que los tres primeros esquemas, por lo cual se dice que esos esquemas contienen a ambos cromosomas. El último esquema, en la parte inferior de la figura, no contiene a los individuos, ya que difiere de estos en al menos un gen.

De esta forma, un esquema puede ser visto como una regla de correspondencia que define un conjunto de individuos en cierta forma similares, los cuales se esperaría tengan también características similares.

Otra manera de ver a un esquema es como una región o *hiperplano* en un espacio n-dimensional, donde n es el tamaño de un cromosoma, siendo los puntos sobre esa región individuos que el esquema contiene, con lo cual los esquemas nos dan una forma de dividir el espacio de soluciones, lo que facilita la tarea de buscar en qué subregión se encuentran las mejores soluciones.

La idea es que un cromosoma es representante de cada uno de los esquemas que lo contienen, por lo cual, al evaluar la aptitud de un individuo, se está recopilando información sobre la calidad de muchos cromosomas. Es debido a ello, según la teoría de los esquemas, que los AG son tan eficientes para encontrar buenas soluciones en relativamente pocas generaciones.

Así, cuando se dice que el operador de cruzamiento tiene como objetivo propagar las características deseables de los individuos, lo que eso significa es que el cruzamiento propaga los esquemas que producen buenos individuos, lo cual se logra propiciando que se generen cromosomas contenidos en esos esquemas deseables.

3 Procedimiento propuesto

En este capítulo se describe el Algoritmo Genético propuesto, detallando primero la manera en que se representan las posibles soluciones, luego la estructura general del AG en sí y finalmente la implementación de los operadores genéticos empleados, así como algunas otras subrutinas.

3.1 Representación de una solución

Primero, dada una solución $R = \{r_1, r_2, ..., r_m\}$, necesitamos definir su representación como individuo del AG:

Aunque los términos "individuo" y "cromosoma" suelen utilizarse de forma indistinta en la literatura sobre AG, en esta implementación usaremos dos cromosomas para cada individuo, con lo cual el individuo que representa a R queda definido por los siguientes dos arreglos de enteros:

- $ruta[1..n]$ con entradas en el intervalo $[1, m]$, que representa para cada cliente el índice del vehículo que lo surtirá; es decir, $ruta[i] = j$, si $c_i \in r_j$.
- $orden[1..n]$ cuyas entradas son una permutación de los números $1, 2, ..., n$, la cual indica el orden en el cual se visitarán los clientes; es decir, $orden[1] = i$ si c_i es el primer cliente que será visitado, $orden[2] = j$ si c_j es el segundo cliente en ser visitado, y así sucesivamente.
  Aunque esta representación indica un orden total de los clientes, solamente nos interesará la relación entre clientes que sean visitados por el mismo vehículo.

Así, la forma en la que el AG crea soluciones es eligiendo, para cada vehículo, a cuáles clientes surtirá y en qué orden los visitará.

$T(R)$, el costo total de la solución, puede ser calculado simplemente recorriendo los arreglos para reconstruir las rutas y sumando las distancias entre clientes consecutivos. El valor así calculado se usa para obtener la aptitud del individuo, la cual será inversamente proporcional a $T(R)$, ya que mientras menor sea el costo total, mejor será la solución. El procedimiento concreto se describe en la Sección 3.2, al definir el procedimiento de evaluación empleado.

Cabe mencionar que el valor de aptitud así encontrado, también se almacena en un registro del individuo, con el fin de evitar realizar el cálculo más de una vez.

3.2 Estructura del algoritmo genético

Con base en lo presentado en el Capítulo 2, el Listado 1 a continuación presenta la estructura general del AG propuesto en forma de pseudocódigo:

Listado 1 Algoritmo Genético

Entrada: X, número de parejas en una solución; Y, número máximo de generaciones

Salida: Muestra en pantalla la ruta para cada vehículo en la mejor solución producida por el Agloritmo Genético.

```
1: maximo ← calcularMax( ); // calcula aptitud máxima
2: poblacion ← generarInicial(X); // crea arreglo de 2X + 1 individuos aleatorios
3: elite ← nuevo individuo;
4: elite.aptitud ← −1;
5: EvaluaPoblacion(poblacion, maximo); // asigna aptitud a cada individuo
6: repetir
7:    pobAux ← ∅; // arreglo auxiliar de tamaño 2X
8:    para cada ind desde 1 hasta n/2 hacer
9:       p1, p2 ← SeleccionPadres(poblacion); // índices de los padres
10:      h1, h2 ← Cruzamiento(p1, p2); // genera los individuos hijos
11:      mutacion(h1); // puede mutar el primer hijo
12:      mutacion(h2); // puede mutar el segundo hijo
13:      pobAux[2 ∗ ind − 1] ← h1; // almacena el primer hijo
14:      pobAux[2 ∗ ind] ← h2; // almacena el segundo hijo
15:   termina para cada
16:   EvaluaPoblacion(pobAux, maximo);
17:   poblacion ← pobAux;
18: hasta alcanzar Y generaciones
19: imprimeIndividuo(mejorIndividuo); // muestra el resultado
```

Nota: se supone que todos los procesos tienen acceso a los datos del ejemplar para el cual se ejecuta el AG; a saber: el número n de clientes, el número m de vehículos disponibles, la función D de demandas, la función P de pesos y las cotas M_D y M_P.

El pseudocódigo para las rutinas correspondientes a los operadores genéticos y otras rutinas auxiliares se incluye en las dos secciones siguientes.

3.3 Operadores genéticos

En esta Sección se describen los operadores genéticos empleados en el AG propuesto para llevar a cabo el proceso evolutivo sobre los individuos descritos en la Sección 3.1. Asimismo, se presenta en forma de pseudocódigo la implementación de dichos operadores genéticos, así como algunas otras rutinas auxiliares.

Nota: además del acceso a los parámetros del ejemplar, como se mencionó en la sección anterior, también se presupone la existencia de una función $random(num)$, la cual produce un número aleatorio en el intervalo real $(0,\ num]$ tomado mediante una distribución uniforme, así como una función $randInt(num)$, la cual genera un número entero aleatorio en el intervalo $[1,\ num]$ tomado mediante una distribución uniforme.

Selección:

El operador de selección es el conocido como Algoritmo de Ruleta, que es comúnmente utilizado con los AG. En este método se calculan los valores de adaptación de todos los individuos en la población actual y después se itera hasta completar la siguiente población, eligiendo en cada iteración a dos individuos aleatoriamente, los cuales serán los padres de dos individuos de la nueva población. Nótese que debido a esto es preferible que el tamaño de la población siempre sea un múltiplo de dos, para poder llenar la nueva población exactamente con un número entero de iteraciones.

Sin embargo, los padres no se eligen de forma completamente azarosa, sino que cada individuo tiene una probabilidad de ser elegido, la cual es mayor mientras mejor sea su aptitud.

Conceptualmente esto forma una especie de "diagrama de pastel", donde el tamaño de la "rebanada" que corresponde a cada individuo es proporcional a su valor de adaptación, y sobre el cual se elige aleatoriamente un punto para determinar al individuo seleccionado, de ahí el nombre de "Algoritmo de Ruleta".

Este proceso le da mayor probabilidad de ser selccionado a un individuo cuya "rebanada" cubre mayor área, como se observa en la Figura 7.

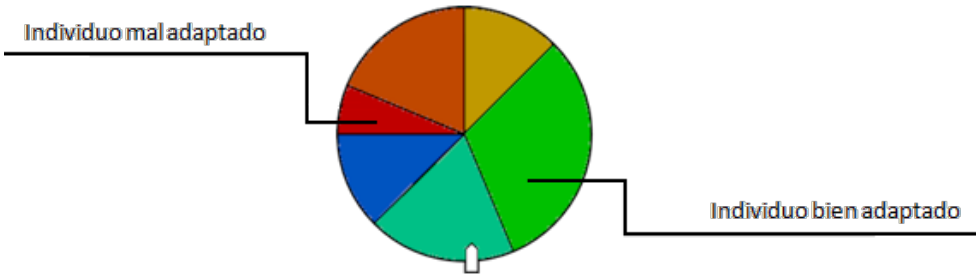

Figura 7: Visualización del Algoritmo de Ruleta

El Listado 2 a continuación presenta el pseudocódigo correspondiente a este proceso:

Listado 2 Selección

Entrada: *poblacion*, arreglo de individuos del cual serán seleccionados dos padres.
Salida: p_1 y p_2, dos individuos seleccionados de la población.

```
1: se inicializa cuenta ← 0;
2: para cada ind ∈ poblacion; hacer
3:    cuenta ← cuenta + ind.aptitud; // calcula el tamaño de la ruleta
4: termina para cada
5: randNum ← random(cuenta); // posición aleatoria en la ruleta
6: para cada ind ∈ poblacion; hacer
7:    si randNum ≤ ind.aptitud entonces
8:       p1 ← ind // la ruleta cayó en ind
9:       interrumpe para cada
10:   en otro caso
11:      randNum ← randNum − ind.aptitud; // se "recorre" la ruleta
12:   termina si-entonces
13: termina para cada
14: randNum ← random(cuenta); // se elige el segundo individuo
15: para cada ind ∈ poblacion; hacer
16:   si randNum ≤ ind.aptitud entonces
17:      p1 ← ind
18:      interrumpe para cada
19:   en otro caso
20:      randNum ← randNum − ind.aptitud;
21:   termina si-entonces
22: termina para cada
23: regresa (p1, p2);
```

Cruzamiento:

Este operador produce dos individuos hijos a partir de los dos padres seleccionados. En este trabajo se utiliza el llamado Cruzamiento de un Punto, el cual consiste en lo siguiente:

Se elige un punto aleatorio del cromosoma, denominado punto de cruce y se divide a los cromosomas de los padres en dos partes: la parte izquierda, formada por todos los genes antes del punto de cruce y la parte derecha, formada por los genes después del punto de cruce.

A continuación se generan los cromosomas de dos individuos hijos, tomando para el primer hijo la parte izquierda del primer padre concatenada con la parte derecha del segundo padre para formar el cromosoma y para el segundo hijo se toma la parte izquierda del segundo padre concatenada con la parte derecha del primer padre. El proceso se ejemplifica en la Figura 8.

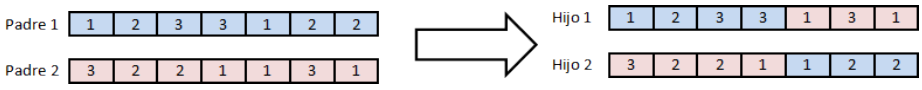

Figura 8: Ejemplo de un cruzamiento por punto

Sin embargo, en este trabajo solo puede emplearse este método directamente para uno de los dos cromosomas definidos, a saber, el que indica en qué ruta se visita cada cliente, ya que el otro cromosoma, que indica el orden, es una permutación de números y tomar las dos partes de los cromosomas padres no garantiza que el hijo resultante sea correcto, como se observa en la Figura 9.

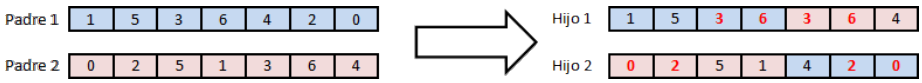

Figura 9: Los individuos resultantes del cruzamiento no son permutaciones correctas

Por ello para el segundo cromosoma se utiliza una variación del cruzamiento de punto, donde en lugar de tomar la parte derecha de un padre tal cual aparece en su cromosoma, se toman los elementos de la permutación faltantes en el orden en que aparecen en el padre del cual no se tomó la parte izquierda. La Figura 10 muestra un ejemplo.

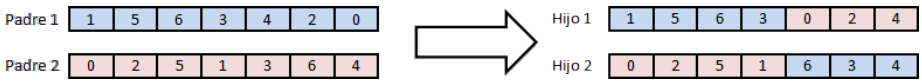

Figura 10: Con esta variación se producen permutaciones correctas

Así, el operador de cruzamiento para este trabajo realiza independientemente los cruzamientos de punto para los dos cromosomas que componen a cada individuo. El Listado 3 a continuación presenta el pseudocódigo correspondiente.

Listado 3 Cruzamiento

Entrada: p_1 y p_2, individuos que serán cruzados.

Salida: h_1 y h_2, dos individuos generados a partir del cruzamiento por punto de p_1 y p_2.

```
1: se crean los arreglos aparece_1[1...n] y aparece_2[1...n]
2: se crean individuos h_1 y h_2
3: para cada i desde 1 hasta n hacer
4:    aparece_1[i] ← falso;
5:    aparece_2[i] ← falso;
6: termina para cada
7: puntoCruce ← randInt(n); // se elige el punto de cruce
8: para cada i desde 1 hasta puntoCruce hacer
9:    h_1.ruta[i] ← p_1.ruta[i]; // primera parte de los hijos
10:   h_1.orden[i] ← p_1.orden[i];
11:   aparece_1[p_1.orden[i]] ← verdadero; // se marcan los indices ya visitados
12:   h_2.ruta[i] ← p_2.ruta[i]; // primera parte de los hijos
13:   h_2.orden[i] ← p_2.orden[i];
14:   aparece_2[p_2.orden[i]] ← verdadero; // se marcan los indices ya visitados
15: termina para cada
16: para cada i desde puntoCruce + 1 hasta n hacer
17:   h_1.ruta[i] ← p_2.ruta[i]; // segunda parte del primer cromosoma
18:   h_2.ruta[i] ← p_1.ruta[i]; // primera parte de los hijos
19: termina para cada
20: j_1 ← puntoCruce + 1;
21: j_2 ← puntoCruce + 1;
22: para cada i desde 1 hasta n hacer
23:   si aparece_1[p2.orden[i]] = falso entonces
24:      h_1.orden[j_1] ← p_2.orden[i]; // segunda parte del segundo cromosoma
25:      j_1 ← j_1 + 1; // se copian índices sin marcar
26:   termina si-entonces
27:   si aparece_2[p1.orden[i]] = falso entonces
28:      h_2.orden[j_2] ← p_1.orden[i];
29:      j_2 ← j_2 + 1;
30:   termina si-entonces
31: termina para cada
32: regresa (h_1, h_2);
```

Elitismo:

La implementación del elitismo consiste simplemente en conservar un registro del individuo mejor adaptado, denominado el individuo de élite: cada vez que se evalúa un individuo, se compara el valor de adaptación recién obtenido con el del individuo de élite y si el nuevo es mejor, se registran sus cromosomas como el nuevo individuo de élite.

Además, cada vez que se termina de construir una nueva población, se agrega a la misma el individuo de élite, para permitir que tenga oportunidad de reproducirse incluso si ninguno de los hijos de la nueva generación sustituyó al individuo de élite de la generación anterior.

Esta copia del individuo de élite no se cuenta en el tamaño de la población cuando se llevan a cabo la selección y el cruzamiento, ya que no se genera en pares como se hace con los hijos, por lo cual si el parámetro que indica el tamaño de la población tiene valor t, la población en realidad consistirá de $t + 1$ individuos, contando al individuo de élite.

Mutación:

El operador de mutación empleado en este trabajo selecciona aleatoriamente cuáles genes se mutarán para cada nuevo individuo generado. Esos genes se operan como sigue:

Para el cromosoma que indica en qué ruta se visita cada cliente, se elige nuevamente una ruta al azar en la cual se surtirá al cliente correspondiente. Para el cromosoma que indica el orden de visita, el valor del gen que se va a mutar es intercambiado con el de otro gen elegido aleatoriamente.

Cabe mencionar que la probabilidad de mutación debe ser muy pequeña, ya que de lo contrario se introduciría demasiada aleatoriedad al proceso y podría perderse frecuentemente el progreso logrado con el fin de mejorar las soluciones.

El Listado 4 presenta el pseudocódigo correspondiente a este operador. Se presupone que el proceso tiene acceso al valor $probaMut$, con $0 < probaMut < 1$, el cual es un parámetro del AG e indica la probabilidad de mutar un individuo.

Listado 4 Mutación

Entrada: *ind*, un individuo que será mutado.
Salida: el individuo de la entrada ha sido mutado con probabilidad $probaMut$.

```
1: randNum ← random(1); // verifica si se mutará al individuo
2: si randNum > probaMut entonces
3:    regresa ind; // no hay mutación
4: termina si-entonces
5: para cada i desde 1 hasta n hacer
6:    randNum ← random(1); // determina mutación para el primer cromosoma
7:    si randNum ≤ 0.5 entonces
8:       ind.ruta[i] ← randInt(n); // un gen muta con probabliidad 1/2
9:    termina si-entonces
10:   randNum ← random(1); // determina mutación para el segundo cromosoma
11:   si randNum ≤ 0.5 entonces
12:      pos ← randInt(n);
13:      tmp ← ind.orden[pos]; // intercambia un gen aleatorio
14:      ind.orden[pos] ← int.orden[i];
15:      int.orden[i] ← tmp;
16:   termina si-entonces
17: termina para cada
```

Evaluación:

Como se mencionó anteriormente, el valor de aptitud de cada individuo será inversamente proporcional al peso total de la solución que representa, para ello al inicio del algoritmo se calcula un valor máximo de peso. Después, al momento de evaluar un individuo, el peso total de su solución es calculado y restado del valor máximo para obtener la aptitud del individuo, con lo cual se consigue que a menor peso total corresponda un mayor valor de aptitud.

El valor máximo se calcula como la suma de los pesos de todas las aristas más uno, de modo que ningún individuo tendrá un peso total igual o mayor que el peso máximo, con lo cual evitamos problemas de "rebanadas" con tamaño negativo en el proceso de selección.

Sin embargo, esto no es suficiente, ya que los operadores de cruzamiento y mutación pueden en ocasiones generar un individuo que sea representación de una solución no factible, pero que aún así tenga un peso total bajo. Por ello, si se detecta que una solución no es factible, su valor de adaptación es sobrescrito con cero, con lo cual se evita que sea devuelto como respuesta final, además de que su parte de la ruleta tiene también tamaño cero y, por lo tanto, las soluciones no factibles no se reproducen.

Cabe mencionar que debido a la representación elegida de los individuos, solamente puede generarse una solución no factible si se sobrepasa la carga de un vehículo, por lo cual la verificación puede ser realizada fácilmente por el mismo proceso que calcula el valor de adaptación del individuo.

El Listado 5 a continuación presenta el pseudocódigo correspondiente al proceso que calcula la cota superior para la aptitud. Nótese que este proceso funciona indiferentemente de que la función de pesos sea simétrica o no:

Listado 5 procedimiento $CalcularMax$

Entrada: Se tiene acceso a la función de pesos P del ejemplar del VRP.
Salida: Un número *maximo* que se usará como cota superior a la aptitud de los individuos. $maximo = 1 + \sum_{i,j \in [0,n]} P(c_i, c_j)$.

```
1: maximo ← 1;
2: para cada i desde 1 hasta n hacer
3:    para cada j desde 0 hasta n hacer
4:       maximo ← maximo + P(c_i, c_j); // se suma el peso de cada arista
5:    termina para cada
6: termina para cada
7: regresa maximo;
```

El Listado 6 a continuación presenta el pseudocódigo correspondiente al proceso que asigna aptitud a todos los individuos de una población:

Listado 6 procedimiento $EvaluarPoblacion$

Entrada: *poblacion*, arreglo de individuos cuyos elementos srán evaluados.
Salida: todos los elementos de *poblacion* tienen valor de adaptación correcto.

```
1: para cada ind ∈ poblacion hacer
2:    EvaluarIndividuo(ind, maximo); // se calcula la aptitud del individuo
3: termina para cada
```

El Listado 7 a continuación presenta el pseudocódigo correspondiente al proceso de Evaluación:

Listado 7 procedimiento *EvaluarIndividuo*

Entrada: *indiv*,un individuo formado por los arreglos $ruta[1..n]$ y $orden[1..n]$ y el registro *aptitud*; así como el valor de *maximo*.

Salida: el registro *aptitud* de *indiv* contiene el valor de adaptación correcto.

```
1: total ← 0;
2: se crean los arreglos capacidad, distancia y ultimo, todos de tamaño m;
3: se inicializan en 0 todas las entradas de capacidad, distancia y ultimo;
4: para cada j desde 1 hasta n hacer
5:    cliente ← indiv.orden[j]; // el j-ésimo cliente en ser visitado
6:    vehiculo ← indiv.ruta[cliente]; // el vehículo que surte a cliente
7:    distancia[j] ← distancia[j] + P(ultimo[vehiculo], cliente); // se actualiza
      el costo de la ruta
8:    ultimo[vehiculo] ← cliente;
9:    capacidad[vehiculo] ← capacidad[vehiculo] + D(cliente); // se actualiza la
      carga del vehículo
10: termina para cada
11: para cada k desde 1 hasta m hacer
12:    distancia[k] ← distancia[k] + P(ultimo[k], 0); // se suma el costo de
       regresar al depósito para cada ruta
13:    si capacidad[k] > M_D o distancia[k] > M_P entonces
14:       total ← maximo; // solución no factible
15:       interrumpe para cada;
16:    termina si-entonces
17:    total ← total + distancia[k]; // se suma el costo de todas las rutas
18: termina para cada
19: indiv.aptitud ← maximo − total; // actualiza la aptitud de indiv
20: si indiv.aptitud > elite.aptitud entonces
21:    elite.aptitud ← indiv.aptitud; // se encontró un nuevo individuo de élite
22:    para cada i desde 1 hasta n hacer
23:       elite.orden[i] ← indiv.orden[i];
24:       elite.ruta[i] ← indiv.ruta[i];
25:    termina para cada
26: termina si-entonces
```

Rutinas auxiliares:

El Listado 8 a continuación presenta el pseudocódigo correspondiente al proceso que genera los individuos de la población inicial de forma pseudoaleatoria:

Listado 8 procedimiento *GenerarInicial*

Entrada: X, el número de parejas en la población que será generada, al incluir el individuo de élite, el tamaño total de la población será $2X + 1$.

Salida: arreglo de tamaño $2X + 1$ con individuos generados de forma pseudoaleatoria.

```
1: se crea pob[1...2X + 1]; // arreglo que contendrá la población generada
2: para cada i desde 1 hasta 2X + 1 hacer
3:    pob[i] ← nuevo indiviuo; // se genera cada individuo de la población
4:    para cada j desde 1 hasta n hacer
5:       pob[i].ruta[j] ← randInt(n); // se elige una ruta al azar para cada gen del
         primer cromosoma
6:       pob[i].orden[j] ← j; // el segundo cromosoma se ordena de 1 a n
7:    termina para cada
8:    para cada k desde 1 hasta n hacer
9:       pos ← randInt(n); // se "revuelve" el segundo cromosoma
10:      tmp ← pob[i].orden[pos];
11:      pob[i].orden[pos] ← pob[i].orden[k]; // intercambia con un gen aleatorio
12:      pob[i].orden[k] ← tmp;
13:   termina para cada
14: termina para cada
15: regresa pob;
```

En la línea 6 se hace que el cromosoma de orden esté ordenado de menor a mayor, esto para garantizar que se tiene una permutación válida al iniciar. Luego, en la segunda iteración sobre los genes, se intercambia aleatoriamente el orden con el fin de obtener una permutación aleatoria.

El Listado 9 a continuación presenta el pseudocódigo correspondiente a la rutina auxiliar que muestra la información codificada en un individuo de forma legible para un humano:

Listado 9 procedimiento *ImprimirIndividuo*

Entrada: *ind*, el individuo cuyo conjunto de rutas equivalente será mostrado en pantalla.

Salida: muestra en pantalla la ruta para cada vehículo codificada en *ind*.

```
1: para cada i desde 1 hasta m hacer
2:    imprime "v_i : c_0"; // i-ésima ruta, inicia en el depósito
3:    para cada j desde 1 hasta n hacer
4:       c ← ind.orden[j]; // j-ésimo cliente visitado
5:       si ind.ruta[c] = i entonces
6:          imprime "→ c_j"; // solo se imprimen clientes en la ruta i
7:       termina si-entonces
8:    termina para cada
9:    imprime "→ c_0"; // la ruta i termina en el depósito
10: termina para cada
```

4 Variantes consideradas

En este Capítulo se presentan algunas variantes consideradas para el Algoritmo Genético propuesto y se describe cuáles fueron utilizadas, cuáles fueron rechazadas y por qué.

4.1 Algoritmo genético base

Aunque ya existen en la literatura varias propuestas de heurísticas que abordan el VRP, la mayoría de ellas utiliza el enfoque de "primero agrupar y segundo enrutar" ("Cluster first, route second" approach), en el cual se divide al VRP en dos problemas: primero se asignan entregas a vehículos, tomando en cuenta únicamente la capacidad de carga del vehículo y la carga requerida por cada entrega; luego se determina el orden en el que cada vehículo realizará las entregas que se le asignaron, minimizando solamente la distancia total recorrida [3].

La desventaja del enfoque "primero agrupar y segundo enrutar" es que, al considerarse solamente las demandas en la primera etapa, puede darse un caso en el cual se asignen al mismo vehículo dos o más entregas que se encuentran muy alejadas entre sí, aún cuando haya otras entregas con peso similar pero más cercanas, lo cual puede llevar a obtener soluciones de menor calidad al concluir la segunda etapa.

El algoritmo genético propuesto en este trabajo tiene por objetivo evitar ese problema, ya que al evolucionar simultáneamente la asignación de rutas y el orden de visita, se compara la calidad de soluciones completas, con lo cual se pretende obtener mejores resultados.

Sin embargo, resultados experimentales preliminares indicaron que se requieren demasiadas generaciones para que se alcancen resultados de calidad comparable a otros métodos de la literatura, a saber, un número de generaciones del orden de decenas de millones, o incluso cientos de millones, según el ejemplar específico y los parámetros de la heurística, lo cual se traduce en tiempos de cómputo demasiado grandes para que el método sea viable.

Por ello se realizaron pruebas con algunas variantes del AG, con el objetivo de incrementar la calidad de las soluciones obtenidas. Dichas variantes se describen a continuación.

4.2 Algoritmo genético anidado

Una causa probable del bajo rendimiento del AG base es que, al evolucionar de forma independiente los dos cromosomas, se tiene que para lograr obtener una mejor solución es necesario que ambos cromosomas cambien de manera tal que sigan siendo "compatibles", mientras que si uno de los cromosomas mejora pero el otro empeora, no se obtendrá un buen valor de aptitud y los cambios positivos del primer cromosoma probablemente no pasen a la siguiente generación.

Para evitar esta situación, se intentó modificar el AG base para que se evolucione uno de los cromosomas en el ciclo principal y el otro cromosoma sea optimizado considerando el primero como fijo; es decir, tratar de obtener la mejor configuración del segundo cromosoma dado el primer cromosoma.

Así, la primera variante considerada realiza la optimización del segundo cromosoma como parte del proceso de evaluación del primer cromosoma. Dicha optimización se realizó inicialmente por medio de otro algoritmo genético, de ahí la denominación de "anidado".

Aunque se observó consistentemente una ligera mejora en la calidad de las soluciones del AG anidado con respecto a los resultados del AG base, el aumento en tiempo de cómputo fue desproporcionalmente mayor, siendo el AG anidado alrededor de 400 veces más tardado en pruebas iniciales, por lo cual esta variante fue descartada.

4.3 Algoritmo genético con evaluación aproximada

Esta variante sigue la misma idea que el AG anidado: el cruzamiento y la mutación operan solamente sobre el primer cromosoma, el que determina la asignación de rutas y los valores del segundo cromosoma se optimizan como parte de la evaluación.

La idea es emplear un método más rápido para aproximar el valor óptimo del segundo cromosoma, en este caso se utiliza el método del vecino más cercano o NN (por sus siglas en inglés, *Nearest Neighbor*):
El segundo cromosoma se divide en rutas según la asignación dada por el primer cromosoma, luego se procede con cada ruta por separado, comenzando en el depósito y asignando como siguiente cliente a visitar al que tiene la mínima distancia del que fue agregado en el paso anterior.

Esta variante dio buenos resultados en pruebas preliminares, arrojando resultados de

calidad significativamente mejor que el AG base, mientras el tiempo de ejecución se mantuvo entre 15 y 40 veces más que el AG base, así que esta variante fue empleada en las pruebas finales.

También se realizaron pruebas en las que las rutas eran calculadas paralelamente en distintos hilos de ejecución, uno por cada ruta, pero en la práctica el tiempo de ejecución aumentó significativamente en lugar de disminuir, por lo cual no se conservó esa modificación.

4.4 Algoritmo genético con evaluación exacta

Otra variante considerada toma como base el AG con evaluación aproximada e intenta optimizar aún más el orden de visita, con el fin de que, al evolucionar una determinada asignación de rutas, el valor de adaptación del individuo sea el óptimo para esa asignación.

Para ello, se emplea un método de Ramificación y Poda o BB (por las siglas en inglés de *Branch and Bound*), en el cual se define una cota máxima a la distancia total de una ruta y luego se realiza una exploración exhaustiva de las permutaciones, descartando las soluciones en las cuales la distancia de una subruta sobrepasa la cota, con lo cual se evita realizar un gran número de evaluaciones.

Así, se toma como cota inicial el valor aproximado obtenido del método NN y se exploran las permutaciones de la ruta, podando aquellas con mayor costo y actualizando la cota si se encuentra una mejor asignación.

En pruebas preliminares, esta variante alcanzó muy buenos resultados, llegando en dos ocasiones a arrojar el mejor valor conocido para ejemplares pequeños. Sin embargo, la cantidad de soluciones omitidas por el método BB fue muy pequeña en relación al número total de soluciones posibles, por lo que el tiempo de ejecución se incrementó aún más que para el AG anidado, llegando a más de 700 veces el tiempo del AG base.

Al intentar reducir el número de generaciones o el tamaño de la población para que el tiempo de ejecución fuera comparable al de las otras variantes, la calidad de las soluciones se vio afectada drásticamente, llegando incluso a dar resultados peores que los del AG base, por lo cual esta variante no fue considerada para las pruebas finales.

4.5 Algoritmo genético con refinamiento de soluciones

La última variante intenta solucionar el problema del AG con evaluación exacta por medio de una reducción en el número de individuos para los cuales se aplica el método BB.

En lugar de llevar a cabo la exploración exhaustiva en cada nuevo individuo generado, se realiza dicho proceso solamente para los individuos prometedores, es decir, aquellos que, evaluados con el método NN, tuvieran aptitud entre las mejores 5 producidas a lo largo de toda la ejecución.

Sin embargo, con esto no se logró una reducción suficiente en el número de evaluaciones con el método BB, ya que al inicio del AG es común que se encuentren muchas soluciones que superan la aptitud de aquellas producidas aleatoriamente para la población inicial.

Debido a ello, el tiempo necesario para esta variante era aún demasiado e igual que en el caso anterior, al intentar reducir el número de generaciones o el tamaño de la población, la calidad de las soluciones bajó hasta ser mucho menor que las obtenidas con otras variantes.

Para solucionar esto, se empleó otra estrategia en la cual al inicio se usaba la variante con evaluación aproximada usando el método NN y una vez transcurrido un número de generaciones se comenzaba a usar el método BB para los ejemplares prometedores.

La idea es que, después de transcurridas varias generaciones, la población ha evolucionado lo suficiente para que sea relativamente poco común encontrar mejoras, de modo que la cantidad de individuos prometedores es lo suficientemente baja como para emplear el método BB sobre los mismos sin llevar el tiempo de ejecución de todo el AG a niveles imprácticos.

Esta variante dió muy buenos resultados, mejorando la calidad de las soluciones en comparación con la variante de evaluación aproximada, pero manteniendo el tiempo de ejecución por debajo de 4 veces el empleado para esa otra variante.

Sin embargo, al examinar más detenidamente los resultados, se encontró que se obtiene un resultado aún mejor si se toma el individuo generado como respuesta por la variante de evaluación aproximada y se aplica el método BB para refinar solamente esa propuesta de solución.

Así, se determinó que la última variante considerada para las pruebas finales consiste en refinar la mejor solución arrojada por las otras variantes, midiendo el tiempo adicional necesario.

4.6 Penalización por número de vehículos

Otro problema encontrado en las heurísticas propuestas en la literatura es que la mayoría de ellas, al procesar primero el agrupamiento de los clientes, dan implícitamente prioridad a minimizar el número de vehículos por encima de minimizar la distancia total.

En algunos casos, esto puede llevar a soluciones subóptimas desde el punto de vista del costo. Como ejemplo de ello,la Figura 11 muestra un ejemplar en el cual, si la capacidad de carga es al menos cuatro, entonces un solo vehículo tiene capacidad suficiente para realizar todas las entregas, por ejemplo siguiendo la ruta resaltada en rojo del lado izquierdo; sin embargo, utilizando dos vehículos se puede lograr mucho menor costo, ilustrado del aldo derecho con las rutas resaltadas en verde y azul.

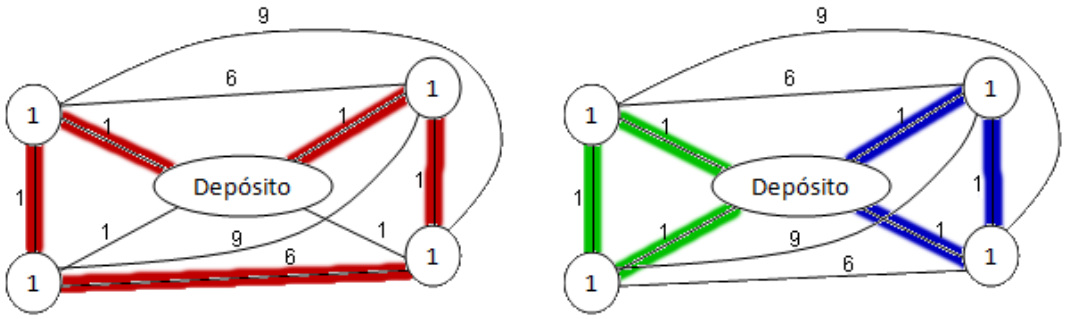

Figura 11: Ejemplar para el cual dos rutas cuestan menos que una.

Para evitar este problema, sin caer en el extremo opuesto donde se da prioridad solo a minimizar la distancia sin considerar el número de vehículos, se agregó a la implementación del AG propuesto un parámetro de penalización por número de vehículos.

Esto significa simplemente que se asigna un valor de penalización, el cual se suma al costo total por cada vehículo utilizado.
Este parámetro permite calibrar qué tanta importancia se le da a cada uno de los

valores que se quiere minimizar.

Las variantes del AG antes definidas fueron probadas tanto con penalización como sin ella.

5 Resultados experimentales

A continuación se presenta la metodología empleada para las pruebas experimentales, así como los resultados obtenidos y las conclusiones del trabajo.

5.1 Metodología empleada

Como se mencionó en el Capítulo 4, los métodos empleados en las pruebas finales fueron el Algoritmo Genético Base, el cual será denotado como AGB en el resto de este capítulo, el Algoritmo Genético con evaluación Aproximada, que se denotará por AGA, y el proceso de evaluación exacta para refinar el resultado del AGA, el cual se denotará por R.

Las tres variantes también fueron probadas en su versión con penalización por número de vehículos. Los resultados de esas ejecuciones se denotan por AGB_p, AGA_p y R_p.

Las pruebas fueron implementadas en el lenguaje Java, y los programas fueron ejecutados en una computadora HP Pavilion con un procesador de doble núcleo, con el ambiente de tiempo de ejecución de Java (JRE) en su versión 8.

Para realizar las pruebas, se usó el conjunto de ejemplares de prueba o *benchmarks* propuesto por Christofides et al. (1979) [1], el cual se detalla en la siguiente sección, y los resultados fueron comparados con los publicados en Laporte et al. (2000) [5].

Para determinar el número máximo de vehículos para cada ejemplar, se realizaron pruebas usando el AGA con un número reducido de generaciones y aumentando gradualmente la cantidad de vehículos hasta que se encontró una solución factible, después de lo cual se incrementó un 20% más para aumentar el espacio de soluciones factibles, con el fin de permitir posibles mejoras en la calidad de las soluciones encontradas.

5.2 Ejemplares de prueba

En esta sección se detalla el conjunto de ejemplares de prueba propuesto por Christofides et al. (1979) [1]:

El conjunto consiste de 14 ejemplares, 7 de ellos tienen un límite de costo por ruta y los otros 7 limitan solamente la capacidad de carga.

En todos los ejemplares, se define a cada cliente por sus coordenadas en un plano cartesiano y su demanda, tomando como función de costo la distancia euclidiana entre los clientes.
El depósito también se define por sus coordenadas en el plano y su distancia a cada cliente también se define como la distancia euclidiana.

Para este conjunto de ejemplares, la función de costo representa tiempo de recorrido y los ejemplares con límite de costo también definen un tiempo de servicio, el cual es constante para todos los clientes.
Dicho costo se implementa simplemente sumando el valor constante a la distancia entre cada par de clientes y a la distancia desde el depósito a cada cliente, pero no desde un cliente hacia el depósito.

A continuación se presentan de forma gráfica los 14 ejemplares. El nodo rojo es el depósito y cada uno de los nodos azules es un cliente, cuya demanda se muestra cerca del nodo.
Nótese que algunos ejemplares tienen varios nodos con las mismas coordenadas, por lo que algunos nodos tienen más de un valor asociado; dichos nodos se marcan en verde.

Los datos generales del ejemplar tales como capacidad de carga de los vehículos o límite de costo por ruta se indican del lado derecho de cada figura.

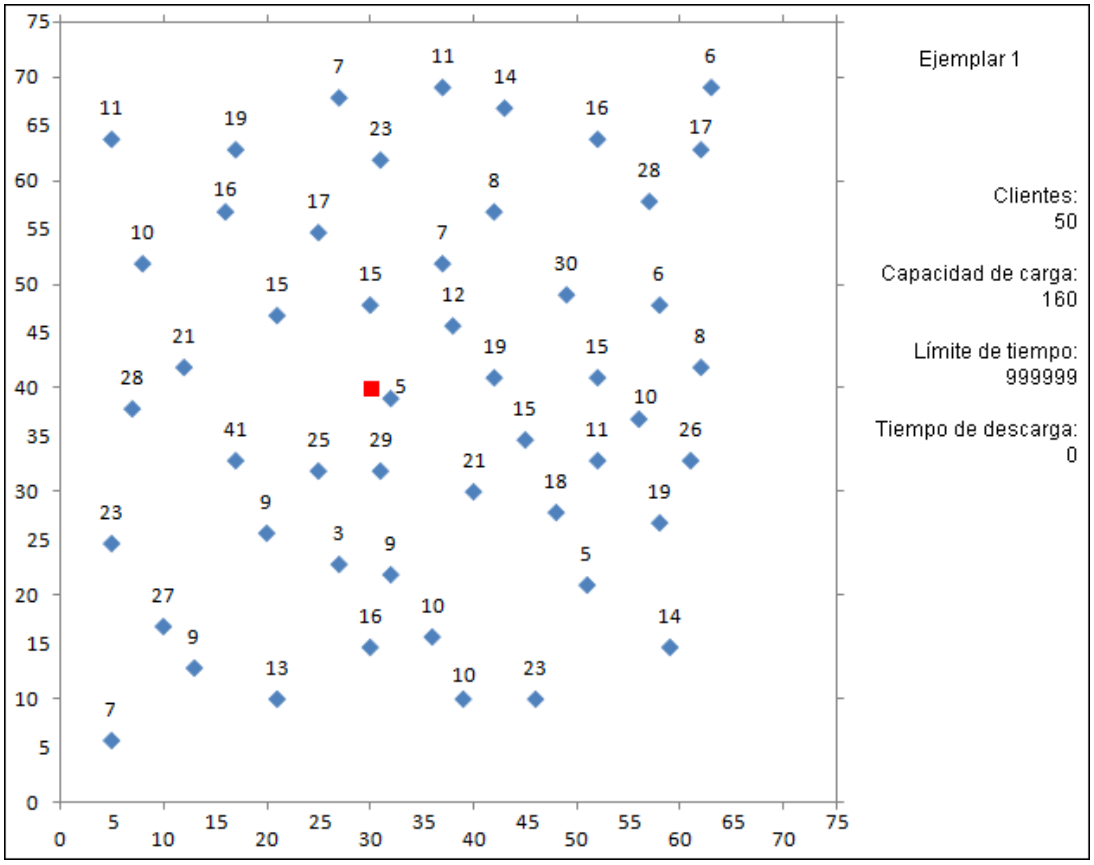

Figura 12: Ejemplar 1 del conjunto de pruebas.

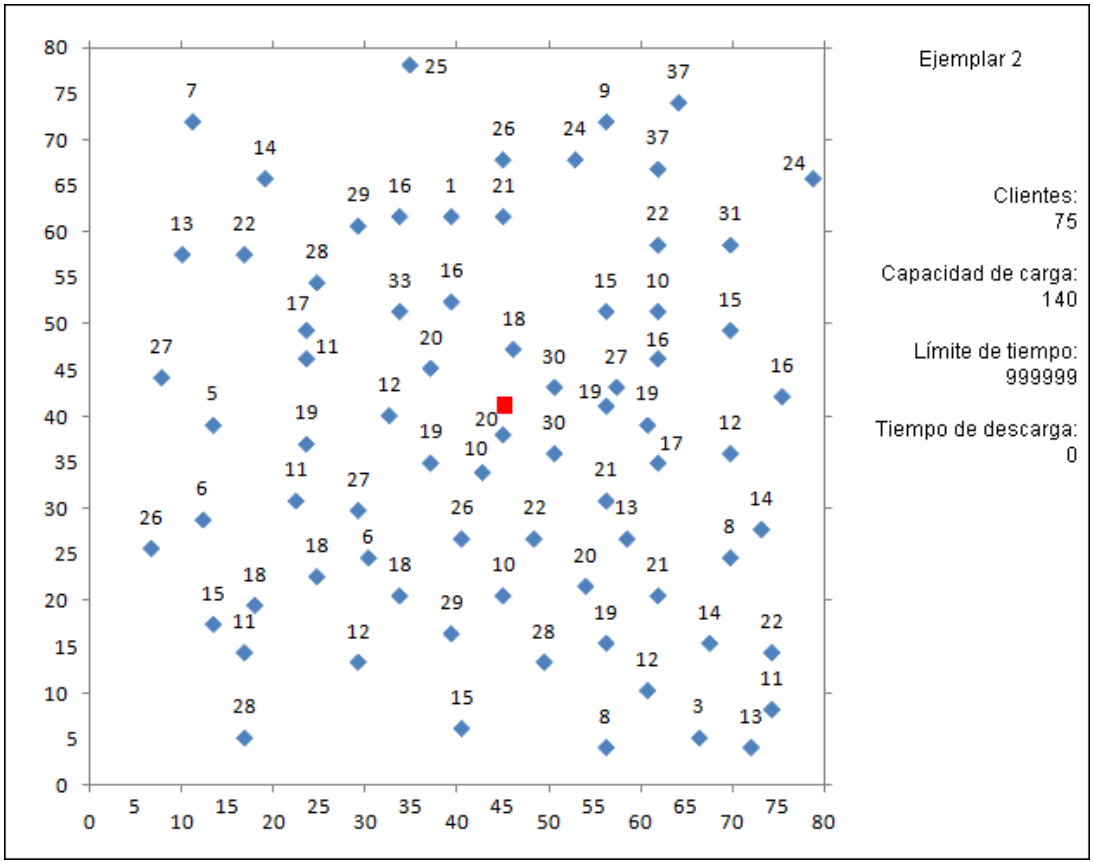

Figura 13: Ejemplar 2 del conjunto de pruebas.

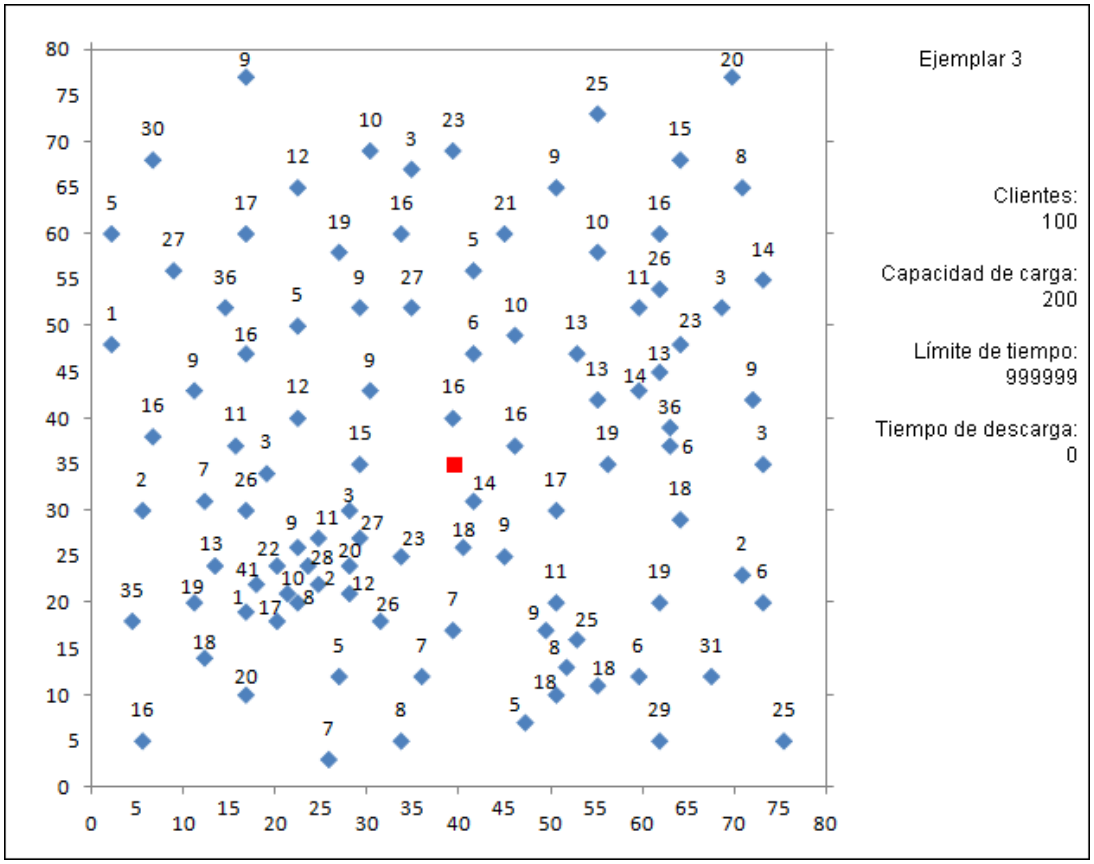

Figura 14: Ejemplar 3 del conjunto de pruebas.

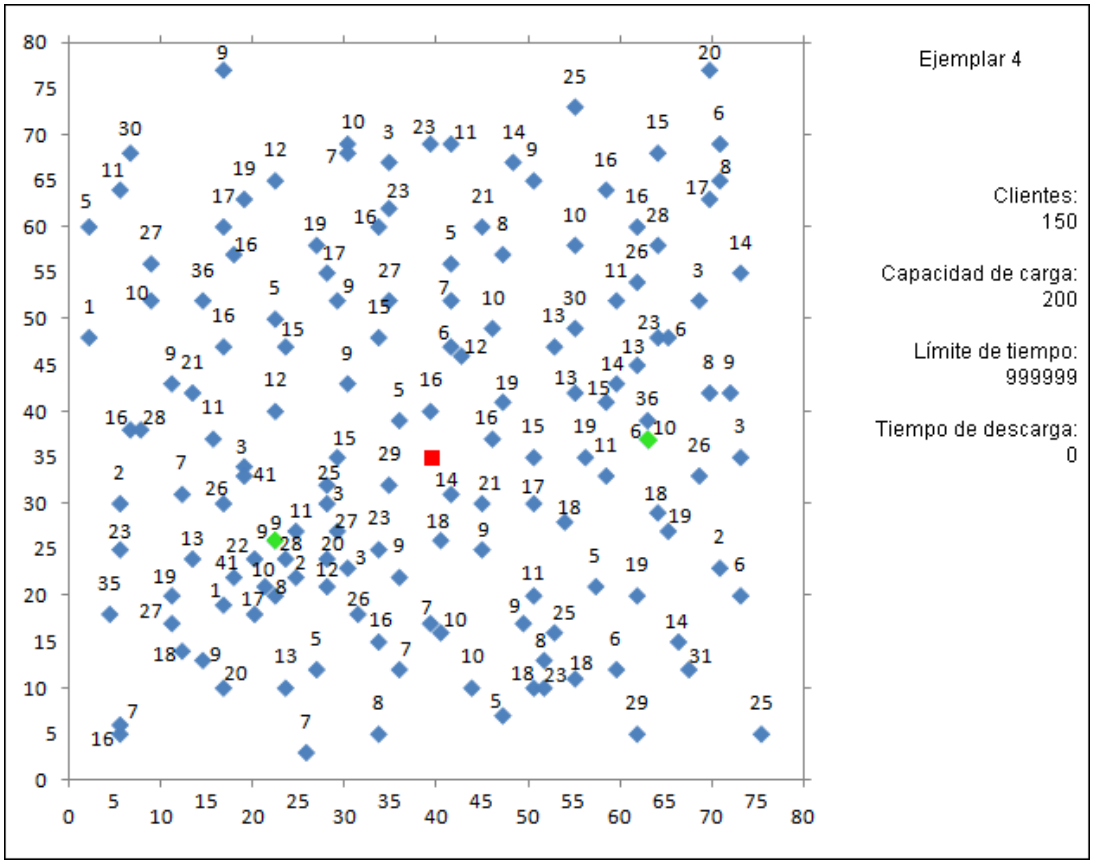

Figura 15: Ejemplar 4 del conjunto de pruebas.

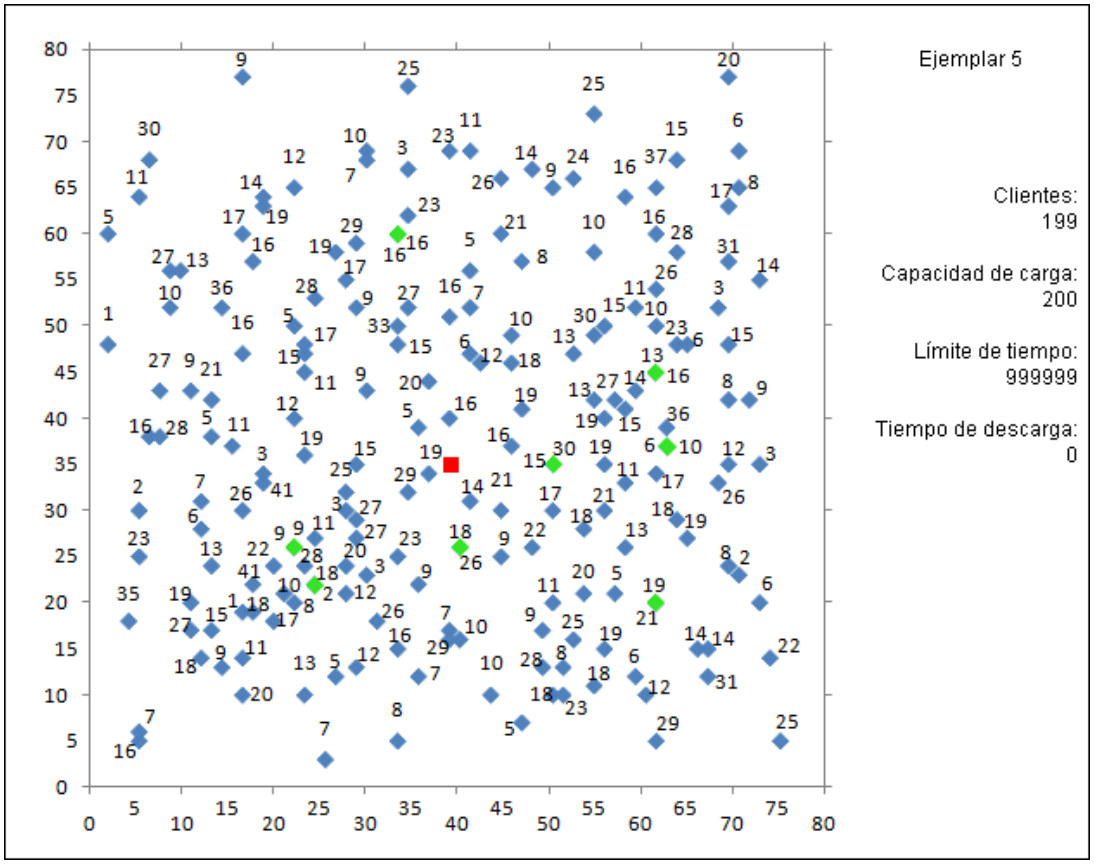

Figura 16: Ejemplar 5 del conjunto de pruebas.

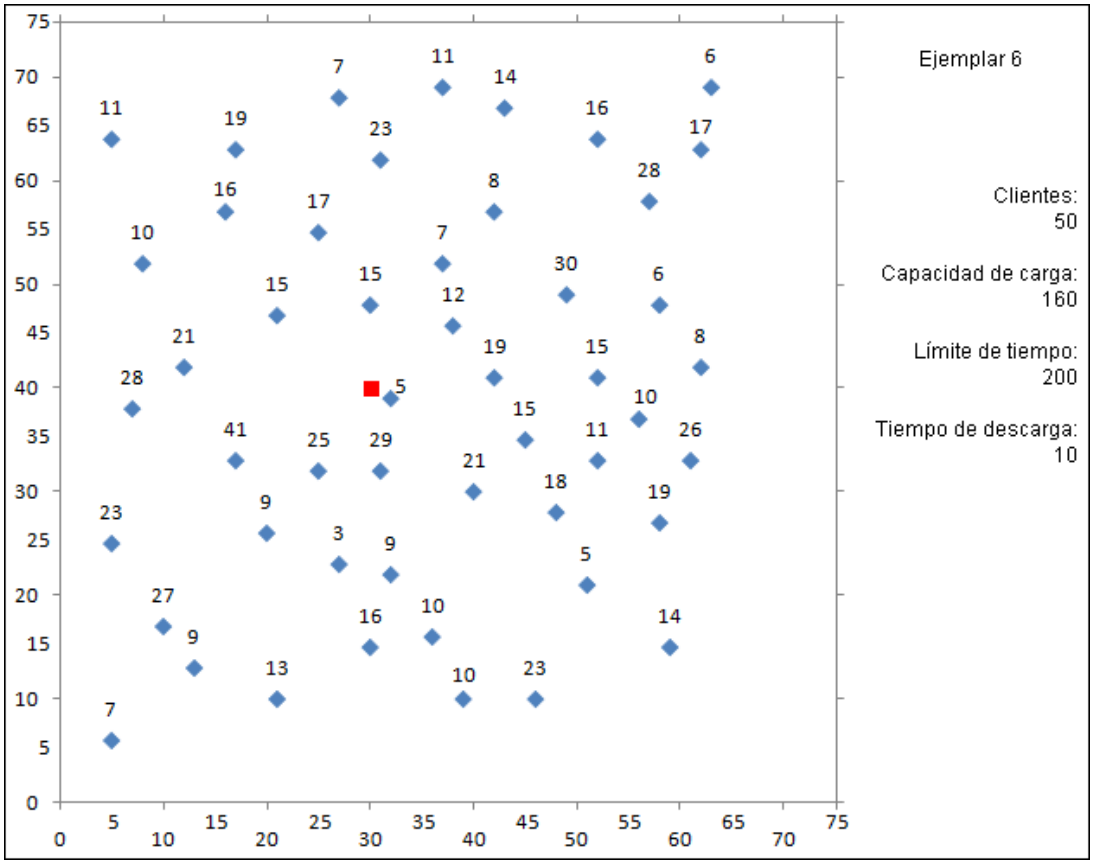

Figura 17: Ejemplar 6 del conjunto de pruebas.

El ejemplar 6 se compone de los mismos clientes y depósito que el ejemplar 1, pero agrega restricciones de tiempo.

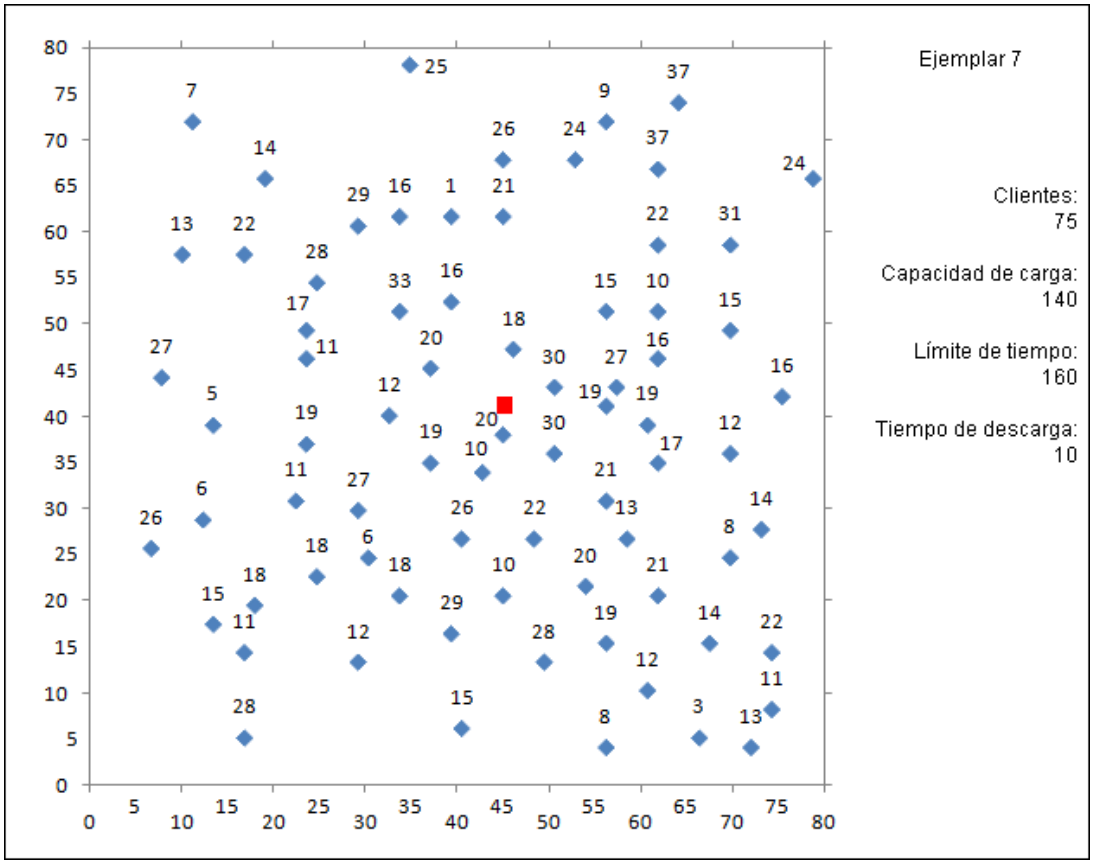

Figura 18: Ejemplar 7 del conjunto de pruebas.

El ejemplar 7 se compone de los mismos clientes y depósito que el ejemplar 2, pero agrega restricciones de tiempo.

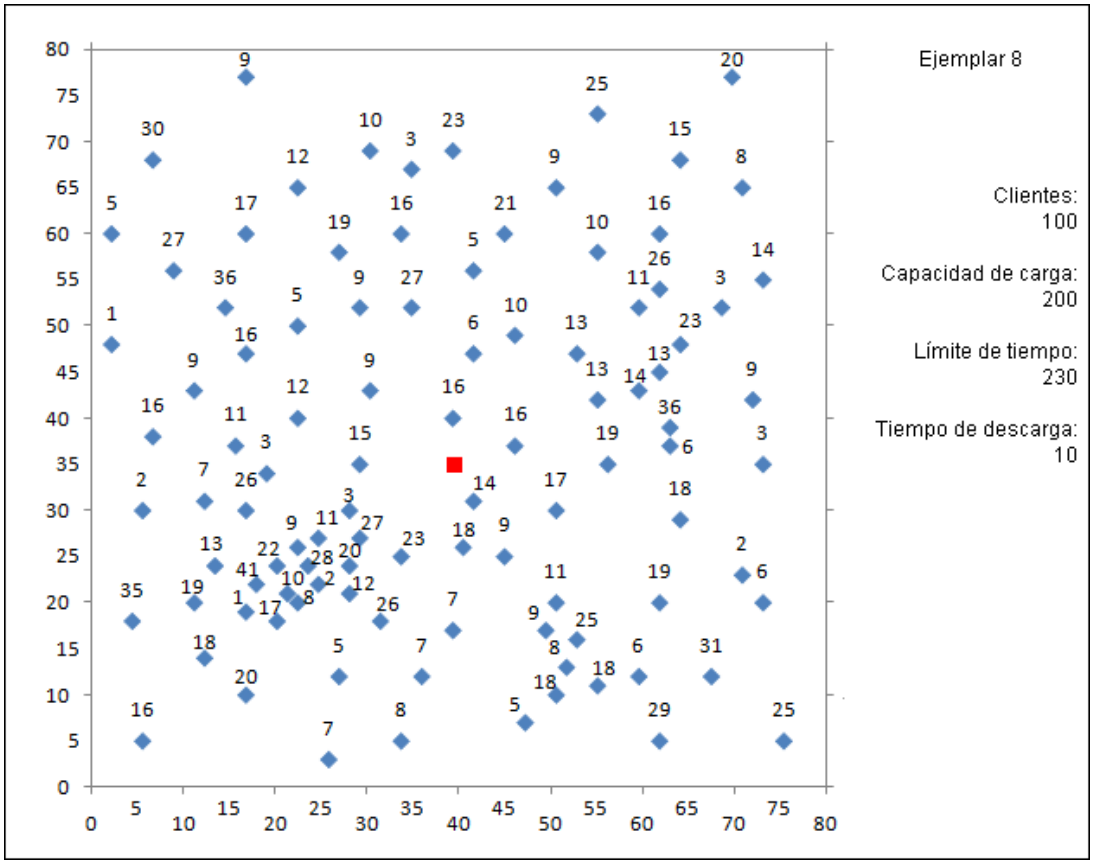

Figura 19: Ejemplar 8 del conjunto de pruebas.

El ejemplar 8 se compone de los mismos clientes y depósito que el ejemplar 3, pero agrega restricciones de tiempo.

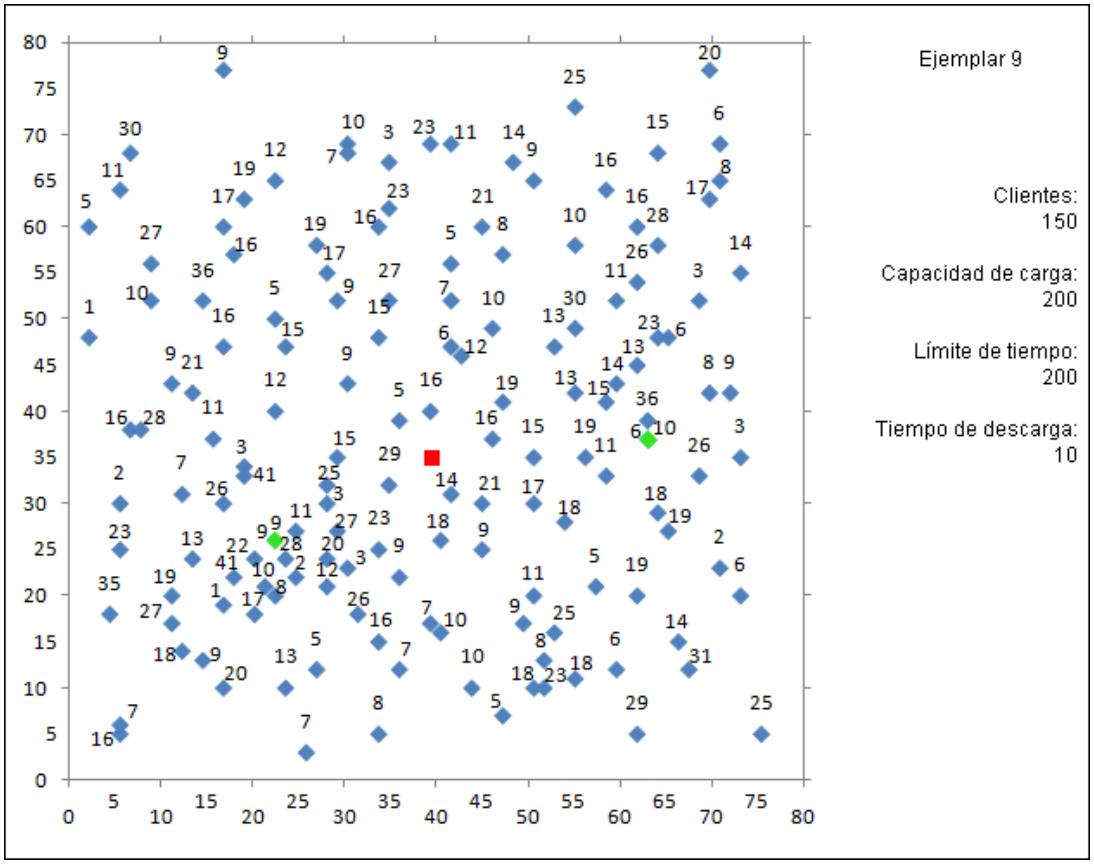

Figura 20: Ejemplar 9 del conjunto de pruebas.

El ejemplar 9 se compone de los mismos clientes y depósito que el ejemplar 4, pero agrega restricciones de tiempo.

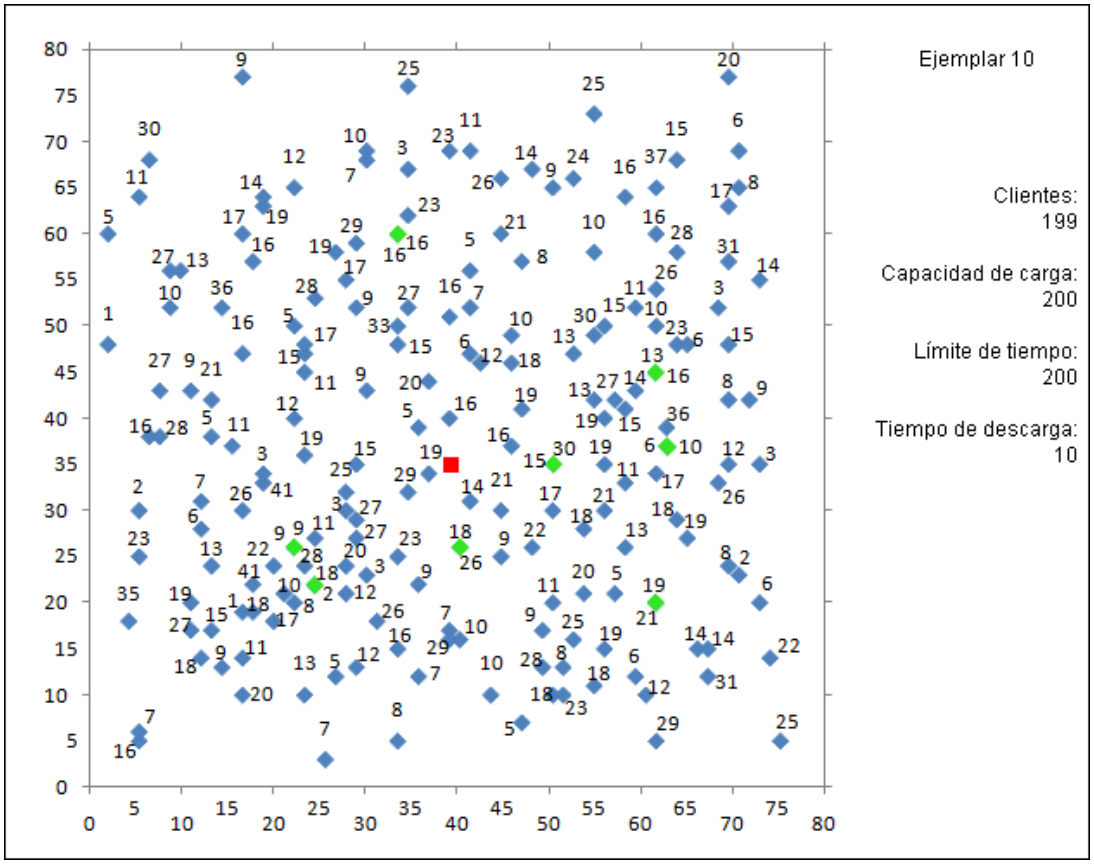

Figura 21: Ejemplar 10 del conjunto de pruebas.

El ejemplar 10 se compone de los mismos clientes y depósito que el ejemplar 5, pero agrega restricciones de tiempo.

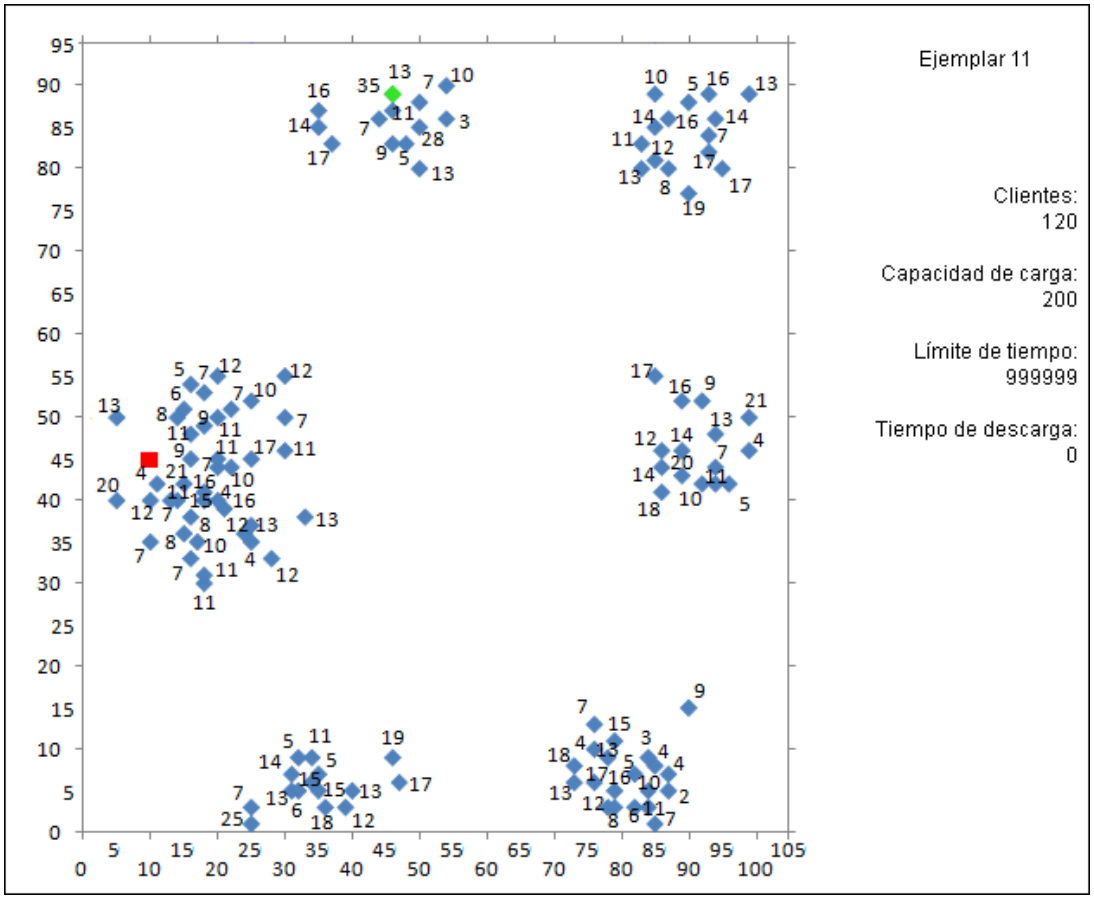

Figura 22: Ejemplar 11 del conjunto de pruebas.

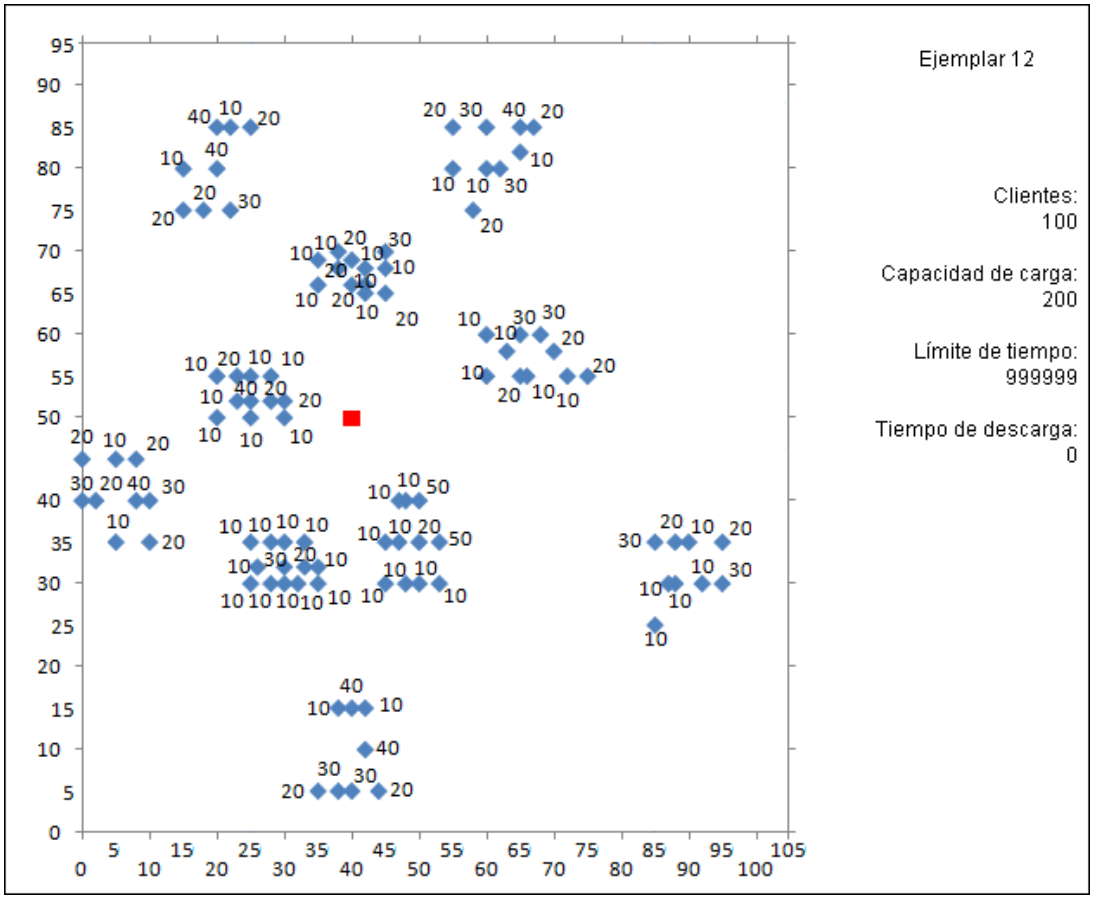

Figura 23: Ejemplar 12 del conjunto de pruebas.

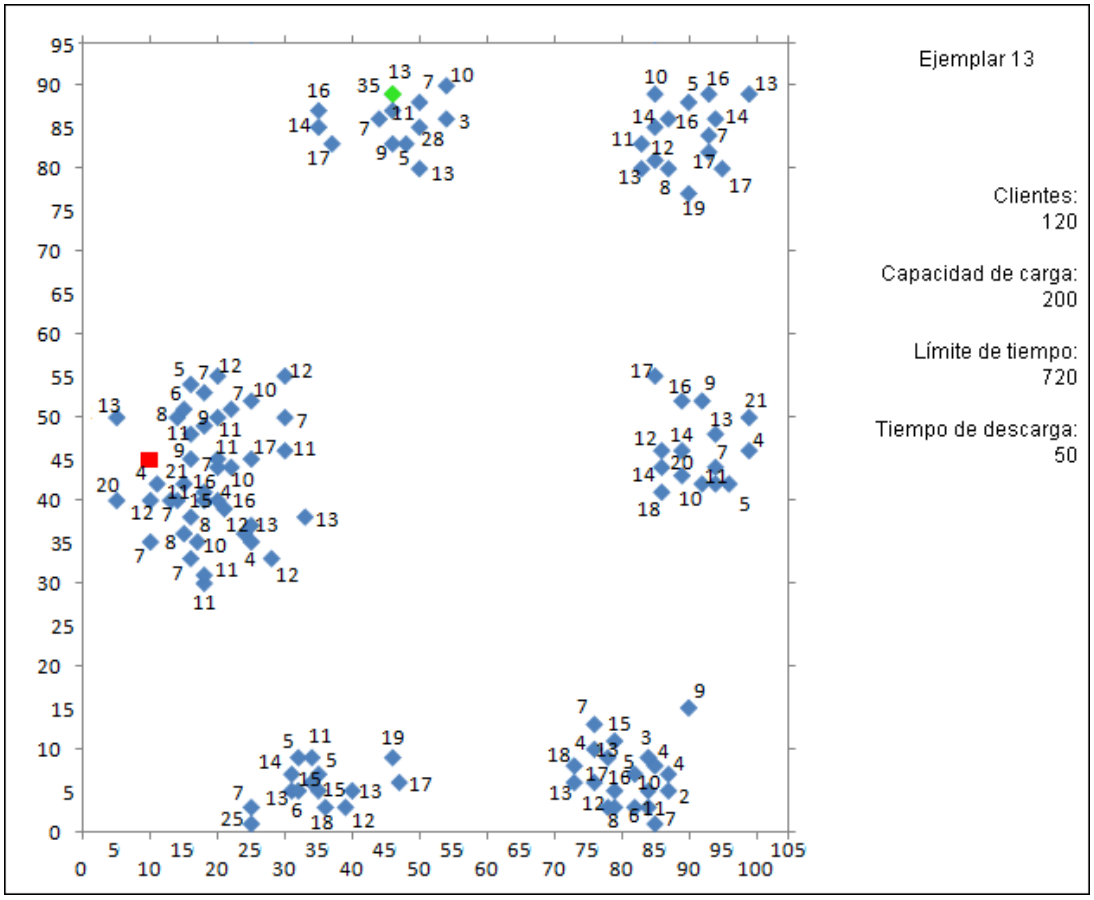

Figura 24: Ejemplar 13 del conjunto de pruebas.

El ejemplar 13 se compone de los mismos clientes y depósito que el ejemplar 11, pero agrega restricciones de tiempo.

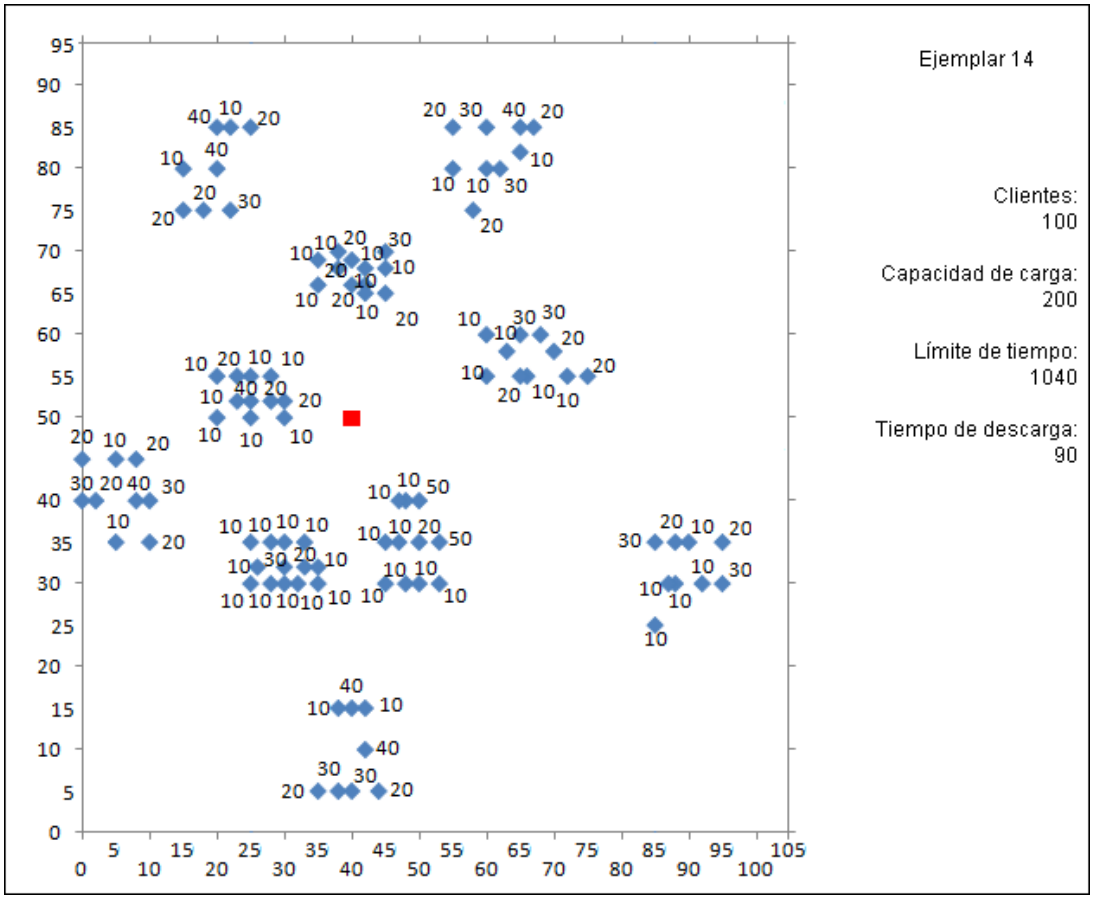

Figura 25: Ejemplar 14 del conjunto de pruebas.

El ejemplar 14 se compone de los mismos clientes y depósito que el ejemplar 12, pero agrega restricciones de tiempo.

5.3 Resultados experimentales

Como se mencionó en la Sección 5.1, se realizaron pruebas de los métodos AGB y AGA, así como el refinamiento de la solución del AGA, denotado por R, sobre el conjunto de ejemplares de Christofides *et al.* (1979) y los resultados se compararon con los presentados por Laporte*et al.* (2000).

Los resultados mostrados en la Figura **??** a continuación corresponden a la ejecución de las tres variantes con una población de tamaño 25, incluyendo el individuo de élite, trabajando por 1,000,000 de generaciones.

El valor *costo* indica el peso total de la solución, m es el número de rutas empleadas y t es el tiempo de ejecución para la prueba, medido en segundos.

Los tiempos para R se toman incluyendo el tiempo para generar la solución del AGA correspondiente.

Los valores resaltados en negritas indican la mejor solución. Cuando el refinamiento no produjo mejoras debido a que el orden de visita ya era óptimo, se toma la solución antes del proceso de refinamiento como la mejor.

La columna L+ indica el valor más alto de los mostrados por Laporte et al. y la columna L- corresponde al valor más bajo tomado de la misma fuente.

Para ambas columnas, el valor de t es el correspondiente al costo mostrado, convertido a segundos a partir del indicado en el artículo, el cual está dado en minutos.

Se omite el número de vehículos correspondiente a L+ y L- debido a que el artículo no lo especifica.

La columna M muestra el mejor valor conocido, también de acuerdo al artículo de Laporte *et al.*

	AGB			AGA			R			L+		L-		M
#	costo	m	t	costo	m	t	costo	m	t	costo	t	costo	t	
1	700.29	6	38	594.94	6	143	**576.61**	6	144	524.61	49	524.61	49	524.61
2	1177.00	14	53	928.37	11	201	**926.31**	11	202	838.60	133	835.26	2928	835.26
3	1438.06	12	69	1003.73	9	338	**1002.61**	9	340	829.45	1104	826.14	1104	826.14
4	2076.62	18	102	1208.18	14	525	**1207.31**	14	530	1044.35	271	1028.42	3528	1028.42
5	3302.39	24	136	1717.99	20	699	**1717.87**	20	983	1334.55	450	1298.58	16351	1291.45
6	1183.14	8	39	1146.10	7	136	**1142.33**	7	137	555.43	52	555.43	52	555.43
7	2051.45	17	67	1832.03	12	222	**1818.16**	12	223	965.62	6128	909.68	165	909.68
8	2682.02	17	84	2286.35	16	300	**2276.54**	16	301	881.38	5889	865.94	1536	865.94
9	4220.01	28	134	3731.82	26	520	**3713.29**	26	521	1173.12	340	1162.55	4260	1162.55
10	5544.10	35	155	**4923.91**	33	741	4923.91	33	742	1439.29	22102	1397.94	5988	1395.85
11	2260.13	12	100	1375.34	8	462	**1372.90**	8	3055	1073.47	1332	1042.11	191	1042.11
12	1617.17	12	71	952.78	10	309	**948.57**	10	322	819.56	66	819.56	66	819.56
13	9610.97	18	97	7879.27	12	377	**7876.19**	12	381	1618.55	12105	1541.14	3552	1541.14
14	10489.74	15	71	**10035.29**	12	299	10035.29	12	300	915.24	9179	866.37	85	866.37

La Figura **??** a continuación muestra los resultados de las pruebas con penalización por número de vehículos, siguiendo el mismo formato que la tabla anterior.

Los valores en cursivas indican un mejor resultado que el obtenido para el mismo ejemplar en las pruebas sin penalización por número de vehículos.

La columna de mejor variante indica con cuál variante del algoritmo se obtuvo el mínimo costo total, sin importar el número de vehículos.

	AGB_P			AGA_P			R_P			Mejor
#	costo	m	t	costo	m	t	costo	m	t	variante
1	686.80	6	38	611.65	5	153	**608.37**	5	154	R
2	1056.12	13	55	**953.35**	11	212	953.35	11	213	R
3	1303.08	11	71	962.52	8	365	***954.86***	8	406	R_P
4	2372.23	17	105	1281.59	12	561	**1279.76**	12	1112	R
5	3249.51	23	138	1720.43	18	737	***1716.15***	18	804	R_P
6	1251.96	8	44	1137.05	7	151	***1131.62***	7	152	R_P
7	***1788.11***	14	68	1879.82	11	242	1879.82	11	243	AGB_P
8	2616.35	17	90	2270.42	15	328	***2269.17***	15	329	R_P
9	4096.84	29	141	3683.00	24	563	***3676.83***	24	564	R_P
10	6168.40	35	156	***4648.66***	30	815	4648.66	30	816	AGA_P
11	2526.24	12	84	1382.33	7	435	**1376.77**	7	1357	R
12	1362.74	12	73	1006.67	10	325	**1004.46**	10	327	R
13	8927.40	16	100	**8208.54**	13	390	8208.54	13	391	R
14	10445.11	15	75	**10052.27**	11	316	10052.27	11	317	AGA

Conclusiones

Aunque en general los resultados son pobres en comparación con otros experimentos en la literatura, nos permiten realizar las siguientes observaciones.

El proceso de refinamiento produjo mejoras en la calidad de las soluciones en casi todos los casos, pero el tiempo adicional necesario para efectuarlo es muy bajo, sobre todo si se considera que el trabajo necesario para realizar la exploración exhaustiva no incrementa si se modifican los parámetros del AG, por ejemplo aumentando el tamaño de la población o el número de generaciones.

De hecho, mientras mejor sea la solución que se va a refinar, también será más justa la cota base para el método BB, por lo cual es probable que se descarten más soluciones y el proceso termine más rápido.

También se observa que la versión con penalización por número de vehículos, que tiene por objeto favorecer las soluciones con menos rutas, solamente obtuvo mejores resultados en la mitad de las pruebas, incluso cuando algunas de ellas sí utilizan menor cantidad de vehículos.

De hecho, la solución encontrada por el AGA para los ejemplares 1, 4, 11 y 13 utiliza más rutas que la solución del AGA_p, pero su costo total es menor.

Por lo tanto, en casos para los cuales es más importante minimizar el costo total de las rutas que la cantidad de rutas, valdría la pena explorar la noción de permitir emplear más rutas y combinarla con otras heurísticas para lograr aún mejores resultados.

Referencias

[1] N. Christofides, A. Mingozzi, P. Toth, and C. Sandi. *Combinatorial optimization.*

[2] N. De Jaegere, M. Defraeye, and I. Van Nieuwenhyuse. The vehicle routing problem: State of the art classification and review.

[3] R. M. Jorgensen, J. Larsen, and K. B. Bergvinsdottir. Solving the dial-a-ride problem using genetic algorithms.

[4] A. Kuri and J. Galaviz. *Algoritmos Genéticos.*

[5] G. Laporte, M. Gendreau, J. Y. Potvin, and F. Semet. Classical and modern heuristics for the vehicle routing problem.

[6] C. R. Reeves. *Modern Heuristic Techniques for Combinatorial Problems.*

[7] D. Toth, P.and Vigo. *The Vehicle Routing Problem.*