Resumen

El problema de enrutamiento de vehículos (VRP) posee gran importancia académica e industrial. La cantidad de variantes que pueden existir es virtualmente infinita, limitada sólo por la imaginación humana. El tiempo necesario para resolver una de estas variantes es, comunmente, entre seis meses y dos años; modificado por las especificaciones involucradas.
Se propone una herramienta para resolver cualquier variante de VRP en un estimado de dos semanas utilizando algoritmos de búsqueda local. Se unen las implementaciones del grafo de evaluación, el árbol de vecindad, la exploración de dos fases y la combinación de estrategias de exploración y selección para resolver los problemas de forma eficiente.

Introducción

Los costos por transportación pueden representar hasta el 60 porciento de los costos logísticos de una empresa. Esto implica que para empresas con capitales millonarios, una buena planeación de las rutas de transportación puede marcar una diferencia igualmente millonaria.

El problema de enrutamiento de vehículos (VRP por sus siglas en inglñés) es un problema de optimización combinatoria cuyo objetivo en su forma más simple es, dado un conunto de clientes, un depósito y una flota de vehículos, encontrar una asignación de rutas que optimice ciertos criterios tales como tiempo y costos de transportación.

Existen además numerosas variaciones del problema clásico tales como el Problema de Enrutamiento de Vehículos con restricciones de capacidad (CVRP), el Problema de Enrutamiento de Vehículos con ventanas de tiempo (VRPTW) y el Problema de Enrutamiento de Vehículos con recogida y entrega (VRPPD). Cada variante tiene especificaciones propias, por lo que resulta difícil la creación de un método de solución universal para la familia de problemas VRP.

El hecho de ser problemas NP-Duros implica la falta de soluciones exactas óptimas para instancias no pequeñas, por tanto, se utilizan técnicas no exactas como heurísticas y metaheurísticas que han sido objeto de estudio por décadas.

El proceso de solución de una instancia arbitraria de VRP es complejo y necesita de una considerable cantidad de tiempo. Siendo un problema de gran importancia tanto académica como industrial, ha inspirado la creación de numerosas herramientas y artículos científicos. Cabe destacar la biblioteca OR-Tools, un software de código abierto útil para resolver problemas de optimización combinatoria entre los que se encuentra VRP.

En la facultad de Matemática y Computación de La Universidad de La Habana este ha sido tema de estudio desde hace unos 6 años y se ha logrado resolver diversas problemáticas. En particular, la metaheurística de búsqueda local infinitamente variable (IVNS) planteada por Camila Pérez

en [4], la generación automática de gramáticas para IVNS hecha por Daniela Gonzáles en [1], la exploración de vecindades a partir de la combinación de distintas estrategias de exploración y selección hecha por Heidy Abreu en [3], el Árbol de vecindad y la exploración de dos fases fueron planteados por Héctor Massón en [5] y el Grafo de evaluación para la evaluación automática y eficiente de soluciones creado por Jose Jorge Rodríguez en [6]. Cada una resuelve por separado un problema distinto.

Hasta el momento, para resolver una instancia de VRP por búsqueda local se le debe invertir muchísimo tiempo a programar aspectos como la forma de evaluar vecinos o de explorar vecindades. A partir de la unión de las ideas anteriores es posible resolver un VRP únicamente programando la evaluación de una solución.

Objetivos

El objetivo de este trabajo es diseñar e implementar un sistema que permita usar los beneficios de años anteriores de investigación para resolver cualquier VRP a partir de las características específicas del problema y el código de evaluación de una solución.

Objetivos específicos

1. Consultar literatura especializada sobre el estado del arte de los problemas VRP.

2. Entender a profundidad las ideas y códigos propuestos en tesis anteriores.

3. Diseñar y programar un sistema que combine estas ideas para resolver cualquier VRP a partir de la evaluación de una solución.

4. Analizar los resultados obtenidos. Se utiliza el sistema para resolver instancias conocidas de problemas de VRP.

Organización de la tesis

El presente documento está organizado en 4 capítulos.

En el capítulo 1 **Preliminares** se describe a profundidad la familia de Problemas de Enrutamiento de Vehículos, se introducen las vías existentes (bibliotecas) para resolverlos, se describen las ideas desarrolladas en cada

una de las tesis anteriores y se hace una breve descripción de Coommon Lisp y algunas de las funcionalidades utilizadas.

El capítulo 2 **Propuesta de Solución** describe el sistema implementado y la forma en que fueron unidas las piezas que lo conforman.

En el capítulo 3 **Método de uso** se describe paso a paso el método de uso el sistema y cómo describir y resolver instancias de VRP.

El capítulo 4 **Experimentos y resultados** comprende los experimentos realizados para validar el modelo, así como las métricas destinadas para su evaluación.

Por último se ofrecen las conclusiones a partir de los objetivos propuestos y los resultados alcanzados. Adicionalmente se brindan algunas ideas y recomendaciones para trabajos futuros.

Capítulo 1

Preliminares

Este trabajo pretende implementar un sistema que resuelva instancias de VRP en cualquiera de sus variantes con el menor trabajo humano posible. En este capítulo se presentan los principales elementos de la investigación realizada para lograrlo.

Primeramente se comienza con una visión general del problema de enrutamiento de vehículos (VRP) y algunas de sus variantes con la sección 1.1. Se explica cómo describir a partir de código una solución de VRP.

En 1.2 se muestran las bibliotecas de clases existentes hasta el momento que pueden ser utilizadas para encontrar soluciones a instancias de VRP.

En 1.4 se explica cómo crear un Árbol de vecindad a partir de una solución inicial y un criterio de vecindad. Este árbol es utilizado para obtener la cardinalidad de vecindades y realizar una exploración de dos fases que utiliza técnincas estadísticas.

En 1.5 se expone el concepto de Grafo de evaluación y cómo es esto utilizado para evaluar soluciones de forma eficiente y automática.

En 1.3 se presenta un mecanismo para explorar vecindades de forma automática a partir de combinaciones de cualesquiera estrategias de exploración y selección.

Finalmente en 1.6 se describe brevemente algunas características y funcionalidades del lenguaje Common Lisp que resultaron especialmente útiles para el desarrollo del sistema.

1.1. Problema de Enrutamiento de Vehículos

La primera referencia al VRP fue hecha por Dantzing y Ramser en [2] en el año 1959. Se propone una formulación matemática, una aproximación

algorítmica y se describe una aplicación real entregando gasolina a varias estaciones de servicio.

En su versión más simple, el problema consta de una flota de vehículos que salen de un depósito y deben satisfacer las demandas de una serie de clientes. El objetivo es encontrar una distribución de caminos a asignar a los vehículos de forma que se optimice determinada métrica (tiempo, combustible, etc). Con más de 50 años de estudios se ha ramificado en una inmensa cantidad de variantes entre las que se pueden contar las siguientes:

- CVRP - VRP con restricciones de capacidad. Cada vehículo tiene una capacidad que no debe ser excedida.
- VRPTW - VRP con ventanas de tiempo. Cada cliente posee un período de tiempo fijo durante el cual puede ser atendido.
- VRPPD - VRP con recogida y entrega. Los bienes deben ser entregados y recogidos en cantidades fijas.
- MDVRP - VRP con múltiples depósitos. Se cuenta con múltiples depósitos desde los que pueden salir los vehículos.

Esta es una familia de problemas NP-Duros, por lo las soluciones exactas no son factibles para instancias de grandes tamaños. Para buscar aproximaciones a la solución se utilizan heurísticas y metaheurísticas. Se destaca la búsqueda local como metaheurística que ha dado muy buenos resultados y es la seleccionada en el presente trabajo como se explica en 1.1.2.

1.1.1. Representación de soluciones del VRP

Las soluciones son representadas (en su versión más simple) como una serie de listas de clientes denominadas rutas. Si se define a P_1 como un problema clásico que consta de 6 clientes: $[c_1, c_2, c_3, c_4, c_5, c_6]$, entonces una solución s_1 se puede definir como:

$$s_1 = [(c_2, c_3), (c_1, c_4, c_5), (c_6)] \tag{1.1}$$

En s_1 se representa una solución con tres rutas. El vehículo perteneciente a la primera (r_1) ruta visita a c_2 y c_3, el vehículo de la segunda ruta (r_2) visita a c_1, c_4 y c_5 y el de la tercera (r_3) sólo visita a c_6.

1.1.2. Metaheurísticas de búsqueda local

Los algoritmos basados en búsquda local son aquellos en que se define una solución inicial y a partir de determinado criterio de vecindad se busca la solución óptima iterando por los vecinos de la vecindad formada por dicho criterio. A continuación se muestran algunos ejemplos de criterios de vecindad:

1. Cambiar de posición a un cliente dentro de su ruta.
2. Mover a un cliente de ruta.
3. Intercambiar dos clientes de posición.
4. Cambiar vehículo de ruta.
5. intercambiar dos subrutas entre sí.
6. invertir orden de una subruta.

Los criterios de vecindad dependen también de la variante del problema sobre la que se trabaje. Por ejemplo, el criterio de **Çambiar vehículo"** no tiene sentido para el problema P_1 pues en este todos los vehículos son iguales.

Estas operaciones pueden ser obtenidas a partir de un subconjunto de operaciones más simples a las que se denomina operaciones elementales. Entre las operaciones elementales se encuentran: **selección de ruta**, **selección de cliente** e **inserción de cliente**.

Por ejemplo, **intercambiar dos clientes de posición** puede ser realizado a partir de dos **selección de ruta**, dos **selección de cliente** y dos **inserción de cliente**.

La importancia de las operaciones elementales para la implementación del sistema se explicarán en con más profundidad en **??**.

A partir de un criterio y una solución inicial se pueden generar nuevas soluciones. El proceso de obtención y selección de nuevas soluciones se denomina exploración de la vecindad. Mientras más grande la vecindad es más probable encontrar mejores soluciones, pero esto puede requerir una gran cantidad de tiempo. Por tanto, las estrategias a seguir durante la exploración son un factor vital a tener en cuenta para la implementación del sistema y se explicarán en 1.4 y 1.3. Además, a la hora de comparar el costo entre dos soluciones es necesario evaluarlas, lo cual posee también un costo computacional considerable. Para la evaluación de soluciones se tiene el Grafo de evaluación explicado en 1.5.

1.2. Vías de solución existentes

Una buena herramienta para resolver problemas de VRP en la actualidad es la biblioteca OR-Tools (Google Optimization Tools). Un software de código abierto útil para problemas de optimización combinatoria entre los que se encuentra el Problema de Enrutamiento.

NOTA PARA FERNANDO: Profe, no sé qué más poner aquí XD.

1.3. Combinación de estrategias de exploración y selección.

TODO

1.4. Árbol de vecindad y exploración de dos fases

La exploración de vecindades puede ser ineficiente y difícil de programar. La idea que se propone es utilizar técnicas estadísticas para analizar cuáles son las mejores regiones de las vecindades para intensificar la búsqueda en estas.

Buscando aplicar dichas técnicas estadísticas es necesario saber la cardinalidad de las vecindades y separarlas en regiones. Para lograr esto de forma eficiente (sin iterar por todos los elementos de una vecindad) se propone la creación de un Árbol de vecindad. El Árbol de vecindad utiliza oun concepro de *solucion* diferente al explicado en 1.1.1. Previamente se definió una solución de VRP en su forma más trivial como una lista de caminos conformados a su vez por la lista de clientes que se visitan en cada uno, por ejemplo:

$$s_1 = [(c_2, c_3), (c_1, c_4, c_5), (c_6)] \tag{1.2}$$

Con el propósito de contar la cantidad de soluciones que tiene una vecindad, pierde importancia saber qué clientes se visita en cada ruta, en cambio sólo es necesario conocer la cantidad de clientes visitados en cada una. A este tipo de solución se le denomina solución de conteo y aplicado a s_1 daría como resultado:

$$sc_1 = [2, 3, 1] \tag{1.3}$$

Teniendo una solución inicial, todas las soluciones de una vecindad pueden ser obtenidas aplicando sobre esta el criterio de vecindad en cuestión, con todos sus valores posibles. A la asignación de valores de un criterio de vecindad sobre una solución se le llamará una instanciación de dicho criterio. Por ejemplo, dado el criterio **mover cliente** dado por:

- Seleccionar ruta (r).
- Seleccionar cliente en ruta (a).
- Seleccionar ruta (r).
- Insertar cliente en ruta (b).

Las letras r, a y b son los símbolos representativos de cada operación.

Un ejemplo de instanciación de este criterio sobre la solución s_1 se presenta como:

- $r_1 = 1 \rightarrow r_1$ es la ruta de la que el cliente c_1 es extraído.
- c_1.position = $1 \rightarrow c_1$ es el primer cliente de la ruta.
- $r_2 = 1 \rightarrow r_2$ es la ruta de la que el cliente c_1 será insertado.
- c_1.position = $2 \rightarrow c_1$ es insertado en la posición 2 de la ruta seleccionada.

A cada vecino de un criterio se le puede hacer corresponder una instanciación del mismo y por tanto, encontrar la cardinalidad de una vecindad es similar a encontrar el número de instanciaciones posibles del criterio asociado.

Al generar las distintas instanciaciones de un criterio de vecindad para una solución dada se cumple que muchas de estas comparten una secuencia común de operaciones instanciadas. La estrategia propuesta se basa en agrupar aquellos criterios instanciados para los cuales dicha secuencia común comience en la primera operación de los mismos, pues de esta forma el resto de las operaciones de tales criterios instanciados no se ven afectadas, y contar para cada una de estas el número de posibles secuencias de operaciones instanciadas que unidas con la secuencia común forman un criterio instanciado.

Al computar la cardinalidad de una vecindad del VRP, se utiliza una estrategia recursiva que consiste en contar para una operación el número

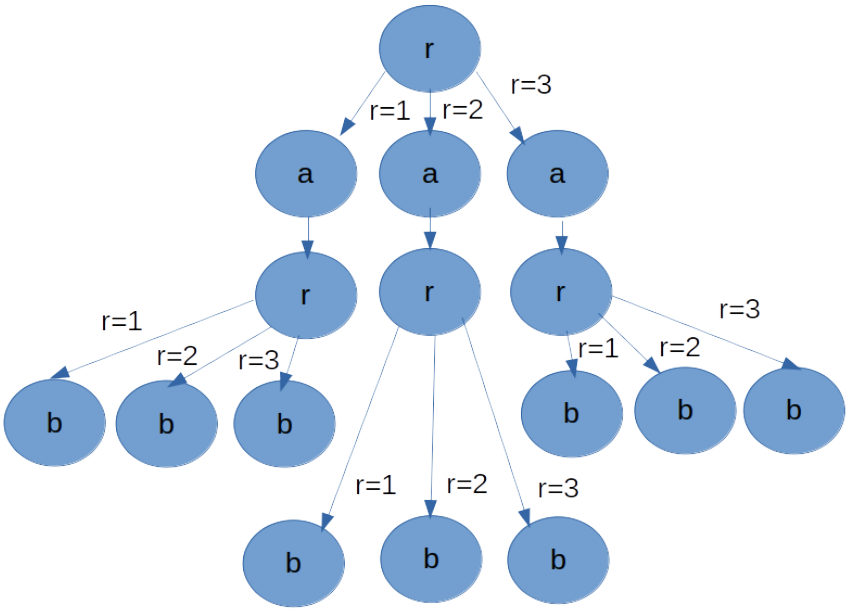

Figura 1.1: Árbol de vecindad asociado a *rarb*

de secuencias de operaciones instanciadas que se pueden formar con el resto de las operaciones del criterio, así como combinar las mismas con las posibles instanciaciones de la operación actual aprovechando que para las operaciones modificadoras estas secuencias son comunes a todas las posibles instanciaciones de las mismas.

Es posible utilizar este algoritmo, para almacenar toda la información necesaria para cualquier procesamiento posterior sobre dicha vecindad en una estructura arbórea que será llamada árbol de vecindad y que constituye una representación de la vecindad en cuestión. En 1.1 se muestra una representación del árbol de vecindad asociado a l criterio **mover cliente** (*rarb*). Cada nodo del árbol, representa una operación de vecindad y almacena toda la información necesaria para instanciar dicha operación y de esta forma, a partir del proceso que computa la cardinalidad de la vecindad, se pueden pasar a generar todas las soluciones de la misma.

Cada rama del árbol representa un conjunto de soluciones, por tanto, el conjunto de ramas representa una partición de la vecindad a la que está

asociado dicho árbol. A los conjuntos hechos por esta partición se les denomina regiones y resultan útiles para encontrar características compartidas en grupos de vecindades. Por ejemplo, es conveniente analizar cuáles son las regiones con mejores soluciones para intensificar en estas la búsqueda de soluciones óptimas.

La exploración en dos fases es una heurística útil para explorar una vecindad de un VRP realizando el análisis de la misma en dos etapas. Una primera, llamada de exploración, donde se intenta construir una muestra representativa de la vecindad en cuestión, y otra etapa posterior de explotación (o de intensificación), en la cual se exploran las regiones más prometedoras de la vecindad que fueron identificadas mediante el uso de ténicas estadísticas sobre la muestra generada en la fase anterior.

1.5. Grafo de Evaluación

TODO

1.6. Common Lisp y sus funcionalidades

Common Lisp es un lenguaje de programación multi paradigma (soporta una combinación de paradigmas de programación tales como la programación imperativa, funcional y orientada a objetos). Facilita el desarrollo de software evolutivo e incremental, con la compilación iterativa de programas eficientes en tiempo de ejecución.

NOTA PARA FERNANDO: Tampoco sé muy bien qué decir aquí. Pensaba en explicar el modo .org y cómo se van tangleando los scripts y eso, pero me parece que eso es emacs, ¿no? *emoji pensativo*. O puedo hablar de lo útil que resulta la herencia múltiple para definir las clases a partir de la combinación de varias clases abstractas y cómo se usa esto después en los métodos para ver qué especializaciones usar... Pero en esta parte me parece violación pues no he explicado nada de código *otro emoji pensativo*

Conclusiones

Recomendaciones

Bibliografía

[1] Daniela González Beltrán. Generación automática de gramáticas para la obtención de infinitos criterios de vecindad en el problema de enrutamiento de vehículos. 2019.

[2] PaoloVigo George B Dantzig and John H Ramser. The truck dispatching problem. *Management science*, 6(1):80–91, 1959.

[3] Heidy Abreu Fumero. Exploración de una vecindad para una solución de vrp combinando diferentes estrategias de exploración y de selección. 2019.

[4] Camila Pérez Mosquera. Primeras aproximaciones a la búsqueda de vecindad infinitamente variable. 2017.

[5] Héctor Felipe Massón Rosquete. Exploración de vecindades grandes en el problema de enrutamiento de vehículos usando técnicas estadísticas. 2020.

[6] José Jorge Rodríguez Salgado. 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. 2020.