Universidad de La Habana
Facultad de Matemática y Computación

Una propuesta para la evaluación automática de soluciones vecinas en un Problema de Enrutamiento de Vehículos a partir del grafo de evaluación de una solución

Autor: **José Jorge Rodríguez Salgado**

Tutor: **MsC. Fernando Raúl Rodríguez Flores**

Trabajo de Diploma
presentado en opción al título de
Licenciado en Ciencia de la Computación

Junio de 2020

A mis padres...

Agradecimientos

El primer agradecimiento va para mi tutor, Fernando Rodríguez, por ser la persona que más ha contribuido con el presente trabajo. Le agradezco su valiosísimo tiempo y sus buenas ideas.

Agradezco a mi madre, mi primera profesora y guía. A mi padre por sus consejos y el ejemplo. A toda la familia que me hace sentirme tan afortunado cada día.

A la otra familia que me he encontrado también por gran fortuna, mis amigos.

Agradezco a todos los profesores de MATCOM que me impartieron clases durante estos cinco años. De todos me llevo algo.

Opinión del tutor

El Problema de Enrutamiento de Vehículos (VRP) ha sido estudiado durante más de 50 años y en la literatura se pueden encontrar más de 30 variantes, además de las que están apareciendo constantemente. Resolver una sola de estas variantes es un proceso que requiere tiempo y esfuerzo por parte de los investigadores.

Con los resultados de esta tesis será posible resolver variantes de este problema en mucho menos tiempo, usando Algoritmos de Búsqueda Local, pues no será necesario dedicar tiempo a programar la evaluación de soluciones vecinas a partir de la actual, ya que esto se podrá hacer de manera automática y eficiente.

Este trabajo, al igual que los demás en los que he participado con el diplomante a lo largo de varios años, demuestra que José Jorge Rodríguez Salgado es capaz de utilizar los conocimientos adquiridos durante la carrera de una manera creativa, de identificar y solucionar problemas que aparezcan durante el desarrollo de la investigación y de adquirir de manera autodidacta los conocimientos y habilidades para ello.

En otras palabras, creo que estamos en presencia de un trabajo excelente que demuestra que José Jorge reúne todos los requisitos para desempeñarse como un excelente Científico de la Computación.

MSc. Fernando Raul Rodríguez Flores
Facultad de Matemática y Computación
Universidad de la Habana
Mayo, 2020

Resumen

En este trabajo se propone un método que permite obtener el costo de soluciones vecinas de una solución dada en un Problema de Enrutamiento de Vehículos (VRP) cualquiera, sin tener que implementar un algoritmo para ello. Lo único que se necesita es el código de evaluación de una solución dada para la variante del VRP deseada. El costo de cada solución vecina se calcula de manera eficiente en el sentido de que garantiza que solo se evalúan aquellos fragmentos de la nueva solución en los que esta se diferencia de la solución inicial. El método propuesto permite un ahorro de tiempo en la implementación de la solución del problema debido a que el usuario no tiene que escribir el código para la evaluación eficiente de las soluciones vecinas.

Abstract

In this work we propose a method that allows the user to compute the cost of any neighbor solution of a given one in a Vehicle Routing Problem (VRP), without the need of implementing an algorithm for that purpose. The only requirement is the source code for the evaluation of a solution for that VRP variant. The cost of every neighbor solution is computed efficiently in the sense that only those fragments of the new solution that are different from the initial solution are recomputed. The proposal allows a time saving in the implementation of the solution of the problem due to the fact that the user does not have to write any code to evaluate the neighbor solutions in an efficient way.

Índice general

Capítulo 1

Introducción

Bajo el nombre Problema de Enrutamiento de Vehículos (VRP) se engloba una familia de problemas que tienen en común el objetivo de satisfacer las demandas de clientes geográficamente dispersos al mismo tiempo que se optimiza el uso de algún recurso (tiempo, combustible, etc). Estos problemas se presentan con frecuencia en la práctica, principalmente en el área de la logística.

Estos son problemas NP-Duros [2], por lo que los métodos exactos para solucionarlos son muy costosos y solo permiten resolver instancias pequeñas. Para solucionar los VRP que se presentan en la práctica se utilizan métodos aproximados, como heurísticas o metaheurísticas. De estos, entre los que han dado mejores resultados están los métodos de búsqueda local [4].

Para solucionar un VRP mediante un algoritmo de búsqueda local se parte de una solución inicial S_0 a la que se le hacen ciertas transformaciones, como cambiar clientes de rutas o cambiar vehículos, y a partir de estos cambios se obtiene un conjunto de nuevas soluciones que se denomina *vecindad* de S_0. De este conjunto se selecciona una solución S_1 y a partir de esta última se repite el proceso hasta que se cumpla algún criterio de parada.

Al proceso que se sigue para seleccionar S_1 se le llama *exploración de la vecindad*. El criterio para escoger la solución S_1 entre las restantes nuevas soluciones depende, en la mayoría de los casos, del valor que toma la función objetivo en cada una de las nuevas soluciones, por lo que parte del proceso de exploración consiste en evaluar la función objetivo en algunas o en todas las soluciones de la vecindad.

La forma eficiente de calcular el costo de estas soluciones vecinas es

reevaluar solamente los fragmentos que sean distintos de la original, ya que las transformaciones que se le realizan a esta para obtener las vecinas no cambian mucho la estructura de las mismas.

El objetivo de este trabajo es diseñar e implementar una estrategia que permita evaluar eficientemente cualquier solución de cualquier vecindad de un VRP dado, sin que el usuario tenga que programar el código para esta evaluación. Solamente es necesario escribir el código para la evaluación de una solución. Que el proceso sea eficiente significa que solamente se reevalúen las regiones de las nuevas soluciones que cambien con respecto a la solución que les dio origen.

Con un método de evaluación automática el usuario no está obligado a enfrentar el problema de elaborar un algoritmo que evalúe las soluciones vecinas de manera eficiente, el cual puede resultar complejo. Por ejemplo, en un Problema de Enrutamiento de Vehículos con Ventanas de Tiempo (TWVRP), donde cada cliente solamente puede ser atendido en un intervalo de tiempo determinado, cuando se extrae un cliente, disminuye el tiempo en que el vehículo llega a los clientes restantes en la ruta. Es por eso que se debe calcular cuál es el momento en el que se llega a cada cliente en la nueva solución, y hacer este cálculo de manera eficiente requiere un algoritmo que toma tiempo diseñar e implementar. Asimismo, este algoritmo no se puede aplicar para cualquier otra variante de VRP, para la cual sería necesario elaborar un nuevo algoritmo.

Con la propuesta del presente trabajo el usuario solo tendría que escribir el código para evaluar una solución. A partir de este código y de una solución inicial S_0 se construye un *grafo de evaluación*, que representa la evaluación de S_0 en la función objetivo.

A partir del *grafo de evaluación* será posible evaluar cualquier solución vecina de manera eficiente y automática. Para evaluar un vecino solo hay que modificar un fragmento del grafo. Además, esas modificaciones se pueden hacer de manera automática a partir de las transformaciones que se le aplican a la solución inicial para obtener una solución vecina.

Utilizando la técnica propuesta e implementada en este trabajo es posible evaluar soluciones vecinas de una dada en el Problema de Enrutamiento de Vehículos con Restricción de Capacidad (CVRP), en el Problema de Enrutamiento de Vehículos con Múltiples Depósitos (MDVRP) y en el Problema de Enrutamiento de Vehículos con Flota Heterogénea e Infinita (HFVRP). En estos casos se emplearon las operaciones *extraer cliente de una ruta*, *insertar cliente en una ruta*, *cambiar depósito* y *cambiar vehículo* según tuviera sentido en el problema. Además, al ser implementadas estas variantes, se puede resolver cualquier combinación de ellas, por ejemplo

un Problema de Enrutamiento de Vehículos con Múltiples Depósitos y Flota Heterogénea e Infinita.

El método que se expone en el presente trabajo es extensible debido a que se puede aplicar a la solución de otras variantes si se agregan los componentes necesarios. Por ejemplo, para implementar un Problema de Enrutamiento de Vehículos con Entregas y Recogidas (PDVRP), es necesario agregar un nuevo tipo de vehículo que realiza recogidas, al contrario del otro tipo de vehículo que realiza entregas. Además se debe implementar una nueva operación que se lleva a cabo en la evaluación de una solución, la *recogida de la demanda del cliente*. Con estos nuevos elementos y lo que se tiene implementado se pudiera aplicar el método propuesto en el presente trabajo a la variante de Problema de Enrutamiento de Vehículos con Entregas y Recogidas.

No obstante, existen variantes para las que todavía no es posible aplicar esta propuesta. Estas variantes son aquellas que incluyen penalizaciones durante el recorrido de la ruta. Por ejemplo, en el Problema de Enrutamiento de Vehículos con Entregas y Recogidas Simultáneas cada cliente es visitado una sola vez, pero es necesario entregar y también recoger cierta cantidad de algún producto a cada cliente. Por tanto, el vehículo que atiende cada ruta debe tener capacidad suficiente para satisfacer las demandas de cada cliente. Al contrario del Problema de Enrutamiento de Vehículos con Restricción de Capacidad, cuando hay entregas y recogidas simultáneas puede que la cantidad total de producto que se deba entregar sea menor que la capacidad del vehículo pero que durante el recorrido de la ruta se incumpla la restricción de capacidad debido a la cantidad de producto que se recoge a los clientes. Las soluciones en las que esto sucede no son factibles y es usual que se les aplique una penalización. Es por eso que al evaluar una solución de esta variante se verifica que la carga del vehículo no exceda su capacidad luego de atender a cada cliente, en caso de que exista un exceso de carga entonces se aplica una penalización. A las variantes de VRP como esta, en las que se aplican penalizaciones durante el recorrido de la ruta no es posible aplicarles aún la propuesta del presente trabajo.

Este documento está estructurado de la siguiente forma. En el siguiente capítulo se presentan los preliminares. En el capítulo 3 se presenta la idea de grafo de evaluación y el proceso para construirlo. En el capítulo 4 se presenta una definición formal de la estructura *grafo de evaluación*. En el capítulo número 5 se presentan los detalles de la implementación de la propuesta del presente trabajo. En el capítulo 6 se mencionan las variantes de VRP que se han implementado con el uso de la *evaluación automática* que se presenta en este trabajo, mientras que en el capítulo 6.2 se presentan las

conclusiones y las recomendaciones.

En el siguiente capítulo se presenta el Problema de Enrutamiento de Vehículos, los algoritmos de Búsqueda Local, los criterios de vecindad y el proceso de exploración de una vecindad.

Capítulo 2

Preliminares

En en este capítulo se presentan los elementos básicos del Problema de Enrutamiento de Vehículos, los algoritmos de búsqueda local, los criterios de vecindad y la exploración de vecindades.

Al abordar el tema de la exploración de vecindades como parte de los algoritmos de búsqueda local que se utilizan para resolver el Problema de Enrutamiento de Vehículos se expondrá una deficiencia que se se presenta en este proceso. Esta consiste en la necesidad de elaborar un algoritmo para evaluar las soluciones vecinas sin realizar operaciones innecesarias. Como se expondrá, implementar este algoritmo puede demorar el proceso de implementación de la solución al problema. Esta deficiencia se resuelve con la evaluación automática que se propone en el presente trabajo

2.1. Problema de Enrutamiento de Vehículos

El primer artículo sobre Problemas de Enrutamiento de Vehículos (VRP) data de 1959 [1], cuando Dantzig y Ramser abordan un problema concerniente a la distribución de gasolina. En este artículo los autores presentan la formulación matemática del VRP y una aproximación algorítmica para resolverlo.

Los VRP constituyen un grupo de problemas que comparten el objetivo de satisfacer las demandas de clientes dispersos geográficamente, para lo cual se emplea una flota de vehículos y además se quiere optimizar el uso de algún recurso como el tiempo o el combustible. En la literatura se pueden encontrar más de 30 variantes [3]. A continuación se mencionan algunas de ellas.

El Problema de Enrutamiento de Vehículos con Restricción de Capaci-

dad (CVRP) incluye vehículos que tienen una capacidad de carga limitada y clientes con una determinada demanda. El objetivo es satisfacer a cada cliente de manera óptima y sin que la demanda total de cada ruta exceda la capacidad del vehículo que recorre dicha ruta. Si los vehículos que intervienen en el problema poseen capacidades o características distintas, entonces se dice que el problema tiene flota heterogénea.

En un Problema de Enrutamiento de Vehículos con Flota Heterogénea (HFVRP) se tienen vehículos que se diferencian en al menos una característica que puede ser la capacidad de carga, el consumo de combustible, o cualquier otra relevante para el problema. Existen a su vez dos variantes de HFVRP, con flota finita o infinita. En un HFVRP con flota finita existe una cantidad limitada de vehículos de cada tipo mientras que en los casos de HFVRP con flota infinita no existe esta restricción, o sea, que de cada tipo de vehículo se pueden usar tantos como se desee.

En el Problema de Enrutamiento de Vehículos con Entregas y Recogidas (PDVRP) los clientes tienen dos tipos de demanda, una que refleja cuánto se les debe entregar y otra que refleja cuánto se les debe recoger. Por tanto, en este problema los vehículos no solo descargan productos sino que también los recogen. Al igual que en CVRP, en este caso, no se permite que el vehículo que recorre cierta ruta cargue más de lo que permite su capacidad. Como diferencias con respecto al CVRP se tiene que en este problema los clientes pueden ser visitados más de una vez, además se tienen dos tipos de vehículos, los que recogen y los que entregan.

Otra variante del VRP es el Problema de Enrutamiento de Vehículos con Ventanas de Tiempo (TWVRP). En este caso, cada cliente solo puede ser atendido en un determinado intervalo de tiempo. El objetivo es satisfacer a los clientes de manera óptima sin que ningún vehículo llegue a un cliente fuera del intervalo de tiempo para atenderlo. En este problema puede existir un tiempo de servicio para antender a cada cliente.

Para cada variante de VRP existen soluciones que no son factibles, o sea, que no cumplen con las restricciones del problema. Por ejemplo, una solución de un CVRP donde la demanda total de los clientes de una ruta sobrepase la capacidad del vehículo no es factible. Es usual incluir un término en la función objetivo para que las soluciones no factibles resulten más costosas que las factibles. A este término se le denomina penalización. Una forma de penalizar una solución no factible de un CVRP cuando la demanda total de una ruta excede la capacidad del vehículo, es sumar el exceso de carga multiplicado por un determinado factor al resultado de evaluar dicha solución, de forma tal que se asegure que sea más costosa que cualquier solución factible.

Todos los Problemas de Enrutamiento de Vehículos son NP-duros [2], por lo que los métodos exactos resultan muy costosos. En la práctica se emplean métodos aproximados para resolverlos. Los algoritmos de Búsqueda Local son unos de los métodos aproximados que mejores resultados han arrojado en la resolución de estos problemas [6, 7]. En la siguiente sección se presentan estos algoritmos.

2.2. Algoritmos de Búsqueda Local

En los algoritmos de búsqueda local se parte de una solución inicial S_0, que se modifica de varias maneras para obtener un conjunto N_{S_0} llamado conjunto de soluciones *vecinas* de S_0, o vecindad de S_0. A las reglas que se aplican para modificar una solución y obtener las soluciones vecinas se les conoce con el nombre de *estructura de entorno* o *criterio de vecindad*.

Del conjunto de soluciones N_{S_0} se escoge una solución S_1 bajo algún criterio que en la mayoría de los casos depende de la evaluación de la función objetivo en todas las soluciones vecinas. El proceso que se realiza para seleccionar la solución S_1 entre las demás que componen la vecindad N_{S_0} se conoce como *exploración de la vecindad*. Tomando S_1 como nueva solución inicial se repite el proceso hasta que se cumpla algún criterio de parada.

A modo de ejemplo, sea $S_0 = [[1,2],[3]]$ la solución inicial que está formada por dos rutas. En la primera ruta se parte del depósito, se visita al cliente 1, luego al 2 y, finalmente se retorna al depósito central. En la segunda ruta se parte del depósito, se visita al cliente 3 y se retorna. Si se considera el criterio de vecindad *cambiar un cliente de posición* y se le aplica a S_0 se puede obtener la vecindad N_{S_0} que se muestra en la figura 2.1, donde todas las soluciones vecinas son el resultado de cambiar un cliente de posición en la solución S_0. En un algoritmo de búsqueda local, de esta vecindad N_{S_0}, se seleccionaría una solución S_1 y se repitiría el proceso con S_1.

Existen variantes de los métodos de Búsqueda Local que se diferencian en la forma de explorar las vecindades, en la forma de seleccionar el mejor vecino, y en los criterios de parada del algoritmo. Por ejemplo, en *Recocido Simulado* se selecciona una solución al azar de toda la vecindad y en Búsqueda de Vecindad Variable (*VNS*) se exploran todas las soluciones vecinas o una parte de ellas. En VNS siempre se aceptan soluciones mejores que la actual, y en Recocido Simulado se pueden aceptar soluciones peores para escapar de los mínimos locales. Estas soluciones con peor evaluación de la función objetivo se aceptan con cierta probabilidad que depende de un parámetro del algoritmo [4].

- [[2,1],[3]]
- [[2],[1,3]]
- [[2],[3,1]]
- [[1],[2,3]]
- [[1],[3,2]]
- [[3,1,2]]
- [[1,3,2]]
- [[1,2,3]]

Figura 2.1: Vecindad de la solución S_0 al aplicar el criterio *cambiar un cliente de posición.*

En la siguiente sección se presentan los criterios de vecindad que son los cambios que se le aplican a las soluciones para obtener las vecindades. Este proceso de obtención de soluciones vecinas forma parte de la exploración de una vecindad, y en este trabajo se propone automatizar la creación del código para realizar esta exploración.

2.3. Criterios de Vecindad

En esta sección se presentan las estructuras de entorno o criterios de vecindad. Estos criterios se emplean en los algoritmos de búsqueda local para transformar las soluciones y obtener otras a las que se les denomina vecinas. Algunos de estos criterios de vecindad son *"cambiar un cliente de posición"*, *"intercambiar dos clientes"*, o *"cambiar el vehículo de una ruta"*.

Los criterios de vecindad que modifican las soluciones de un VRP están compuestos por operaciones elementales como *extraer un cliente de una ruta, insertar un cliente en una ruta, intercambiar los vehículos de dos rutas, cambiar el depósito de una ruta*, etc.

Por ejemplo, el criterio *cambiar un cliente de posición* está compuesto por las operaciones *extraer cliente de una posición dada* e *insertar al cliente en otra posición*, mientras que el criterio *cambiar el vehículo de una ruta* consta solamente de la operación *asignar a una ruta dada un determinado vehículo disponible*.

Los criterios de vecindad que se usan en un método de búsqueda local dependen de la variante de VRP que se esté resolviendo [5] porque, por ejemplo, en un problema donde todos los vehículos son del mismo tipo pero existen varios depósitos, la operación *cambiar vehículo* no tiene sentido pero *cambiar depósito* sí puede aplicarse.

A partir de una estructura de entorno y una solución inicial pueden obtenerse nuevas soluciones. El proceso de obtener estas nuevas soluciones y de seleccionar una de ellas bajo algún criterio se denomina exploración de la vecindad. En la siguiente sección se presenta este proceso.

2.4. Exploración de una vecindad

En la presente sección se presenta el proceso de exploración de vecindades que se realiza en los algoritmos de búsqueda local, y se plantea una deficiencia, que consiste en lo complejo que resulta diseñar un algoritmo eficiente para evaluar las soluciones vecinas y el tiempo que puede tomar implementar dicho algoritmo.

En la mayoría de los métodos de búsqueda local la selección de la nueva solución depende del valor que toma la función objetivo en cada una de las soluciones vecinas. Por ejemplo, se puede seleccionar la solución vecina que minimice la función objetivo entre todas las soluciones que componen la vecindad, o se puede seleccionar cualquier solución mejor que la actual.

Para realizar la exploración de una vecindad resulta ineficiente reevaluar completamente cada una de las soluciones vecinas debido a que el tamaño de las vecindades puede ser exponencial con respecto a la cantidad de clientes. Por ejemplo, para un problema con 65 clientes, el tamaño de una vecindad puede ser del orden de 10^7. Por otra parte, las estructuras de entorno usualmente están conformadas por pocas operaciones elementales (en la mayoría de los casos menos de 4 o 5), lo que implica que en las evaluaciones de las soluciones vecinas se repitan muchas de las operaciones que se realizaron para evaluar la solución inicial. Por tanto, resulta mucho más eficiente evaluar las soluciones vecinas evitando la repetición de operaciones ya realizadas en la evaluación de la solución inicial.

No obstante, realizar esta evaluación de manera eficiente resulta complejo debido a que las operaciones que conforman los criterios de vecindad cambian de una variante de VRP a otra y, por tanto, el algoritmo para ejecutar la evaluación también cambia.

Otro inconveniente es la cantidad de datos que se deben almacenar. Por ejemplo en un VRP con restricciones de capacidad y flota heterogénea, don-

de se pueden cambiar los vehículos, y estos vehículos tienen capacidades distintas, sería necesario guardar la demanda total de cada ruta por separado para que, al cambiar el vehículo se pueda determinar de manera eficiente si se inclumple la restricción de capacidad o no. La determinación de qué datos se tienen que almacenar para ser usados en la exploración, cómo obtenerlos y de qué manera almacenarlos también depende del tipo de VRP que se quiera resolver. Este es un proceso que consume tiempo, lo que hace que demore más la implementación del algoritmo para solucionar el problema.

En el presente trabajo se propone un método para que la evaluación de las soluciones vecinas se pueda realizar sin que sea necesario escribir código para este fin, y que, además, la misma se lleve a cabo sin reevaluar completamente las nuevas soluciones.

Este objetivo se logra a partir de la creación de una estructura que se definirá como *grafo de evaluación*, que es una representación en forma de grafo de la evaluación de la función objetivo en una determinada solución. En el siguiente capítulo se introduce el concepto de grafo de evaluación para, posteriormente, mostrar cómo a partir de él es posible evaluar soluciones vecinas sin que el usuario tenga que escribir código para ello.

Capítulo 3

Construcción de un grafo de evaluación

En este capítulo se muestra, de manera informal, cómo se puede construir un grafo de evaluación para una instancia de Problema de Enrutamiento de Vehículos con Restricción de Capacidad (CVRP). En capítulos posteriores se realiza una definición formal de *grafo de evaluación* y se demuestra que el proceso expuesto en el presente capítulo, en efecto, construye dicho grafo.

A modo de ejemplo, se desea resolver un CVRP que consta de 3 clientes, utilizando un algoritmo de búsqueda local, partiendo de la solución inicial $S_0 = [[1,2],[3]]$.

S_0 es una solución con dos rutas. En la primera se sale del depósito, se visitan a los clientes 1 y 2, y se regresa al depósito; mientras que en la segunda se sale del depósito, se visita al cliente 3 y se regresa al depósito. En la figura 3.1 de la página 12 se muestra un posible código de evaluación para una solución de un CVRP.

Para evaluar una solución de un CVRP se itera por cada cliente de cada ruta. Se aumenta la distancia total recorrida entre el *cliente previo* de la ruta y el cliente actual. También se aumenta la demanda total de la ruta en la demanda del cliente actual. Al finalizar el recorrido por los clientes de cada ruta se penaliza el exceso de demanda en la ruta, y dicha penalización se le adiciona a la distancia total. El *cliente previo* de cada ruta inicialmente es el depósito y se actualiza cada vez que se visita un cliente.

A partir de la solución S_0 y del código propuesto se puede construir un *grafo de evaluación* que será una representación en forma de grafo de la evaluación de S_0. Este grafo se usará para evaluar las soluciones vecinas

```
1  evalúa(S):
2    new_var distancia_total = 0
3    para toda ruta r de S:
4        new_var demanda_ruta = 0
5        new_var cliente_previo = deposito(r)
6        para todo cliente c en r:
7            incrementa distancia_total con...
                        d(cliente_previo, c)
8            incrementa demanda_ruta con demanda del cliente c
9            cliente_previo = c
10       incrementa distancia_total con...
                        d(c, depósito)
11       incrementa distancia_total con...
                    penalizar(demanda_ruta, capacidad_vehículo)
12    devuelve distancia_total
```

Figura 3.1: Código de evaluación de una solución. La operación `d(x,y)` representa la distancia entre `x` e `y`.

automática y eficientemente. Cuando se dice que la evaluación será automática se refiere a que no es necesario escribir código para realizarla, y eficiente se refiere a que no se evalúa totalmente la nueva solución sino que se evita repetir las operaciones realizadas para evaluar la solución inicial. Por ejemplo, como en la evaluación de S_0 se incrementó la distancia total en la distancia entre el depósito y el cliente 3, en una solución vecina donde el cliente 3 se encuentre a continuación del depósito no será necesario repetir el incremento de distancia para evaluar esta solución.

El proceso de construcción de un grafo de evaluación se divide en dos etapas: *inicialización* y *construcción*. En la siguiente sección se presenta la etapa de inicialización y en la sección 3.2 se presenta la etapa de construcción.

3.1. Inicialización del grafo de evaluación

En esta sección se aborda la etapa de inicialización, que es el primer paso para construir un grafo de evaluación.

En este paso se utiliza solamente la solución S_0, y por cada cliente y depósito se crea un nuevo nodo en el grafo. En la figura 3.2 se muestra el grafo

inicializado a partir de la solución S_0. Se incluye un nodo que representa al depósito por cada vez que se sale o se entra al mismo en la solución inicial. Un grafo inicializado no contiene ninguna arista debido a que en un grafo de evaluación las aristas indican las entradas y las salidas de las operaciones que se realizan en la evaluación de la solución. Estas operaciones se insertarán en el grafo en el próximo paso el cual se presenta en la siguiente sección.

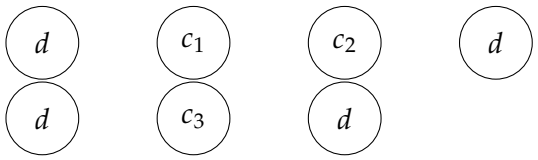

Figura 3.2: Grafo de evaluación inicializado.

3.2. Construcción del grafo de evaluación

A partir del grafo inicializado se puede construir el grafo de evaluación. Esta construcción se lleva a cabo a partir de la evaluación de la solución. En la figura 3.1 se presenta el código que evalúa una solución en la función objetivo.

En la figura 3.3 se muestran las operaciones que intervienen en el cálculo del costo de la solución. En la tabla no se muestran las operaciones como los ciclos y la actualización del cliente previo porque estas operaciones no tributan directamente al valor numérico del costo de la solución. Cada vez que una de estas operaciones se ejecuta al evaluar la solución inicial, se crean nuevos nodos y aristas del grafo como se describirá a continuación. Los diferentes nodos pueden representar clientes, depósitos, vehículos, variables utilizadas en la evaluación como *distancia total* o *demanda de la ruta* y operaciones como *incrementar distancia*, *incrementar demanda* y *penalizar*.

Por ejemplo, las operaciones 1 y 2 de la figura 3.3 crean nuevos nodos en el grafo que representan variables con el valor inicial 0 (figura 3.5).

La operación 3 (incrementar *distancia-total* en $d(\text{depósito}, c_1)$) crea un nodo *incrementa distancia* que está conectado a los nodos del grafo que corresponden al depósito y al cliente 1, e incrementa el valor almacenado en el nodo que representa a la variable *distancia-total* en la distancia entre el cliente y el depósito (figura 3.6).

La operación 4 (incrementar *demanda-ruta* en la demanda de c_1) es muy parecida a la 3, solo que esta vez se crea un nodo *incrementa-demanda* que estará conectado al vértice c_1 y que incrementa el valor almacenado en el nodo *demanda-ruta* en la demanda de ese cliente (figura 3.7).

En la operación 8 (incrementar *distancia-total* con penalización), se crea un nuevo nodo *penaliza* que está conectado al vértice *demanda-ruta*, en este nodo *penaliza* se calcula la penalización de la ruta y se le incrementa al valor contenido en el nodo *distancia-total* (figura 3.8).

Desde la operación 9 a la 13 se repiten las mismas operaciones pero en ese caso se aplican a la segunda ruta de la solución. La operación 14 y última marca el nodo *distancia-total* para representar que es en ese nodo donde se almacena el resultado final.

En las figuras desde la 3.5 a la 3.9 se ilustra el proceso de la construcción para el ejemplo que se ha presentado. En la figura 3.4 se presenta la distancia entre cada par de clientes en el problema, así como las demandas de cada cliente. En este caso, la capacidad del vehículo es de 5 unidades y se penaliza multiplicando por 100 el exceso de demanda de la ruta, donde el exceso de demanda se define como la suma de la demanda de todos los clientes en la ruta menos la capacidad del vehículo.

	Operación
1	crear variable *distancia-total* con valor 0
2	crear variable *demanda-ruta* con valor 0
3	incrementar *distancia-total* en $d(\text{depósito}, c_1)$
4	incrementar *demanda-ruta* en la demanda de c_1
5	incrementar *distancia-total* en $d(c_1, c_2)$
6	incrementar *demanda-ruta* en la demanda de c_2
7	incrementar *distancia-total* en $d(c_2, \text{depósito})$
8	incrementar *distancia-total* con penalización
9	crear variable *demanda-ruta* con valor 0
10	incrementar *distancia-total* en $d(\text{depósito}, c_3)$
11	incrementar *demanda-ruta* en la demanda de c_3
12	incrementar *distancia-total* en $d(c_3, \text{depósito})$
13	incrementar *distancia-total* con penalización
14	retornar *distancia-total*

Figura 3.3: Lista ordenada de las operaciones que se realizan para evaluar la solución S_0.

	d	c_1	c_2	c_3
d	0	1	2	3
c_1	1	0	2	1
c_2	2	2	0	7
c_3	3	1	7	0

d	c_1	c_2	c_3
0	4	3	1

Figura 3.4: Matriz de distancia y demanda de cada cliente del ejemplo.

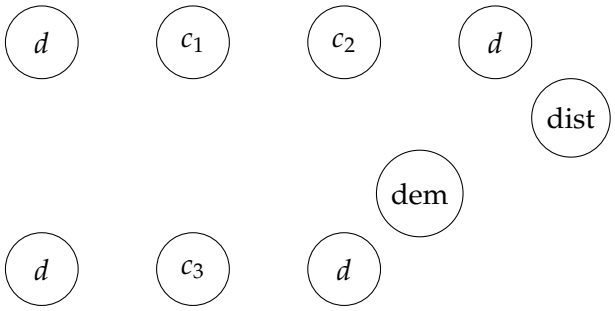

Figura 3.5: Después de ejecutarse las instrucciones *new_var distancia_total = 0* y *new_var demanda_ruta = 0*, esta última por primera vez, se crean los nodos que representan la distancia total y la demanda de la primera ruta en el grafo respectivamente.

De esta manera queda construido un grafo de evaluación para este CVRP con la solución inicial S_0. En la siguiente sección se muestra cómo puede ser utilizado el grafo para obtener automáticamente la evaluación de una solución vecina, donde dicha solución se describe en términos de las operaciones que hay que aplicarle a la solución actual para obtenerla.

3.3. Evaluación automática de soluciones vecinas a partir del grafo

Con el grafo de evaluación se puede realizar la evaluación de las soluciones vecinas sin que haya que escribir código para tal fin, que es el objetivo del presente trabajo.

A continuación se ilustra cómo obtener, a partir de la solución S_0 del

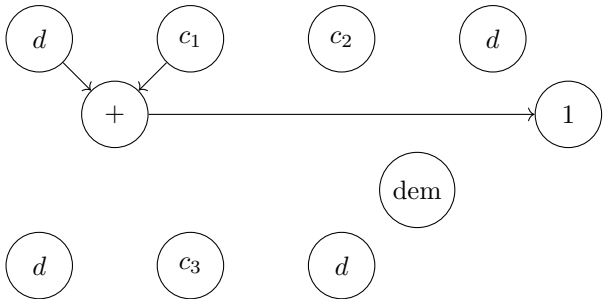

Figura 3.6: Después de ejecutarse la instrucción *incrementa distancia_total con d(cliente_previo, c)* por primera vez. Se crea un nodo *incrementa distancia* que aumenta el valor de *distancia_total* en la distancia entre el depósito y el cliente 1.

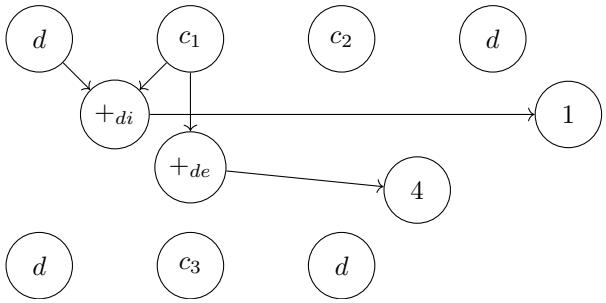

Figura 3.7: Después de ejecutarse la instrucción *incrementa demanda_ruta con demanda del cliente c* por primera vez. Se crea un nodo *incrementa demanda* que aumenta el valor de *demanda_ruta* en la demanda del cliente 1.

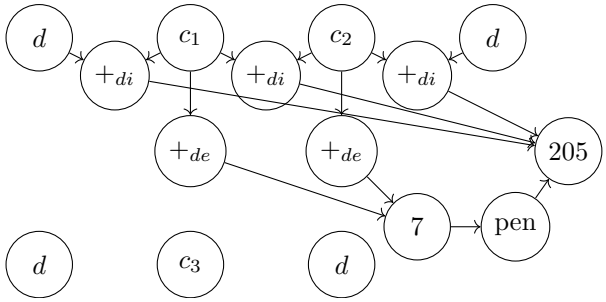

Figura 3.8: Al finalizar el ciclo que recorre toda la ruta se ejecuta *incrementa distancia_total con penalizar(demanda_ruta, capacidad_vehículo)* que construye el nodo penalizador correspondiente. La siguiente ruta se construye de manera análoga

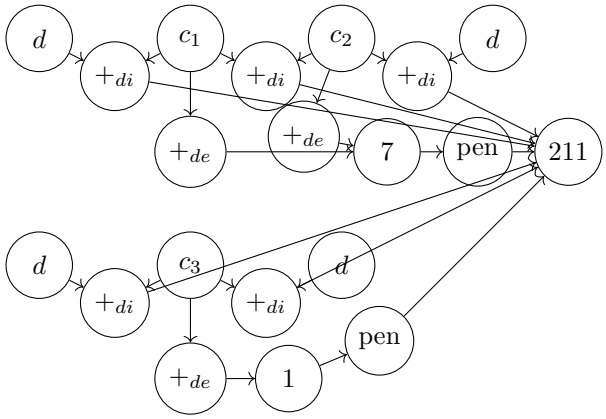

Figura 3.9: Grafo de evaluación completamente construido.

ejemplo analizado en la sección anterior que ya fue evaluada, la evaluación de una solución vecina definida mediante las operaciones *cambiar al cliente de la primera posición de la primera ruta a la segunda posición de la segunda ruta*.

Debido a la forma en la que se construye el grafo de evaluación, en especial a la fase de inicialización, existe una correspondencia entre los clientes y depósitos del problema y los nodos del grafo, por lo que se puede saber cuál es el nodo del grafo que le corresponde al cliente de la primera posición de la primera ruta.

Al identificar el nodo que se quiere extraer, se determinan las operaciones en las que este nodo participa, que son los nodos a los cuales está conectado. En este caso serían dos operaciones *incrementa distancia*, y una operación *incrementa demanda*, exactamente las operaciones 3, 4 y 5 de la tabla de la figura 3.3.

Cuando se determinan los nodos que representan a las operaciones en las que interviene el cliente que se desea extraer, se procede a realizar lo que se define como *desevaluación* de estas operaciones. *Desevaluar* una operación *incrementa distancia* consiste en decrementar la distancia previamente aumentada. Análogamente, *desevaluar* una operación *incrementa demanda* consiste en decrementar la demanda aumentada. Esta *desevaluación* se realiza con cada una de las operaciones en las que interviene el nodo que se quiere extraer.

Una vez que las operaciones han sido *desevaluadas*, se extrae el nodo que corresponde al cliente de la primera posición de la primera ruta. Para que el grafo de evaluación quede como si nunca se hubiese atendido al cliente 1, es necesario agregar, además, un nodo *incrementa distancia* entre el déposito y el cliente 2. En la figura 3.10 se ilustra el grafo de evaluación después de terminada la primera de las operaciones para obtener la solución vecina.

Para completar las operaciones que definen al vecino es necesario insertar en la segunda posición de la segunda ruta al cliente que se extrajo. Para ello se identifican los nodos correspondientes al cliente que ocupa esa posición (cliente 3) y al que está a continuación de este en la solución, que en este caso, es el nodo que representa al depósito. Entre estos dos nodos se debe colocar al vértice que representa al cliente 1.

Para lograrlo, se *desevalúan* las operaciones en las que intervienen, al mismo tiempo, el cliente 3 y el depósito. En este caso se trata de una operación *incrementa distancia*, específicamente la número 12 en la tabla de la figura 3.3. Después de *desevaluar* esta operación, se inserta al nodo que representa al cliente 1 en la posición deseada. Por último se agregan las operaciones necesarias para calcular el valor de la función objetivo en la nueva solución, en este caso serían *incrementa distancia entre cliente 3 y cliente 1*,

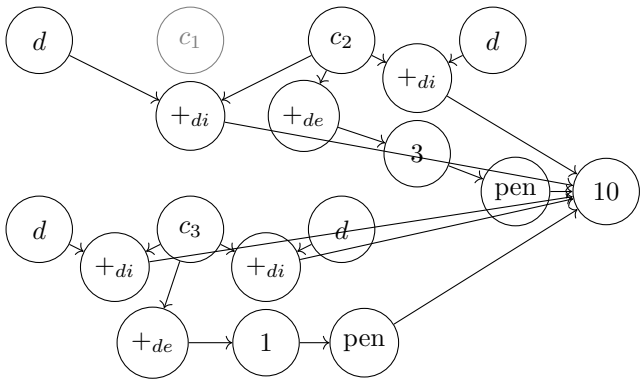

Figura 3.10: Grafo de evaluación luego de extraer al cliente 1

incrementa distancia ente cliente 1 y depósito e *incrementa demanda del cliente 1*. En la figura 3.11 se ilustra el grafo de evaluación luego de insertar al nodo correspondiente al cliente 1 en la posición descrita.

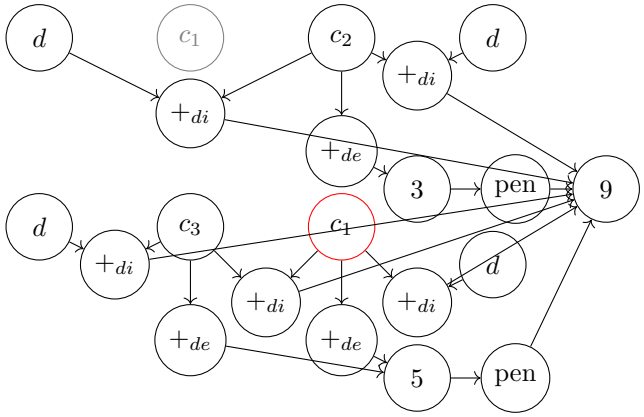

Figura 3.11: Nuevo grafo de evaluación cuando se ha aplicado el criterio de vecindad completamente.

El grafo representa ahora la evaluación de la nueva solución vecina y solamente se ha modificado el subgrafo que representa a las partes de la solución original que cambiaron por el criterio de vecindad. Además, el

usuario no necesitó escribir código para la evaluación de la nueva solución porque las operaciones que conforman los criterios de vecindad son métodos que realizan todas estas transformaciones del grafo. El usuario solamente necesita hacer llamados a esos métodos según el criterio de vecindad que esté aplicando.

En la siguiente sección se muestra cómo es posible lograr estas transformaciones en el grafo de forma tal que se obtenga el resultado de evaluar la función objetivo en la nueva solución sin necesidad de reevaluarla totalmente.

3.4. Propiedades de los grafos de evaluación que permiten la evaluación automática de soluciones vecinas

En esta sección se presentan las propiedades de los nodos de un grafo de evaluación a través de las cuales se logra explorar las vecindades de un VRP de manera eficiente y automática.

La correspondencia entre un elemento de la solución (cliente, depósito, vehículo, etc) y un nodo del grafo, que se logra en la fase de inicialización, permite expresar las transformaciones de la solución en términos de transformaciones del grafo. Por ejemplo, para extraer al cliente de la primera posición de la primera ruta e insertarlo en la segunda posición de la segunda ruta, existe una transformación en el grafo de evaluación que es equivalente y que se mostró en la sección anterior

Los nodos que representan operaciones (*incrementar distancia*, *incrementar demanda* y *penalizar*) pueden ser *desevaluados*. Esto significa que un nodo que represente a una operación no solo realiza esta operación sino que también puede hacer la operación inversa. En el caso del nodo que representa a la operación *incrementar distancia*, esta operación inversa sería *decrementar distancia*. En general la operación inversa es aquella que revierte el cambio en la evaluación que provocó la operación original.

En el ejemplo que se ha estado desarrollando, el nodo *incrementar distancia* incrementa el valor contenido en el nodo *distancia-total* pero en caso que sea necesario se puede decrementar el valor previamente incrementado. Algo similar ocurre con los nodos *incrementa demanda*, que se puede incrementar y decrementar la demanda total de la ruta, y *penalizar*, con el que se puede penalizar un exceso de carga o cambiar esa penalización si la ruta varía su demanda.

Otra propiedad de los nodos que representan operaciones es la de *actualización en cadena*. Esta propiedad permite que cuando un nodo que representa una operación se evalúa o desevalúa, también se evalúen o desevalúen los demás nodos que representan operaciones y que dependen del primero.

Por ejemplo, cuando se extrae el nodo correspondiente al cliente 1, se desevalúa el nodo *incrementa demanda* al que el nodo c_1 está conectado, con lo que se decrementa el valor de la demanda de la ruta. Pero como el nodo *demanda total* de la ruta 1 está conectado a su vez al nodo penalizar, este último depende de manera indirecta del nodo *incrementa demanda*, por lo que también se desevalúa. Al insertar un cliente, se adiciona un nuevo nodo *incrementa demanda*, para aumentar la demanda del nuevo cliente que se inserta en la ruta. Esto provoca que cambie el valor de la demanda de la ruta y que también se deba desevaluar el nodo penalizar de dicha ruta.

Las propiedades descritas en este epígrafe posibilitan el funcionamiento de la evaluación automática de soluciones vecinas. En el siguiente capítulo se abordan formalmente las propiedades de los grafos de evaluación y se define esta estructura junto a los elementos que la componen.

Capítulo 4

Grafo de evaluación. Definición y propiedades

En este capítulo se definen formalmente los grafos de evaluación y sus propiedades. En el capítulo anterior se mencionaron los elementos que componen a un grafo de evaluación y se mostró un ejemplo de cómo construir esta estructura. En la siguiente sección se definen estos conceptos.

4.1. Definiciones

En esta sección se define formalmente la estructura grafo de evaluación. Para ello es necesario caracterizar varios elementos que formarán parte de la definición final.

Definición 1 *Entidad del problema: Cada uno de los elementos que componen la solución de una determinada variante de un VRP.*

Por ejemplo, en un problema con flota heterogénea las entidades del problema serían los clientes, los vehículos y el depósito central.

Definición 2 *Información del problema: Dato que forma parte de la entrada del problema.*

Un ejemplo de información en el CVRP es la demanda de cada cliente y también la distancia entre cada cliente. Para el HFVRP, la capacidad de cada vehículo es también una información del problema.

Definición 3 *Variable: Se refiere a toda variable que se use durante la evaluación de una solución y contenga un valor numérico que no sea información del problema ni entidad del problema.*

Ejemplos de variables serían *distancia-total*, *demanda-de-ruta*, etc.

Definición 4 *Operación: Función que se aplica durante la evaluación de una solución.*

Durante la evaluación de una determinada solución es común aplicar operaciones como *incrementar distancia total*, *aumentar o disminuir la carga de un vehículo*, *penalizar exceso de carga*, etc.

Definición 5 *Nodo operacional: Nodo que representa una operación.*

Un nodo operacional es la representación en el grafo de evaluación de una operación. Toda operación recibe una entrada y provoca que se transforme una determinada salida. Por ejemplo, cuando se aplica la operación *incrementa la distancia total en la distancia entre los clientes 1 y 2* se incrementa el valor contenido en la variable *distancia-total* (salida) en dependencia de la distancia entre los clientes 1 y 2 (entradas).

En la definición 5 se establece la representación de una operación, por tanto se hace necesario definir la representación de las entradas y las salidas de dichas operaciones.

Definición 6 *Nodo de entrada/salida: Nodo que representa la entrada o la salida de alguna operación.*

Por ejemplo, en la evaluación de la solución del ejemplo se *incrementa la distancia total en la distancia entre los clientes 1 y 2*, entonces existen nodos de *entrada/salida* que representan a los clientes 1 y 2 en el grafo y que constituyen la entrada de un nodo operacional que represente la operación *incrementar distancia*. La salida de dicha operación sería la variable *distancia-total*, por lo que también se crea un nodo de *entrada/salida* que representa a esta variable.

Los arcos dirigidos en un grafo de evaluación representan la relación *ser entrada de* o *ser salida de*. Todos los arcos en un grafo de evaluación unen a nodos de entrada/salida con nodos operacionales. En caso de que un arco apunte a un nodo operacional, esto implica que lo que está representado por el nodo de entrada/salida (entidad del problema o variable) es entrada de la operación representada por el nodo operacional. Si, por el contrario,

Figura 4.1: En este ejemplo los antecesores de $+$ (que representa la operación *incrementa distancia*) son c_1 y c_2. El sucesor de $+$ es el nodo *distancia-total*. Al mismo tiempo, los antecesores de $+$ son los nodos que representan las entradas de la operación *incrementa distancia* y el sucesor representa la salida de dicha operación.

el arco apunta a un nodo de entrada/salida, ese nodo al que apunta el arco representa la salida de la operación que está representada por el nodo del cual sale la arista. En la figura 4.1 se tiene un ejemplo de un nodo que representa la operación *incrementa distancia*. En este ejemplo las entradas están representadas por los nodos c_1 y c_2, y la salida es el nodo *distancia-total*

Se denota con P al conjunto de las entidades del problema y las variables. También se denota con Op al conjunto de todas las operaciones.

Definición 7 *Función de entrada y función de salida: Se define como función de entrada a* $I : Op \rightarrow 2^P$ *que es la función que dada una operación devuelve sus entradas. Análogamente se define la función de salida* $O : Op \rightarrow 2^P$ *para las salidas.*

Si p es la operación *incrementa la distancia total en la distancia entre los clientes 1 y 2*, entonces $I(p)$ devuelve los clientes 1 y 2 que constituyen la entrada de p. Asimismo $O(p)$ devuelve la variable *distancia-total*.

Se denota con V_{io} al conjunto de los nodos de entrada/salida y con V_o al conjunto de los nodos operacionales. En lo adelante se denota por A el conjunto de arcos.

A continuación se define la función antecesor y sucesor para todo nodo v de un grafo dirigido. La función antecesor devuelve todos los nodos de los cuales sale un arco que apunta a v. Análogamene, la función sucesor devuelve todos los nodos a los cuales apunta alguna arista que salga de v. Formalmente:

Definición 8 *Funciones de antecesor y sucesor en un grafo: Se define* $I_G : V_o \rightarrow$

$2^{V_{io}}$ *tal que* $I_G(v) = \{w :< w, v >\in A\}$*, como la función antecesor. Análogamente se define la función sucesor como* $O_G(v) = \{w :< v, w >\in A\}$.

Por ejemplo en el grafo de la figura 4.1, si se toma como v el nodo que representa la operación *incrementa distancia entre el cliente 1 y el 2*, entonces $I_G(v) = \{c_1, c_2\}$, y $O_G(v) = \{$ *distancia-total*$\}$.

Para que un grafo sea de evaluación debe existir una correspondencia entre cada operación que se realiza durante la evaluación de una solución y los nodos del grafo. Además las entradas y salidas de cada operación deben estar relacionadas con los nodos conectados al vértice que representa la operación. Esta noción de grafo de evaluación se formaliza con las siguientes dos definiciones.

Definición 9 *Función de correspondencia: La función* $\Phi : P \cup Op \rightarrow V_o \cup V_{io}$ *que hace corresponder cada entidad del problema, variable u operación con el nodo que la representa en el grafo se denomina función de correspondencia.*

En el ejemplo que se ha estado desarrollando, $\Phi($*cliente 1*$) = c_1$. O sea, al cliente 1 le corresponde el nodo del grafo denotado por c_1.

A partir de las definiciones anteriores se puede definir formalmente la estructura *grafo de evaluación*

Definición 10 *Grafo de evaluación: Un grafo* $G = \{V = V_o \cup V_{io}, A\}$ *dirigido y bipartito es un grafo de evaluación de una solución de una variante de VRP si al evaluar dicha solución en la función objetivo se cumple que* $\forall p \in Op$, $I(p) = \Phi^{-1}(I_G(\Phi(p)))$ *y* $O(p) = \Phi^{-1}(O_G(\Phi(p)))$.

Para que un grafo de evaluación quede definido es necesario que la función de correspondencia Φ sea inversible. Si se analiza el listado de la figura 3.3 y el grafo de la figura 3.9 se puede verificar que para cada operación que se realiza en el código de evaluación, cada variable y cada entidad del problema, existe un único nodo en el grafo que le corresponde, y viceversa. Además, para cada operación p, la relación *ser entrada de* (*salida de*) p es equivalente a la relación *tener una arista que apunte a* (*que salga de*) $\Phi(p)$.

La condición de bipartito viene dada por la propiedad de que todos los arcos se definen entre un nodo de *entrada/salida* y uno *operacional*.

En el capítulo anterior se ejemplificó el método para construir un grafo de evaluación basado en el código que evalúa una solución en la función objetivo. En la siguiente sección se demuestra que dicho método construye, en efecto, un grafo de evaluación tal y como fue definido en la definición 10.

4.2. Definición del proceso de construcción del grafo

Como se expuso con anterioridad, el proceso de construcción de un grafo de evaluación se realiza a partir de la evaluación de una solución en la función objetivo de un VRP dado. Cada instrucción es un método que crea los nodos y aristas correspondientes en el grafo. En la presente sección se demuestra que a partir del método descrito en el capítulo anterior se obtiene un grafo de evaluación como se caracteriza en la definición 10.

En lo adelante S_0 es una solución de una determinada instancia de VRP.

La primera fase en la construcción de un grafo de evaluación es la de *inicialización*, en la cual se crea un nodo del grafo por cada entidad del problema. Al finalizar esta fase la función de correspondencia Φ es biyectiva si se restringe su dominio al conjunto de las entidades del problema.

Cada vez que se declara una variable, se crea un nuevo nodo en el grafo que representa a esta variable. Se puede afirmar entonces que Φ es biyectiva si se toma como su dominio al conjunto P de entidades y variables del problema ($\Phi : P \rightarrow V_{io}$). Específicamente es la función que hace corresponder cada entidad o variable con el nodo que fue creado para representarla. Esta biyectividad de Φ con dominio en P se mantendrá invariante debido a que no se eliminarán ni se agregarán nodos que representen elementos de P en lo adelante, o sea, los conjuntos P y V_{io} se mantienen fijos.

Cada instrucción que denote una operación crea el nodo operacional correspondiente. Además, si se denota con $p \in Op$ a la operación que se realiza, se añaden los arcos $< u,v >$ y $< v,w >$ para todo $u \in \Phi(I(p))$ y $w \in \Phi(O(p))$ donde v es el nuevo nodo que representa a la operación p ($v = \Phi(p)$). Al igual que en el párrafo anterior, la función $\Phi : Op \rightarrow V_o$ es biyectiva.

Como $\Phi : P \rightarrow V_{io}$ y $\Phi : Op \rightarrow V_o$ son biyectivas, se tiene que $\Phi : P \cup Op \rightarrow V_{io} \cup V_o$ también es biyectiva. Además por la construcción que se ha hecho del grafo se cumplen las igualdades descritas en la definición de grafo de evaluación presentada en la sección anterior.

De esta forma queda demostrado que el método expuesto construye un grafo de evaluación a partir de la evaluación de una solución en un VRP dado.

La definición 10 establece qué es un grafo de evaluación debido a que define cuáles son sus nodos y aristas a partir de una solución inicial y un código de evaluación. El objetivo del presente trabajo es emplear esta estructura para evaluar de manera autómatica las soluciones vecinas de una solución dada. Para lograr tal objetivo se implementan métodos para modificar el grafo de evaluación a partir de un criterio de vecindad. En el si-

guiente capítulo se abordan los temas de implementación de la evaluación automática.

Capítulo 5

Implementación de la evaluación automática

En este capítulo se presentan los detalles de la implementación de los métodos que hacen posible la evaluación automática a partir de un grafo de evaluación. En la siguiente sección se presenta cómo se obtiene la evaluación de la solución inicial. En la sección 5.2 se introduce la evaluación eficiente de las soluciones vecinas a partir del grafo de evaluación.

5.1. Obtención del costo de la solución inicial

En el capítulo 3 se ejemplificó la construcción de un grafo para una instancia de CVRP. En este ejemplo el grafo quedaba construido y el nodo *distancia-total* contenía el valor de la función objetivo al ser evaluada en la solución inicial. En el capítulo anterior se demostró que este método construye un grafo de evaluación tal y como fue definido pero no se abordó el tema de la evaluación de la solución. En esta sección se describe cómo se logra la evaluación de la solución inicial.

Desde cualquier nodo de entrada/salida con el que se represente una entidad del problema se puede acceder a la información del problema que es relevante a esa entidad. Por ejemplo, si con el nodo se representa a un cliente, entonces desde el nodo se podrá acceder a la distancia entre ese cliente y cualquier otro cliente o cualquier depósito, también se podrá acceder a la demanda de ese cliente y, en general, a cualquier otra información que sea relevante para el cliente.

Todo nodo de entrada/salida que represente una variabe del problema contiene el valor de esa variable, de forma tal que al terminar la evalua-

ción el nodo contiene el valor final que toma la variable. Cuando esta se declara con un valor inicial no solo se crea el nodo de entrada/salida que la representa sino que además se le asocia a ese nodo dicho valor inicial. Por ejemplo cuado se declara la variable *demanda-ruta* con valor inicial 0, se crea el nodo *demanda-ruta* al que se le asocia el valor 0. Cuando una operación modifica el valor de la variable *demanda-ruta*, lo que realmente se modifica es el valor contenido en el nodo que representa a dicha variable. Esto se logra debido al proceso que se expone a continuación.

A cada nodo operacional se le asocian dos métodos: un método de *evaluación* que modifica a los nodos de salida a partir de los nodos de entrada tal y como se modificarían las variables de salida a partir de las variables de entrada en el código de evaluación, y un método de *desevaluación* que deshace la evaluación asociada al nodo operacional.

Por ejemplo, un nodo que representa una operación de *incrementa demanda de la ruta* tiene como entrada un nodo que representa a un cliente y su salida es un nodo que representa a la variable *demanda-ruta*. El método de *evaluación* asociado a este nodo incrementa el valor que se encuentra en el nodo *demanda-ruta* en la demanda del cliente de entrada. El método de *desevaluación* asociado al nodo operacional decrementa el valor del nodo *demanda-ruta* en la demanda del cliente de entrada, de esta forma se cancela la evaluación que se haya realizado del nodo operacional. En lo adelante la expresión "evaluar" un nodo operacional se refiere a ejecutar su método de evaluación asociado y, análogamente, la expresión "desevaluar" un nodo operacional se refiere a ejecutar su método de desevaluación asociado.

Siempre que se crea un nodo operacional, se le asocian los métodos de evaluación y desevaluación correspondientes y se evalúa. Por tanto, al concluir la construcción del grafo, el nodo que representa a la variable que contiene el valor de la función objetivo contiene dicho valor.

A partir del grafo de evaluación de la solución inicial, que contiene además los valores de cada variable, es posible evaluar automáticamente las soluciones vecinas que se puedan obtener a partir de la aplicación de un criterio de vecindad. En la siguiente sección se presenta cómo es posible obtener esta evaluación.

5.2. Evaluación de una solución vecina

Con el grafo de evaluación de la solución inicial se puede obtener el grafo de evaluación de cualquier solución vecina y el valor de esa solución al evaluarla en la función objetivo. En la presente sección se presenta el

procedimiento para lograr este objetivo.

Las soluciones vecinas se obtienen a partir de aplicarle a la solución inicial ciertas transformaciones. Por ejemplo, a partir de la solución $S_0 = [[1,2],[3]]$ se puede obtener la solución $S_1 = [[2],[1,3]]$ si se le aplican las operaciones *cambiar al cliente de la primera posición de la primera ruta a la primera posición de la segunda ruta*.

Cada operación que realiza una modificación a la solución actual, es un método que transforma el grafo de evaluación y evalúa y/o desevalúa los nodos operacionales correspondientes.

Por ejemplo, la transformación *extraer cliente* es un método que recibe las coordenadas del cliente, o sea, el índice de la ruta en la que se encuentra y su posición en dicha ruta. A partir de las coordenadas se determina el nodo del grafo que corresponde al cliente que se quiere extraer y se desevalúan todos los nodos operacionales de los cuales este nodo es entrada. Por último se extrae del grafo el nodo, se añaden los nuevos nodos operacionales que correspondan y se evalúan. El nodo extraído se almacena en una lista de *clientes extraídos*.

El método que extrae un cliente del grafo de evaluación es *polimórfico* con respecto al tipo de nodo que se quiere extraer del grafo, o sea que cambia en dependencia del tipo de nodo. Por ejemplo, en el caso de un nodo que representa a un cliente de un CVRP, este método desevalúa las operaciones *incrementa distancia* y la operación *incrementa demanda* en las que interviene el nodo y adiciona una operación *incrementa distancia* entre los nodos correspondientes. Si se tratara de un nodo que representa a un cliente de un Problema de Enrutamiento de Vehículos con Ventanas de Tiempo (TWVRP), también se desevaluarían las operaciones de incrementar el tiempo total, y se adicionarían, además, las operaciones referentes al tiempo necesarias para obtener el valor de la solución vecina. Esto es posible debido a que si se conoce que el nodo representa a un cliente de un CVRP, entonces se sabe además que este nodo tiene que estar conectado a dos operaciones que incrementan distancia y a una operación que incrementa demanda. Si se conocen las operaciones a las que debe estar conectado el nodo entonces se puede determinar qué operaciones hay que desevaluar.

Un ejemplo de extracción de un cliente se ilustra en la figura 3.10. Al extraer al nodo que representa al cliente 1 se desevalúan los nodos *incrementa distancia* de los que este nodo es entrada y también el correspondiente nodo *incrementa demanda*, se extrae al nodo que representa al cliente 1 y se añade y evalúa la operación *incrementa distancia* entre el depósito y el cliente 2. De esta forma se obtiene un nuevo grafo de evaluación que representa a la solución $S'_0 = [[2],[3]]$.

El método *insertar cliente* recibe tres argumentos k, i y j. Donde (i,j) son las coordenadas donde se desea insertar al cliente y k es el índice que tiene el cliente en la lista de clientes extraídos.

Con las coordenadas en las que se quiere insertar al cliente, pueden identificarse los nodos entre los cuales se insertará el nodo que representa al cliente previamente extraído. Como las rutas siempre comienzan y terminan en un depósito, el nodo se insertará siempre entre dos nodos v y w. Con v y w identificados, se procede a desevaluar todo nodo operacional que incluya como entrada a v y a w al mismo tiempo. A continuación se inserta al nodo entre v y w y se adicionan y evalúan los nodos operacionales correspondientes.

Al igual que en el caso de la extracción, el método de inserción también es *polimórfico* con respecto al nodo que se inserta. En este caso, al conocer el tipo de nodo que se inserta, se conocen qué operaciones se deben desevaluar y cuáles se deben adicionar al grafo.

La inserción de un cliente se ilustra en la figura 3.11. Para insertar al cliente 1 entre el cliente 3 y el último depósito de la segunda ruta se desevalúa el nodo *incrementa distancia* que tiene como entrada al cliente 3 y al depósito. Luego de añadir al cliente 1 se adicionan los nodos *incrementa distancia entre cliente* 3 *y cliente* 1, *incrementa distancia entre el cliente* 1 *y el depósito e incrementa demanda de la segunda ruta en la demanda del cliente* 1. Al evaluar los nodos adicionados el grafo de evaluación representa a la solución $S_1 = [[2],[3,1]]$ vecina de S_0.

Para obtener la evaluación de la nueva solución solamente fue necesario ejecutar los métodos que representan a cada una de las operaciones que intervienen en el criterio de vecindad. Además, solamente se reevaluaron los nodos del grafo que se modificaron por la aplicación de dichas operaciones. Por tanto, se ha logrado el objetivo de obtener automáticamente la evaluación de una solución vecina sin afectar la eficiencia del algoritmo.

No obstante, existe una fuente de ineficiencia oculta en este procedimiento. En la siguiente sección se presentan las *cadenas de evaluación* y las consecuencias que pueden acarrear. También se describe una solución a este inconveniente que se definirá como *evaluación perezosa* para dejar definido totalmente el método de evaluación automática.

5.3. Cadenas de evaluación y evaluación *perezosa*

En esta sección se presenta cómo se realizan las evaluaciones de los nodos operacionales cuando ocurre una transformación en el grafo de eva-

luación. Este procedimiento evita la repetición innecesaria de operaciones que pudieran aparecer como consecuencia de las *cadenas de evaluación*, que se definen a continuación.

La evaluación de un nodo operacional puede implicar que se deba evaluar también otro nodo. Por ejemplo, en la instancia del CVRP con la que se ha trabajado en este documento, al extraer al cliente 1 de la primera ruta y desevaluar al nodo *incrementa demanda de la ruta*, se hace necesario reevaluar el nodo *penaliza exceso de carga*, debido a que la carga necesaria para atender la ruta ha disminuido.

A continuación se definen las *cadenas de evaluación* y se verá cómo pueden afectar la eficiencia del método de evaluación automática descrito hasta el momento.

Definición 11 *A la sucesión $o_1, o_2, \cdots, o_n$ de nodos operacionales tal que si se evalúa o desevalúa o_i, se modifica la entrada de o_{i+1} para $1 \leq i \leq n-1$, se le denomina cadena de evaluación de tamaño n.*

En el ejemplo expuesto los nodos *incrementa demanda del cliente* 1 y *penaliza exceso de carga de la ruta* 1 forman una cadena de evaluación de tamaño dos.

Si el nodo que representa al cliente 1 se extrae, el nodo *incrementa demanda* debe ser desevaluado, esto implica una modificación en la entrada del nodo *penaliza exceso de carga* por lo que este último tendría que ser reevaluado. No obstante, si el nodo extraído se inserta luego en la misma ruta, entonces se evalúa al nodo *incrementa demanda* y nuevamente se debe reevaluar el nodo *penaliza exceso de carga*, con lo que la primera evaluación del penalizador se podía haber dejado de realizar sin que esto tuviera ninguna consecuencia sobre la evaluación final.

Para evitar las evaluaciones de nodos que no son necesarias se utiliza la *evaluación perezosa*. Para utilizar este procedimiento es necesario que cada método *evalúa* y *desevalúa* de los nodos operacionales devuelva la operación que le sigue en la cadena de evaluación a la que pertenecen o un valor nulo en caso de que dicha operación no exista.

Con la *evaluación perezosa* cada vez que se evalúa o desevalúa un nodo operacional, en vez de reevaluar todas las operaciones que le siguen en la cadena, se almacena la operación que retorna el método *evalúa* o *desevalúa* del nodo (la próxima en la cadena) en una cola de operaciones, a menos que esta operación ya se encontrara previamente en la cola, en cuyo caso no se reinserta sino que su posición en la cola pasa a ser la última.

El ejemplo de la sección anterior, para cambiar al cliente de la primera posición de la primera ruta a la segunda posición de la primera ruta sin usar

la evaluación perezosa se realizan las siguientes operaciones en el mismo orden en que se muestran.

- Desevalúa incrementa distancia entre depósito y cliente 1
- Desevalúa incrementa distancia entre cliente 1 y cliente 2
- Desevalúa incrementa demanda del cliente 1
- Evalúa penaliza demanda total de la ruta 1
- Evalúa incrementa distancia entre depósito y cliente 1
- Desevalúa incrementa distancia entre el cliente 2 y el depósito
- Evalúa distancia entre cliente 2 y cliente 1
- Evalúa distancia entre cliente 1 y el depósito
- Evalúa incrementa demanda de cliente 1
- Evalúa penaliza demanda total de la ruta 1

Se puede apreciar que se evalúa la operación *penaliza* demanda de la ruta en dos ocasiones. Si esta evaluación solo se realizara al final, el resultado de la evaluación de la nueva solución no se hubiese alterado.

El ejemplo anterior con la evaluación perezosa se realizaría de la siguiente manera. La cola inicialmente se llena con las primeras evaluaciones y desevaluaciones que deben realizarse por la transformación, en este caso serían

- Desevalúa incrementa distancia entre depósito y cliente 1
- Desevalúa incrementa distancia entre cliente 1 y cliente 2
- Desevalúa incrementa demanda del cliente 1
- Evalúa incrementa distancia entre depósito y cliente 1

Todas estas son las que se realizan debido a la extracción del cliente 1. Si el cliente luego se va a insertar en la misma ruta pero justo después del cliente 2 entonces se adicionan también las operaciones.

- Desevalúa incrementa distancia entre el cliente 2 y el depósito

- Evalúa incrementa distancia entre cliente 2 y cliente 1
- Evalúa incrementa distancia entre cliente 1 y el depósito
- Evalúa incrementa demanda de cliente 1

La cola se inicializa entonces con todas las operaciones anteriores.

- Desevalúa incrementa distancia entre depósito y cliente 1
- Desevalúa incrementa distancia entre cliente 1 y cliente 2
- Desevalúa incrementa demanda del cliente 1
- Evalúa incrementa distancia entre depósito y cliente 1
- Desevalúa incrementa distancia entre el cliente 2 y el depósito
- Evalúa incrementa distancia entre cliente 2 y cliente 1
- Evalúa incrementa distancia entre cliente 1 y el depósito
- Evalúa incrementa demanda de cliente 1

Estas son las operaciones que se hacen directamente cuando se extrae y se inserta al cliente. La operación *penaliza* no se incluye en esta cola inicial, porque no forma parte de las operaciones que se deben realizar directamente al extraer o insertar, sino que es consecuencia de la operación *incrementa demanda*.

Las dos primeras operaciones de la cola (*incrementa distancia*) no tienen ninguna otra operación que les suceda en una cadena de evaluación. Esto quiere decir que al ejecutarlas no se modifica la entrada de ninguna otra operación y, por tanto, no devuelven una nueva operación. Por tanto la cola queda con la siguiente estructura luego de ejecutar sus dos primeras operaciones.

- Desevalúa incrementa demanda del cliente 1
- Evalúa incrementa distancia entre depósito y cliente 1
- Desevalúa incrementa distancia entre el cliente 2 y el depósito
- Evalúa incrementa distancia entre cliente 2 y cliente 1
- Evalúa incrementa distancia entre cliente 1 y el depósito

- Evalúa incrementa demanda de cliente 1

Cuando se desevalúa la operación *incrementa demanda*, cambia el valor de la *demanda total de la ruta* que es entrada de la operación *penaliza*. Por tanto, al ejecutar *desevalúa incrementa demanda* se obtiene como resultado una nueva evaluación para insertar en la cola que es precisamente la de la operación *penaliza*. Como esta última no se encuentra aún en la cola, se inserta al final de la misma. La cola queda entonces de esta forma:

- Evalúa incrementa distancia entre depósito y cliente 1
- Desevalúa incrementa distancia entre el cliente 2 y el depósito
- Evalúa incrementa distancia entre cliente 2 y cliente 1
- Evalúa incrementa distancia entre cliente 1 y el depósito
- Evalúa incrementa demanda de cliente 1
- Evalúa penaliza demanda total de la ruta 1

Todas las operaciones hasta la penúltima del listado anterior se realizan sin cambios que resaltar hasta que en la cola solo quedan:

- Evalúa incrementa demanda de cliente 1
- Evalúa penaliza demanda total de la ruta 1

Al evaluar la operación *incrementa demanda* se vuelve a obtener la operación *penaliza* debido a que esta le sigue en la cadena de evaluación. Pero ya esta operación está insertada en la cola. Por tanto lo que se hace es cambiar su posición para el final de la cola, en este caso ya está en esa posición.

A modo de resumen, el algoritmo de evaluación perezosa comienza inicializando una cola con las operaciones que se deben realizar cuando ocurre una transformación de la solución. Con la cola de operaciones inicializada se procede a realizar las operaciones en orden y el resultado de cada operación (la siguiente operación de la cadena) se adiciona al final de la cola si no es nulo. Si ya se había adicionado a la cola con anterioridad entonces se extrae de la posición en la que estaba y se coloca al final. La evaluación termina cuando la cola queda vacía.

Procediendo de la manera que establece la *evaluación perezosa* no se realiza ninguna operación más de una vez y cada operación se realiza cuando

todas las posibles operaciones que pueden antecederle en una cadena se han realizado.

Con el algoritmo de *evaluación perezosa* queda definido cómo se realizan las evaluaciones y desevaluaciones cuando se quiere obtener el valor de la función objetivo en una solución vecina y, en general, ha quedado definida la evaluación automática de una solución vecina.

Existen variantes de VRP a las cuales no se les puede aplicar aún la propuesta del presente trabajo. En la siguiente sección se presentan las limitaciones de esta implementación.

5.4. Limitaciones de la propuesta

En esta sección se presentan las variantes de VRP en las cuales no es posible aplicar la evaluación automática de soluciones vecinas tal y como se ha implementado en el presente trabajo. Para ello se tomará como ejemplo el Problema de Enrutamiento de Vehículos con Entregas y Recogidas Simultáneas que es una de las variantes en las que es imposible aplicar la evaluación automática.

En el Problema de Enrutamiento de Vehículos con Entregas y Recogidas Simultáneas a cada cliente se le entrega pero también se le recoge cierta cantidad de producto. En este caso, además, el cliente es visitado una sola vez, de ahí el nombre de la variante, ya que la recogida y la entrega se realizan al mismo tiempo y por el mismo vehículo. Como consecuencia, es posible que se incumpla la restricción de capacidad sin que la cantidad total de producto que se debe entregar a cada cliente de la ruta exceda la capacidad del vehículo. Por ejemplo, puede que el exceso de carga lo provoque la recogida de productos al atender a un cliente durante el recorrido de la ruta.

En esta variante se penaliza el exceso de carga al visitar cada cliente de la ruta, al contrario de como se procedía en la evaluación de una solución de CVRP donde la penalización se realiza al finalizar el recorrido de cada ruta solamente. A los problemas en donde la función objetivo incluya penalizaciones durante el recorrido de una ruta no es posible aplicarles la evaluación automática tal y como está implementada en el presente trabajo.

Las operaciones de penalización reciben variables como entrada. En el caso del CVRP la penalización se realizaba una vez que la variable de entrada *demanda-total* contenía su valor final. Cuando el problema incluye entregas y recogidas simultáneas se penaliza tomando como entrada esta misma variable pero con la diferencia de que luego de realizar la penalización di-

cha variable se puede modificar nuevamente. Es por eso que, a pesar de que se puede obtener un grafo de evaluación y el resultado de la evaluación de la solución inicial, no sucede lo mismo con las soluciones vecinas de esta. Por ejemplo, al extraer un cliente no se conoce el valor que contenía la variable *demanda-total* cuando en la evaluación de la solución inicial se visitó a ese cliente y, por tanto, no se puede revertir la penalización que se pudo haber aplicado en ese momento.

En general no es posible aplicar la evaluación automática a problemas en los que se realicen operaciones durante el recorrido de una ruta que dependan de variables que se modifiquen en ese recorrido.

No obstante existen varias instancias de VRP en las que se puede aplicar la propuesta del presente trabajo. Para utilizar el método de evaluación automática en una nueva variante de VRP se deben definir los nuevos nodos que son necesarios, los métodos de evaluación y desevaluación para los nodos operacionales y las nuevas operaciones que componen a los criterios de vecindad que se deseen aplicar a las soluciones. En el siguiente capítulo se mencionan las variantes de VRP a las que se les ha adicionado el método de evaluación automática que se expuso en el presente trabajo.

Capítulo 6

Extensibilidad

En este capítulo se presentan varias instancias de VRP a los que se les ha aplicado la evaluación automática propuesta en este trabajo y se muestran los pasos necesarios para agregar nuevas variantes.

En este documento se ha ilustrado el método de la evaluación automática con un ejemplo de Problema de Enrutamiento de Vehículos con Restricción de Capacidad (CVRP). Esta es una de las variantes en la que se ha podido aplicar la evaluación automática. Las otras variantes de VRP a las que se les ha aplicado este método son el Problema de Enrutamiento de Vehículos con Flota Heterogénea Infinita (HFVRP) y el Problema de Enrutamiento de Vehículos con Múltiples Depósitos (MDVRP). Al implementar estas variantes también se tienen implementadas sus posibles combinaciones, por ejemplo, el Problema de Enrutamiento de Vehículos con Flota Heterogénea Infinita y Múltiples Depósitos. Esto se debe a que los nodos que representan a cada uno de los elementos de estos problemas están representados por clases y con tener clases que hereden de las anteriores ya se tiene lo necesario para aplicar la evaluación automática.

A continuación se presenta cómo se debe proceder para aplicar la evaluación automática a un Problema de Enrutamiento de Vehículos con Flota Heterogénea e Infinita (HFVRP)

6.1. Método para aplicar la evaluación automática al Problema de Enrutamiento de Vehículos con Flota Heterogénea e Infinita

En la presente sección se ejemplifica cómo aplicar la evaluación automática en la solución de un HFVRP. En este caso existen vehículos que se diferencian en su capacidad.

En este problema se deben cumplir las mismas restricciones que en el CVRP, o sea, cada ruta debe ser atendida por un vehículo y se penaliza el exceso de demanda de una ruta con respecto a la capacidad del vehículo que recorra dicha ruta. El código de evaluación de una solución de HFVRP se presenta en la figura 6.1.

```
1  evalúa(S):
2    new_var distancia_total = 0
3    para toda ruta r de S:
4        new_var demanda_ruta = 0
5        new_var cliente_previo = deposito(r)
6        para todo cliente c en r:
7            incrementa distancia_total con...
                       d(cliente_previo, c)
8            incrementa demanda_ruta con demanda de c
9            cliente_previo = c
10       incrementa distancia_total con...
              d(c, depósito)
11       incrementa distancia_total con...
              penalizar(demanda_ruta,
                  capacidad del vehículo de la ruta r)
12    devuelve distancia_total
```

Figura 6.1: Código de evaluación de una solución de un HFVRP. La operación `d(x,y)` representa la distancia entre `x` e `y`.''

Además de cambiar clientes de posición, en este problema también tiene sentido cambiar el vehículo que atiende una ruta. Es por eso que se debe incluir esta transformación para obtener soluciones vecinas.

Por tanto, aplicar la evaluación automática para resolver este problema implica definir nuevos nodos en el grafo para representar vehículos y

nuevas transformaciones sobre el grafo de evaluación.

Un nodo de *entrada/salida* en el grafo de evaluación está representado por una clase. En este caso se crea una nueva clase para representar un nodo vehículo. Esta clase tiene la capacidad de dicho vehículo y las operaciones a las que está conectado. En este caso esa operación sería la de penalizar, que recibe además la demanda total de la ruta y en base a dicha demanda y a la capacidad del vehículo aplica la penalización.

En un HFVRP se tiene una lista de todos los tipos de vehículos, por lo que se puede realizar la transformación *cambiar al vehículo de la ruta r por el que está en la posición i de la lista.* Para adicionar esta transformación se debe implementar un método que la realice. Al cambiar el vehículo de una ruta se debe cambiar el nodo que representa a dicho vehículo en la ruta por el que le corresponde al nuevo vehículo. Al realizar este cambio se procede entonces a reevaluar todas las operaciones que dependen del nodo que representa al vehículo de la ruta. En este caso es solo la operación *penaliza demanda de la ruta* debido a que la capacidad del vehículo ha cambiado. Por tanto el método que se debe implementar para agregar la operación *cambiar vehículo* recibe el índice de la ruta a la cual se le realizará el cambio y el índice del nuevo vehículo en la lista de vehículos disponibles. En este método se cambia el nodo vehículo de la ruta por el nodo correspondiente y se reevalúan todas las operaciones de las cuales es entrada el nodo vehículo, en este caso sería la operación *penaliza*.

A continuación se presenta un listado con los nodos de *entrada/salida* que componen un grafo de evaluación de una solución de un HFVRP si esta se evalúa con el código de la figura 6.1

- Nodo depósito
- Nodo cliente
- Nodo vehículo
- Nodo demanda de la ruta (variable)
- Nodo distancia total (variable y resultado final)

En este caso la única diferencia con el CVRP es la adición del nodo vehículo.

Las operaciones que se realizan en la evaluación de la solución (nodos operacionales) son las mismas que en el CVRP

- Incrementa distancia total

- Incrementa demanda de la ruta
- Penaliza demanda de la ruta

Y las transformaciones que se pueden realizar en el grafo son

- Cambiar un cliente de posición
- Cambiar el vehículo de una ruta

Con todos estos elementos se puede aplicar la evaluación automática al HFVRP. A continuación se comenta lo que se debería realizar para usar la evaluación automática en el Problema de Enrutamiento Vehículos con Múltiples Depósitos.

6.2. Comentarios sobre cómo extender la propuesta para resolver el Problema de Enrutamiento de Vehículos con Múltiples Depósitos y otras variantes

En el caso del Problema de Enrutamiento de Vehículos con Múltiples Depósitos (MDVRP) se tiene una lista de depósitos disponibles y las rutas pueden comenzar y terminar en depósitos distintos. Se cuenta, además, con la información de la distancia entre todos los cientes y todos los depósitos. Por este motivo es necesario implementar las operaciones *cambiar el primer depósito de una ruta* y *cambiar el último depósito de una ruta*. Estas operaciones pueden realizarse del mismo modo que se realizan las inserciones y extracciones de clientes, con la diferencia de que el nuevo depósito que se inserta se toma de la lista de depósitos disponibles.

En los casos de MDVRP y HFVRP no es necesario agregar nuevas operaciones a la evaluación de una solución. Con las operaciones que se tenían para el CVRP (*incrementar distancia total* e *incrementar demanda de la ruta*) se pueden evaluar las nuevas soluciones. No obstante, si una variante exige que se implementen nuevas operaciones esto sería posible. En ese caso se tendrían que agregar los nodos operacionales correspondientes, además de los métodos que se utilizarían en el código de evaluación para estas operaciones, o sea, aquellos que construirían el segmento del grafo correspondiente a partir de la evaluación de la solución.

Como se ha expuesto en los párrafos anteriores, el proceso para agregar la evaluación automática a las diversas variantes de VRP consiste en definir los nodos que se deben agregar al grafo de evaluación, las nuevas

operaciones que se ejecutan durante la evaluación de una solución y las modificaciones que se realizan en este grafo debido a los nuevos criterios de vecindad.

Por ejemplo, para implementar el HFVRP fue necesario agregar un nuevo nodo de *entrada/salida* que representara al vehículo de la ruta. Además, debido a que en este problema se pueden cambiar los vehículos de las rutas, fue necesario implementar cómo se modifica el grafo de evaluación cuando ocurre uno de estos cambios. En este caso, esa modificación consiste en cambiar el nodo del vehículo anterior de la ruta por el del nuevo vehículo y reevaluar todas las operaciones de las que ese nodo es entrada (operación *penalizar* para el ejemplo que se expuso).

Para extender la aplicación de la evaluación automática a las soluciones de otras varianes de VRP se deben seguir los pasos que se expusieron en el presente capítulo.

A continuación se presentan las conclusiones y recomendaciones del presente trabajo.

Conclusiones y recomendaciones

En el presente trabajo se definió el proceso de evaluación automática de soluciones vecinas en un Problema de Enrutamiento de Vehículos basado en un grafo de evaluación. Con este método se logra evaluar las soluciones de una vecindad de manera eficiente y sin que se tenga que escribir código para tal fin.

Para alcanzar este objetivo se definió la estructura *grafo de evaluación*, que es una representación de la evaluación de una determinada solución en la función objetivo. A partir de este grafo se puede obtener el valor de cualquier solución vecina al ser evaluada en la función objetivo.

Se definió además el proceso de *evaluación perezosa* que evita que se repitan operaciones innecesarias en el momento de evaluar la nueva solución. Se estableció una limitación que tiene la evaluación automática que no permite que se pueda aplicar a problemas en los que se realicen penalizaciones durante el recorrido de una ruta. Finalmente se describió cómo el método de evaluación automática puede ser extendido a otras variantes del VRP con los nuevos criterios de vecindad que se pueden aplicar en cada una.

Como recomendaciones y trabajos futuros se propone implementar la evaluación automática de soluciones vecinas para otras variantes del VRP como el Problema de Enrutamiento de Vehículos con Entregas y Recogidas Simultáneas (VRPSPD) y el Problema de Enrutamiento de Vehículos con Ventanas de Tiempo (TWVRP) donde se aplican penalizaciones durante el recorrido de una ruta. Con cada nueva variante que se adicione se obtienen las implementaciones para cualquier combinación de esa nueva variante con todas las anteriores.

Se recomienda investigar sobre la aplicación de la evaluación automática de soluciones vecinas a los algoritmos genéticos para poder obtener automáticamente el resultado de los cruzamientos entre soluciones y el costo

de estos resultados sin tener que programarlo.

Las soluciones a diversas variantes de VRP se han implementado en múltiples lenguajes [4, 8, 9]. Por tanto, diseñar una biblioteca que genere el código de la evaluación automática en cualquier lenguaje de programación permitiría que el método propuesto en el presente trabajo se aplique en soluciones ya existentes. Esa sería una extensión atractiva de este trabajo.

Bibliografía

[1] J. H. Dantzig, G. B.; Ramser. The truck dispatching problem. Management Science, 6, 10 1959. (Citado en la página 5).

[2] Paolo Toth; Daniele Vigo. *The Vehicle Routing Problem (Monographs on Discrete Mathematics and Applications)*. Monographs on Discrete Mathematics and Applications. SIAM, 2001. (Citado en las páginas 1, 5 y 15). (Citado en las páginas 1 y 7).

[3] Jorge Rodríguez Pérez. *Caracterización, modelado y determinación de las rutas de la flota en una empresa de rendering*. Master's thesis, 2012. (Citado en la página 5).

[4] Camila Pérez Mosquera. *Primeras aproximaciones a la Búsqueda de Vecindad Infinitamente Variable*. Trabajo de diploma. MATCOM, Universidad de La Habana. 2017. (Citado en las páginas 1, 7 y 44).

[5] Daniela González Beltrán. *Generación automática de gramáticas para la obtención de infinitos criterios de vecindad en el problema de enrutamiento de vehículos*. Trabajo de diploma. MATCOM, Universidad de La Habana. 2019. (Citado en la página 9).

[6] Anand; Ochi Luiz Satoru Penna, Puca Huachi Vaz; Subramanian. *An iterated local search heuristic for the heterogeneous fleet vehicle rou- ting problem.* Journal of Heuristics, 19, 04 2013. (Citado en la página 7).

[7] Leonora Bianchi; Ann Melissa Campbell. *Extension of the 2-p-opt and 1-shift algorithms to the heterogeneous probabilistic traveling sales- man problem.* European Journal of Operational Research, 176, 2007. (Citado en la página 7).

[8] Dalia Díaz Sistachs. *Estrategia de Hibridación Heurística para la solución del Problema de Entrega y Recogida Simultánea con Trasbordo y Limitación*

de Tiempo. Tesis de Maestría. MATCOM. Universidad de La Habana. 2018. (Citado en la página 44).

[9] Alina Fernández Arias. *Modelos Y métodos para el Problema de enrutamiento de vehículos con recogida y entrega simultánea.* PhD thesis, Fa- cultad de Matemática y Computación. Universidad de La Habana, La Habana, Cuba., 4 2017. (Citado en la página 44).