UNIVERSIDAD DE BUENOS AIRES

Facultad de Ciencias Exactas y Naturales

Departamento de Computación

Una Heurística basada en Programación Lineal Entera para el Problema de Ruteo de Vehículos con Recolección y Entrega

Tesis presentada para optar al título de
Licenciado en Ciencias de la Computación

Agustin Ismael Montero

Directores de tesis: Dr. Juan José Miranda Bront y Dra. Isabel Méndez Díaz.

Lugar de Trabajo: Departamento de Computación, FCEyN, Universidad de Buenos Aires.

Una Heurística basada en Programación Lineal Entera para el Problema de Ruteo de Vehículos con Recolección y Entrega

En esta tesis abordamos el Problema de Ruteo de Vehículos con Recolección y Entrega (VRPPD, por sus siglas en inglés) con un enfoque heurístico. Se considera la variante *one-to-one* (ver, e.g. Berbeglia et al. [6]) donde se debe satisfacer una precedencia particular entre pares de clientes origen-destino. Formalmente, sea $G = (V, E)$ un digrafo completo, con $V = \{0, 1, \ldots, 2n\}$ el conjunto de nodos y E el conjunto de ejes, donde 0 representa el depósito. Cada eje $(i, j) \in E$ tiene un distancia $d_{ij} \geq 0$ asociada, y existe una restricción de precedencia entre los nodos $i, n + i$, para $i = 1, \ldots, n$. El objetivo es encontrar un conjunto de a lo sumo k rutas que cubran cada vértice exactamente una vez y que respete las precedencias, con un mínimo costo total.

En este trabajo, se considera el esquema propuesto por De Franceschi et al. [18] para el VRP con Capacidad y Restricciones de Distancia, que fue aplicado con éxito a otras variantes del VRP (ver, e.g., Toth y Tramontani [55], Naji-Azimi et al. [46]). Partiendo de una solución inicial factible, este esquema se basa en el paradigma de destrucción/reparación, donde un conjunto de nodos es removido de las rutas y reinsertado a través de la resolución heurística de una formulación ILP llamada Reallocation Model (RM) con una cantidad exponencial de variables. Dado que el RM tiene un número exponencial de variables, el enfoque standard es generar columnas heurísticamente considerando los costos reducidos y resolviendo el modelo con un ILP solver de propósito general, como por ejemplo CPLEX.

Se presenta una formulación del RM que incluye las restricciones de precedencia para asegurar la factibilidad de la solución. A diferencia de los casos anteriores, el número de filas en el RM no está fijo *a priori* dado que las restricciones de precedencia tienen que ser incluídas en la formulación dependiendo del conjunto de variables generadas, y la formulación se vuelve entonces un caso particular del problema de *filas dependientes de las columnas* (ver, e.g. en Muter et al. [44]). En consecuencia, sin resolver subproblemas auxiliares, los costos reducidos pueden solamente ser aproximados usando la información presente en el dual. Se estudia entonces el comportamiento computacional del esquema general y se proponen algunas modificaciones a fines de generar variables de buena calidad para el RM. Basados en resultados preliminares, el esquema propuesto muestra mucho potencial para ser aplicado en la práctica y es un buen punto de partida para considerar versiones más complejas del VRPPD.

Keywords: VRP con Recolección y Entrega, Programación Lineal Entera, Reallocation Model, Generación de Columnas

An ILP-based heuristic for the VRP with pickups and deliveries

We address the Vehicle Routing Problem with pickups and deliveries (VRPPD) with a heuristic approach. We consider the *one-to-one* problem (see, *e.g.* Berbeglia et al. [6]) where a particular precedence must be satisfied by a pair of origin-destination clients. Formally, let $G = (V, E)$ be an undirected complete graph, with $V = \{0, 1, \ldots, 2n\}$ the set of vertices and E the set of edges, where 0 represents the depot. Each edge $(i, j) \in E$ has an associated distance $d_{ij} \geq 0$ and there is a precedence constraint between vertices $i, n + i$, for $i = 1, \ldots, n$. The objective is to find a set of at most k routes covering each vertex exactly once at minimum total cost.

In this research, we consider the scheme proposed in De Franceschi et al. [18] for the Distance Constrained Capacitated VRP, which has been successfully applied to other variants of the VRP (see, e.g., Toth and Tramontani [55], Naji-Azimi et al. [46]). Starting from a initial feasible solution, this scheme follows the destroy/repair paradigm where a set of vertices is removed from the routes and reinserted by solving heuristically an associated ILP formulation with an exponential number of variables, named *Reallocation Model* (RM). Given the RM has an exponential number of variables, the standard approach is to heuristically generate columns considering their reduced cost and solve the model by a general purpose ILP solver, such as CPLEX.

Our work presents a formulation of the RM that includes the precedence constraints to guarantee the feasibility of the solution. In contrast to the previous cases, the number of rows in the RM is not fixed *a priori* since precedence constraints have to be included in the formulation depending on the set of variables generated, and the formulation becomes a special case of the column-dependent-rows problem studied, *e.g.*, in Muter et al. [44]. As a result, reduced costs can only be approximated using the current dual information without solving auxiliary subproblems. Therefore, we study the computational behavior of the general scheme and propose some modifications in order to generate good quality variables for the RM. Based on preliminary results, the proposed scheme shows good potential to be applied in practice and a is good starting point to consider more complex versions of the VRPPD.

Keywords: VRP with Pickups and Deliveries, Integer Linear Programming, Reallocation Model, Column Generation

Agradecimientos

La gente rara vez tiene éxito, a menos que se divierta con lo que hace
D. Carnegie

Como lo dije en su momento, más allá del final que este trabajo representa para mi, y que a su vez es un principio de muchas otras cosas, es una etapa que cierro con mucha alegría. Durante todo este camino hubo muchas personas, conocí muchísima gente, es larga la lista y son infinitas las gracias que tengo para darles.

¡GRACIAS TOTALES!

Quiero agradecer desde ya a mis directores, Juanjo e Isabel, por todo el tiempo, dedicación y soporte que me brindaron durante la realización de esta tesis. En particular, a Juanjo por los mates compartidos, los tantos debates y los acertados consejos.

Gracias también al jurado, Javier y Santiago, por la excelente disposición y velocidad con la que evaluaron el trabajo.

Quiero agradecer en particular a toda mi familia, que estuvo, está y seguirá estando siempre. Al invencible de mi viejo, que es un ejemplo. A mi vieja de fierro, que si me soportó a mi, se banca cualquier cosa. Y también a Flor y Fede, por tantos momentos de alegría desde que nacieron.

A Caro, por bancarme y acompañarme en esta vida que compartimos con alegría. A mis suegros, y mi cuñado, con quienes comparto tantos asados.

A los pibes, mis amigos, Andrelo, Cabe y Mario por tanta magia de amistad a lo largo de estos años.

A mis padrinos, Susana y Guillermo, a mis primos y mis tías, por siempre estar ahí, por las risas y chusmerios.

A los pibes de Fullvazo, por todos y cada uno de los findes futboleando.

A los amigos de la facu, en especial a Pape, Nico, Dani, Martu, Lean, Juli y el rayo.

Al goc, a Fede, Braian, Pau, Pablo, e Irene, por la buena onda que tienen laburando.

A los compañeros y docentes de la carrera, por el esfuerzo que ponen para día a día seguir enseñando y aprendiendo.

Y nuevamente, a todas aquellas personas que conocí durante esta etapa del camino, son muchas, es larga la lista, y son infinitas las gracias.

Índice general

1. Introducción

El análisis combinatorio es un área que abarca el estudio de problemas que involucran ordenar, agrupar o seleccionar un conjunto discreto de elementos, generalmente finito, desde un punto de vista matemático.

Tradicionalmente, el estudio de este tipo de problemas llamados *problemas combinatorios*, ha estado ligado a cuestiones de existencia o de enumeración: ¿existe esta configuración de elementos? ó ¿cuántos ordenamientos son posibles?

Por otro lado, muchas veces la existencia de una configuración determinada no es el punto en discusión, o la cantidad de ordenamientos no es importante, y el análisis pasa a estar ligado a otro tipo de pregunta: ¿cuál es la *mejor* configuración de elementos?

La optimización combinatoria es un área particular dentro de la optimización matemática, relacionada con la investigación operativa y teoría algorítmica principalmente, en donde se busca soluciones a problemas combinatorios o de estructura discreta con el objetivo de encontrar la *mejor* solución.

Es un área multidisciplinaria con muchas aplicaciones reales que van desde la Biología, Robótica, Física y Química, hasta las Finanzas, Marketing e Ingeniería. Diferentes metodologías fueron desarrolladas para los problemas combinatorios en distintos contextos, originando una gran variedad de técnicas.

Dentro de la Investigación Operativa (IO), las áreas más comunes de aplicación son la logística, management y la administración de recursos, siendo una de las metodologías más exitosas la Programación Lineal Entera (PLE).

Dos problemas muy conocidos dentro del área son el Problema del Viajante de Comercio (TSP) y el Problema de Ruteo de Vehículos (VRP). Ambos han sido muy estudiados tanto en la teoría como en la práctica debido a la gran cantidad de aplicaciones que tienen en el mundo real. En las siguientes secciones se introduce el Problema de Ruteo de Vehículos y sus variantes más comunes en la literatura.

1.1. Problemas de Ruteo de Vehículos

Los Problemas de Ruteo de Vehículos son una clase de problemas que estudian la distribución de bienes o servicios entre un depósito y los usuarios finales (clientes).

En el año 1959, Dantzig y Ramser [17] proponen en *The truck dispatching problem* "encontrar el óptimo ruteo de una flota de camiones de reparto de gasolina entre una planta terminal y un gran número de estaciones alimentadas por dicha terminal". Mediante una formulación del

problema basada en programación lineal (ver 2.2), calculan una solución de gran calidad, con 4 rutas, para el problema de las 12 estaciones de servicio, y agregan: "todavía no se han realizado aplicaciones prácticas del método".

En los siguientes 50 años [38], el trabajo de Dantzig y Ramser [17] se desarrolla enormemente con muchas aplicaciones prácticas reales. Aparecen una gran cantidad de estudios sobre VRPs, desde trabajos como *Distribution Management*, Christofides et al. [19] en la década del 70, hasta otros más recientes como *The Vehicle Routing Problem*, Toth y Vigo [56], y *The Vehicle Routing Problem: Latest Advances and New Challenges*, Golden et al. [27].

Así, el ruteo de vehículos se considera en la práctica uno de los problemas más importantes de la investigación operativa, en particular en la industria y en el sector de servicios, donde el costo del transporte representa una porción significativa del valor total de los bienes y servicios otorgados.

En la versión general del problema, un conjunto de vehículos realiza movimientos a través de una red de rutas partiendo de puntos fijos (depósitos). Dentro de la red, los tramos (arcos) pueden transitarse en una o ambas direcciones, y tienen asociados un costo o tiempo de viaje que puede depender de muchos factores, como por ejemplo el tipo de vehículo o el momento del día en que es transitado.

Algunas aplicaciones de este tipo de problema son la recolección y distribución de correspondencia, el recorrido de un médico que atiende enfermos a domicilio, el recorrido de ambulancias, entrega de pedidos de comida, recolección de residuos, etc. El ejemplo más conocido de esta familia es el Problema del Viajante de Comercio, en donde se dispone de un único vehículo para visitar un conjunto de ciudades (o clientes), pasando por cada uno de ellos exactamente una vez, retornando finalmente al origen. El objetivo de minimizar el tiempo total de viaje.

Las principales características de VRPs están dadas por las restricciones de operación (reglas de factibilidad) que se deben cumplir, asociadas por ejemplo a la capacidad de los vehículos, o respecto de la relación de precedencia entre las visitas a los clientes, etc. Por lo general, las características clásicas son:

- Cada cliente tiene asociada una demanda o cantidad de mercadería que debe recibir (o entregar), pudiendo existir además restricciones en la forma en que dicha mercadería debe ser entregada (e.g., un único vehículo debe visitar a los clientes, la entrega debe completarse en una sola visita, es posible completar la entrega en a lo sumo m visitas).
- Ventanas de tiempo para visitar clientes, dadas por restricciones en los horarios en que el cliente está disponible, o por limitaciones de tráfico. Es común que en las ciudades la entrega de mercadería por parte de los proveedores a los comercios se haga por la mañana, y muchas veces el horario está regulado por leyes (a fines de evitar congestionamiento de tránsito, etc.). En otros rubros, como por ejemplo la distribución del periódico, los fines de la actividad en sí mismos exigen que la visita a clientes se haga por la mañana.
- Cantidad de vehículos disponibles para visitar los clientes. En la práctica adquirir más vehículos puede estar limitado por el costo de compra, por la capacidad de albergar vehículos en el depósito, etc. En general, se suele diferenciar el caso mono-vehículo y el multi-vehículo.

- La capacidad de los vehículos utilizados para visitar clientes. Los vehículos pueden ser homogéneos (todos tienen la misma capacidad) o pueden ser heterogéneos. Por otro lado, los clientes podrían imponer restricciones sobre la capacidad de los vehículos que los visitan, exigiendo que no superen determinada capacidad (porque no tienen suficiente espacio para atenderlos), o porque las vías de acceso al cliente no permiten superar un determinado peso en el vehículo.
- La cantidad de depósitos. Es posible que exista más de un depósito, y aún más, los clientes podrían presentar restricciones sobre los depósitos que pueden abastecerlos.
- Costo de traslado, pudiendo depender simplemente de la distancia o también del tamaño del vehículo, horario de traslado, etc.
- Relaciones de precedencia entre los clientes. Puede haber restricciones en el orden en que los clientes deben ser visitados dentro de una misma ruta. Los servicios de mensajería por ejemplo, deben recolectar primero un mensaje de determinado cliente, y entregarlo a otro (u otros) clientes luego. Incluso, sistemas de transporte puerta-a-puerta requieren trasladar un conjunto de pasajeros donde cada uno tiene asociado un par origen-destino particular, como es el caso del transporte de pacientes en ambulancias.

Por otro lado, el problema también puede variar en la función objetivo que se desea optimizar. Algunos objetivos típicos de VRP son:

- Minimización del costo total de transporte, dependiendo por ejemplo de la distancia total recorrida (o del tiempo total consumido) y de los costos fijos por utilizar cada vehículo.
- Minimización de la cantidad de vehículos (o conductores) utilizados para visitar clientes.
- Minimización del tiempo que los clientes esperan hasta ser visitados.

Existen variantes más específicas, y muchas veces se suele optimizar una combinación pesada de objetivos que sirven a distintos propósitos, y que incluso pueden ser contradictorios.

Diferentes configuraciones de estos factores determinan distintas variantes de VRP. En el contexto de esta tesis, se introduce a continuación una familia de variantes de VRP conocida como VRP con Recolección y Entrega, y se presenta la versión particular del problema sobre la que se trabaja.

1.2. Problemas de Ruteo de Vehículos con Recolección y Entrega

Los problemas de *VRP con Recolección y Entrega* (VRPPD por sus siglas en inglés, ó PDP) abarcan problemas de ruteo donde un conjunto de productos deben ser transportados entre clientes de recolección y clientes de entrega.

En la versión clásica del VRPPD cada cliente i tiene asociada una cantidad de producto d_i que le debe ser entregada, y una cantidad de producto p_i que debe ser recolectada en el cliente. Se asume que los productos son homogéneos para todos los clientes, y muchas veces se suele utilizar una única cantidad de producto asociada a cada uno, considerando la diferencia neta entre la demanda de recolección y entrega como $\tilde{d}_i = d_i - p_i$.

Siguiendo la notación utilizada por Toth y Vigo [56], dado un cliente i, se tiene asociado otro cliente O_i desde donde debe suministrarse la cantidad d_i del producto, y un cliente D_i al que se debe enviar la cantidad p_i del producto. Se asume que en cada cliente siempre se realiza la entrega antes que la recolección, i.e., primero debe recibir la cantidad d_i que le corresponde, y luego recién es posible recolectar p_i.

El problema consiste entonces en encontrar exactamente k rutas simples (cada una asociada a un vehículo) con un costo mínimo total de forma tal que:

- Cada ruta empieza y termina en el depósito.
- Cada cliente es visitado por exactamente una ruta.
- Dentro de una ruta, la carga del vehículo no debe exceder una carga máxima C.
- Para cada cliente i, si el cliente O_i no es el depósito, entonces debe ser visitado en la misma ruta que i, antes que i.
- Para cada cliente i, si el cliente D_i no es el depósito, entonces debe ser visitado en la misma ruta que i, después que i.

Esta variante es una generalización del VRP clásico con Capacidades (CVRP) donde para cada cliente i se tiene que $O_i = D_i =$ depósito, y $p_i = 0$. Dado que CVRP generaliza TSP, y que este último es $\mathcal{NP} - Hard$ [41], estas variantes del VRP también lo son.

En la literatura es muy común que se consideren también restricciones de ventanas de tiempo, dando lugar al VRP con Recolección y Entrega y Ventanas de Tiempo (VRPPDTW, por sus siglas en inglés). Para una clasificación sobre PDPs en general, ver Berbeglia et al. [6].

En el marco de esta tesis, se trabaja sobre una variante particular del VRPPD sin ventanas de tiempo ni restricción de capacidad en los vehículos (capacidad infinita). Además, se particiona el conjunto de clientes en *Recolecciones* y *Entregas*, de forma que dado un cliente i, este es un cliente de Recolección ó un cliente de Entrega, pero no ambos simultáneamente. La relación de precedencia está dada uno-a-uno, donde para cada cliente de Recolección existe un único cliente de Entrega relacionado, y vice-versa.

En este nuevo escenario, el problema consiste en encontrar a lo sumo k rutas simples (cada una asociada a un vehículo), con un costo mínimo total de forma tal que:

- Cada ruta empieza y termina en el depósito.
- Cada cliente es visitado por exactamente una ruta.
- Dado un par de clientes relacionados P_k, D_k, donde P_k es Recolección y D_k es Entrega, P_k debe ser visitado antes que D_k, y ambos deben pertenecer a la misma ruta.

Esta versión, que se denota como VRPPD1-1, también es $\mathcal{NP} - Hard\ strong$ dado que es una generalización de VRP.

1.3. Objetivo de la tesis

La versión del VRP con Entrega y Recolección, como así también muchas otras variantes, son problemas $\mathcal{NP} - Hard$ ampliamente estudiados en la literatura en el marco de algoritmos exactos, donde el objetivo es encontrar la solución óptima del problema, y (meta) heurísticos, donde se pretende encontrar soluciones de buena calidad (sin garantías del óptimo) en el marco de instancias de gran escala, o en contexto donde el tiempo de resolución está acotado.

Para los algoritmos de tipo exacto, la formulación de este tipo de problemas como Problemas de Programación Lineal Entera (PLE) ha mostrado muy buenos resultados en los últimos años, y existe al día de hoy una gran variedad de software de propósito general con un alto nivel de madurez (e.g., CPLEX, Gurobi, SCIP) para resolver este tipo de formulaciones. Si bien actualmente es posible resolver instancias de problemas que hasta hace unos años parecían inalcanzables, este tipo de algoritmos sigue encontrando limitaciones al intentar resolver instancias reales, que suelen ser de gran escala.

En las últimas décadas los avances obtenidos en la resolución de Problemas de Programación Lineal Entera Mixta (PLEM) permitieron la aplicación de este tipo de técnicas para resolver problemas auxiliares de optimización en general, ya sea para la resolución de un PLE en sí mismo, o por ejemplo para explorar grandes vecindarios dentro de algoritmos de problemas específicos. Este tipo de técnica se denomina MIPing, y está incorporada en muchos paquetes de propósito general. Algunos ejemplos son la heurística *Feasibility Pump* [22, 7, 3], para obtener soluciones iniciales de un PLEM, *Local Branching* [23, 35], una estrategia de *branching* que apunta a explorar estratégicamente el árbol de enumeración con el objetivo de mejorar constantemente la cota primal, ó *Relaxation Induced Neighbourhood Search* (RINS) [16], una heurística primal de propósito general que utiliza información de la relajación lineal en cada nodo del árbol.

Con este enfoque, se conoce el desarrollo de metaheurísticas para problemas particulares con el nombre de Matheuristics (Mathematical Programming Heuristics). En [18, 55, 6, 46] se aplica esta idea dentro en un esquema meta-heurístico para el VRP y algunas de sus variantes, obteniendo soluciones de muy buena calidad para instancias de gran tamaño, sugiriendo que el enfoque tiene mucho potencial. Incluso, en algunas instancias, la mejor solución de la literatura es mejorada.

En particular, De Franceschi et al. [18] sugiere investigar la adaptación del esquema al caso donde se tienen restricciones de precedencias entre los clientes. El objetivo de la tesis va en esta dirección. Considerando una versión particular del VRPPD, se pretende estudiar el comportamiento de dicho esquema ante la presencia de precedencias entre clientes, analizando posibles adaptaciones, así como también nuevos desafíos y dificultades intrínsecas en este contexto.

2. Preliminares

Se recuerda que en los problemas de optimización se busca encontrar la mejor configuración para un conjunto de variables que deben cumplir con ciertas restricciones. En particular, cuando las variables involucradas en el problema son enteras, es posible enumerar todas las configuraciones para encontrar aquella que, satisfaciendo las restricciones, es óptima para la instancia del problema. Algoritmos como éste en donde se tiene la garantía de que dada una instancia del problema siempre se encuentra el óptimo, se llaman *exactos*.

Para muchos problemas de optimización que son $\mathcal{NP}$-Hard [25], no existe un algoritmo que resuelva una instancia arbitraria del problema en tiempo polinomial (asumiendo $\mathcal{P} \neq \mathcal{NP}$). Por lo tanto, los métodos exactos requieren un tiempo computacional exponencial en el peor caso, que en general suele ser muy alto a fines prácticos. En consecuencia, se decide utilizar métodos *aproximados* en donde se sacrifica la garantía de optimalidad en la solución, a cambio de encontrar soluciones de *buena calidad* en tiempos computacionales razonables.

El propósito de este capítulo es introducir los algoritmos principales, tanto *exactos* como *aproximados*, utilizados para resolver problemas de optimización combinatoria.

2.1. Programación Lineal

La *Programación Lineal Entera* permite modelar una clase de problemas de optimización con restricciones, en donde algunas variables son enteras.

En la práctica, se suele utilizar el nombre *Programación Lineal Entera Mixta* (PLEM) dado que muchas aplicaciones reales involucran tanto variables continuas como enteras, la función objetivo es lineal, y las restricciones son desigualdades lineales.

Con apenas poco más de 50 años [38], PLEM ha sido desarrollada en aspectos teóricos como prácticos, consiguiendo mucho éxito en el mundo académico y profesional. Actualmente una gran cantidad de problemas difíciles de optimización combinatoria pueden ser modelados usando formulaciones PLEM a través de técnicas standard [11].

A continuación se introducen los conceptos principales y algoritmos clásicos de Programación Lineal Entera Mixta, que son profundizados en detalle por Wolsey [48]. Se asume que el lector está familiarizado con las ideas básicas de Programación Lineal.

Un problema de *Programación Lineal* (PL) es un problema de optimización con restricciones en donde se quiere encontrar un conjunto de valores para variables continuas $(x_1, \ldots, x_n)$ que minimizan o maximizan una función objetivo lineal z, satisfaciendo también un conjunto de restricciones lineales.

Formalmente, un problema PL se define como:

$$\begin{aligned} (PL) \quad \text{mín} \quad & z = \sum_{j=1}^{n} c_j x_j \\ \text{sujeto a} \quad & \sum_{j=1}^{n} a_{ij} x_j \leq b_i \quad (i = 1, 2, \ldots, m) \\ & x_j \in \mathbb{R}_+ \qquad\qquad (j = 1, 2, \ldots, n) \end{aligned} \tag{2.1}$$

donde el vector $c \in \mathbb{R}^n$ representa la función objetivo y los coeficientes a_{ij} pueden representarse como una matriz $A \in \mathbb{R}^{m \times n}$.

Para más detalle sobre PL se recomienda al lector consultar Chvátal [14].

2.2. Programación Lineal Entera Mixta

Un problema de *Programación Lineal Entera* (PLE) es un problema PL donde al menos una variable debe satisfacer restricciones de integralidad (i.e. está restringida a valores enteros). En caso de que no todas las variables sean restringidas a valores enteros, se suele utilizar el término *Programación Lineal Entera Mixta* (PLEM).

Formalmente, un problema PLEM se define como:

$$\begin{aligned} (PLEM) \quad \text{mín} \quad & \sum_{j \in I} c_j x_j + \sum_{j \in C} c_j x_j \\ \text{sujeto a} \quad & \sum_{j \in I} a_{ij} x_j + \sum_{j \in C} a_{ij} x_j \leq b_i \quad (i = 1, 2, \ldots, m) \\ & x_j \in \mathbb{Z}_+ \qquad\qquad \forall\, j \in I \\ & x_j \in \mathbb{R}_+ \qquad\qquad \forall\, j \in C \end{aligned} \tag{2.2}$$

donde I es el conjunto de variables enteras y C el conjunto de variables continuas, con $I \cup C = \{1, \ldots, n\}$. Para simplificar la notación, se reescribe (2.2) como:

$$\text{mín}\{cx \ : \ Ax \leq b,\ x \geq 0,\ x_j \in \mathbb{Z}_+ \ \forall j \in I\} \tag{2.3}$$

Este problema en su versión general pertenece a la clase $\mathcal{NP} - Hard$.

Si se considera el poliedro $P = \{x \in \mathbb{R}^n_+ \ : \ Ax \leq b\}$, con $A \in \mathbb{R}^{m \times n}$ y $b \in \mathbb{R}^m$, el conjunto de las soluciones factibles se define como $S = P \cap \{x \in \mathbb{R}^n_+ \ : \ x_j \in \mathbb{Z}\ \forall\, j \in I\}$, y se llama relajación lineal (LP) del problema (2.2) a:

$$\text{mín}\{cx \ : \ Ax \leq b,\ x \geq 0\} \tag{2.4}$$

La diferencia principal entre el problema (2.1) y el problema (2.2) es que para el primero existe un algoritmo capaz de resolverlo en tiempo polinomial (ver Karmarkar [39]), y para el segundo no, a menos que $\mathcal{P} = \mathcal{NP}$.

Se denota con $conv(S)$ a la cápsula convexa de S (el poliedro más chico tal que $S \subseteq conv(S)$). Si en particular $P = conv(S)$ y el número de restricciones necesarias para describir $conv(S)$ es polinomial, entonces cualquier algoritmo de programación lineal puede utilizarse para resolver el PLEM hasta optimialidad.

Si no se conoce la caracterización de $conv(S)$ a través de restricciones lineales, resulta muy útil la relajación PL. Si esta última es infactible, también lo será el problema PLEM, si la solución óptima de la relajación PL satisface las restricciones de integralidad, también será una solución óptima del problema PLEM, y si en cambio la solución óptima de la relajación PL no satisface las restricciones de integralidad, entonces asumiendo que el problema en cuestión es de minimización (maximización), el valor objetivo óptimo de dicha relajación es una cota inferior (superior) para el valor objetivo óptimo del problema PLEM.

Se tiene entonces que la diferencia entre el valor objetivo z de una solución factible PLEM y el valor objetivo óptimo de la relajación PL es una cota superior para la diferencia relativa entre z y la solución óptima del PLEM, denominada *gap*. Esto es muy útil para evaluar la calidad de una solución o, en algunos casos, para asegurar optimalidad.

Es importante mencionar que existen actualmente paquetes de propósito general que implementan el estado del arte [36, 30, 20, 62] incorporando técnicas muy avanzadas [1, 2, 9, 8, 37, 42, 29], y aún así, algunas formulaciones PLEM no pueden ser efectivamente resueltas, siendo necesario aplicar técnicas no standard de formulación de modelos PLEM (ver Vielma [58]).

Finalmente, cabe mencionar que muchos algoritmos para resolver PLEMs se basan en la relación entre el problema y su relajación PL. Los algoritmos más usados en la práctica son presentandos en la siguiente sección.

2.3. Algoritmos para PLEMs

Los métodos exactos para resolver PLEMs involucran paradigmas muy conocidos en optimización combinatoria: branch and bound (B&B), planos de corte, branch and cut (B&C), y branch and price (B&P). A continuación se introducen brevemente algunos enfoques para resolver este tipo de problemas.

Es importante resaltar que su efectividad depende de la implementación, y si bien cada uno puede dar buenos resultados aplicado de manera independiente, son capaces por sí solos de resolver instancias de tamaño limitado, por lo que muchas veces se aplican algoritmos que utilizan combinaciones de estos diferentes enfoques.

2.3.1. Branch-and-Bound (B&B)

Desarrollado en 1960 [40], B&B es en realidad un enfoque de propósito general capaz de resolver problemas muy genéricos.

Está basado en la idea de explorar todo el espacio de soluciones de manera implícita, dejando de lado grandes partes de dicho espacio a fines de agilizar la búsqueda.

Para realizar esto, se aprovecha la técnica *divide and conquer*, haciendo un *branching* (divide) y un *bounding* (conquer) durante el proceso de exploración, utilizando información del problema para decidir cómo recortar el espacio de búsqueda sin dejar afuera la solución óptima del problema.

Una forma simple de representar dicho proceso es a través de un árbol de B&B, en donde el nodo raíz representa el problema original, y las ramas representan las particiones resultantes del *branching*. En cada nodo se utiliza información del subproblema en cuestión, junto con argumentos de factibilidad y dominancia, para decicir si es posible hacer una poda de alguna rama del árbol, o en caso contrario se debe seguir aplicando un *branching* a fines de separar el subproblema es otros más pequeños, esperando se facilite aún más la búsqueda.

Para decidir si es posible podar, es común utilizar el *bounding*, en donde dentro de un nodo del árbol se calcula una cota inferior (asumiendo que el problema es una minimización) para la solución óptima restringida al espacio de búsqueda en cuestión. Siempre que dicha cota sea mayor a alguna cota superior conocida del problema, no será necesario continuar explorando la rama actual del árbol.

La técnica utilizada para computar cotas debe contemplar un equilibrio entre calidad de la cota calculada, y esfuerzo computacional requerido. Cotas muy fáciles de computar podrían generar mucho *branching* durante el proceso de búsqueda, explorando innecesariamente muchas ramas del árbol. Por otro lado, cotas muy ajustadas podrían necesitar mucho tiempo de cómputo, haciendo poco eficiente el algoritmo en general.

Un enfoque posible es utilizar una relajación del problema. En este contexto, para calcular una cota inferior del problema es válido aprovechar el valor de la respectiva relajación PL. Por otro lado, el valor de la mejor solución obtenida hasta el momento, para el problema original, se puede utilizar como cota superior.

Dicho esto, si se parte del problema original, se resuelve la relajación PL, y si dicha relajación es entera, el problema está resuelto. En caso contrario, la relajación se transforma en la cota inferior inicial, y el nodo raíz se particiona en subproblemas. Cada una de los *subproblemas* resultantes de dicha partición deben cumplir lo siguiente:

- Excluyen la solución no entera hallada en la relajación.
- Los subproblemas originados en la partición, representados por los nodos sucesores, son mutuamente excluyentes y la unión de estos determina la misma región factible de puntos enteros que su predecesor.

Resolviendo la relajación en un nodo es posible determinar cuándo es necesario un *branching* más profundo, o cuando es posible hacer una poda, e incluso permite determinar mejores cotas inferiores sobre el óptimo del problema original.

En particular, hay 3 casos que indican que la rama asociada a un nodo puede podarse:

- El subproblema asociado es *no factible*, (*poda por infactibilidad*).
- El subproblema asociado tiene un óptimo entero, (*poda por optimalidad*).
- La cota inferior del óptimo del subproblema asociado es mayor o igual a la cota superior del problema original, (*poda por cota*).

En el caso en que se pode por optimalidad, el óptimo encontrado puede ser usado para mejorar la cota superior sobre el óptimo del problema original.

Por otro lado, es necesario determinar cómo particionar el problema. En general, entre las variables enteras fraccionarias de la solución de la relajación, se suele elegir una variable de *branching* por alguno de los siguientes criterios:

- Variable con valor fraccional más cercano a 0,5
- Variable con *mayor impacto* en la función objetivo
- Variable con el menor índice

El método más simple de *branching* consiste en seleccionar una variable x_i restringida a tomar valores enteros, con valor fraccional d en la relajación PL actual, y dividir el problema asociado al nodo en cuestión, en dos subproblemas representados como nodos hijos, agregando la restricción $v \leq \lfloor d \rfloor$ en uno de ellos, y $v \geq \lceil d \rceil$ en el otro. De esta manera, la aplicación de este proceso determina la generación de nodos en el árbol de B&B.

También en necesario decidir qué nodo explorar primero. Los criterios más comunes de búsqueda son *depth first* y *best bound first*. El primero pretende encontrar rápidamente una cota inferior al óptimo del problema original, para así poder podar por optimalidad. El segundo aplica el *branching* sobre el nodo activo con el menor valor de solución óptima asociada, con el fin de minimizar el número de nodos en el árbol de B&B.

Nuevamente, la efectividad de los criterios está determinada empíricamente, y suele utilizarse una combinación de ellos, por ejemplo, aplicar primero *depth first* para encontrar rápido una solución entera factible, y luego aplicar algo más híbrido usando también *best bound first*.

2.3.2. Cutting-Planes

El término *plano de corte* hace referencia a una restricción por desigualdad que permite *cortar* la parte fraccionaria de una región factible de un LP, sin excluir soluciones factibles enteras.

El enfoque general para utilizar planos de corte es comenzar ignorando las restricciones de integralidad de las variables y resolver la relajación PL. Si al menos una variable entera tiene valor fraccional, se trata de identificar una restricción lineal para $conv(S)$ que separe dicha solución

LP del conjunto de soluciones enteras factibles S. Agregar esta desigualdad a la formulación resulta en una relajación PL más ajustada, y es posible repetir el proceso.

Se dice que una desigualdad es válida si todo punto factible del problema debe satisfacer la desigualdad. Por otro lado, se dice que una desigualdad es un *corte* si la solución de la relajación PL viola la desigualdad.

Dado x^* fraccionario solución de la relajación PL, el *problema de separación* consiste precisamente en encontrar cortes violados por x^*. La efectividad de este enfoque depende de la posibilidad de resolver el problema de separación asociado, y de agregar los cortes a la formulación. Grötschel et al. [28] aportó uno de los resultados teóricos más importantes que relacionan la complejidad del problema de separación con los problemas de optimización, enunciando que el problema de optimización mín$\{cx : x \in conv(S)\}$ puede resolverse en tiempo polinomial si y sólo si el problema de separación también puede resolverse en tiempo polinomial. De esta manera, se puso en evidencia la dificultad de desarrollar procedimientos de separación, como así también la de encontrar una descripción completa para $conv(S)$.

Existen *cortes* que pueden generarse independientemente de la estructura del problema. Suelen ser *muy débiles*, pero pueden ser aplicados sin limitación alguna. Algunos de los denominados cortes de propósito general son los cortes disjuntivos, cortes cover, cortes clique, etc (ver [48]).

No obstante, también es posible explotar determinados tipos de restricciones dentro de problemas generales aprovechando la estructura particular del problema, e.g., desigualdades válidas para el problema knapsack [43].

2.3.3. Branch-and-Cut (B&C)

Es posible considerar B&C como una generalización del método de B&B, en el cual la idea es utilizar el algoritmo de B&B agregando *planos de corte* en los nodos del árbol de *branching* asociado, antes de realizar una poda o un *branching*.

Desde que los algoritmos de B&B y *planos de corte* (entre otros) fueron desarrollados alrededor de 1960, la resolución de problemas *PLEM* de gran escala estuvo limitada por casi dos décadas. Es así que a mediados de 1980 se propuso un enfoque que integraba los anteriores y que permitiría (con sus variantes) pasar de resolver sistemas con cientos de variables enteras, a resolver sistemas con miles de variables enteras al día de hoy.

Mientras que en B&B solo se aplican 2 cortes por cota en cada nodo y se trata de reoptimizar rápidamente la relajación en cada uno, B&C pretende poner el esfuerzo en obtener *cotas más ajustadas* en el nodo, ya sea buscando cortes fuertes, haciendo preprocesamiento o aplicando heurísticas.

El enfoque no especifica ciertos detalles, como cuántos cortes deben generarse, cuántas iteraciones de generación de planos de corte de deben aplicar, o cómo manejar los cortes generados, entre otros, sumados a los detalles no especificados heredados del B&B. Estas decisiones son un punto clave en la efectividad del algoritmo, y determinan el comportamiento y costo computacional requerido para aplicar el algoritmo en la práctica. Se debe encontrar un equilibrio entre las

diferentes estrategias adoptadas, y en general las decisiones dependen fuertemente del problema en cuestión.

Existe una variante que propone realizar la mayor parte del esfuerzo en la búsqueda de cortes *sólo* en el nodo raíz del árbol de B&B, con el objetivo de mejorar la relajación inicial, y reducir así considerablemente el tamaño del árbol de búsqueda. Dicha variante se conoce como *cut and branch*.

2.3.4. Branch and Price (B&P)

Esta técnica fue introducida por Savelsberg [54], B&P es una generalización del esquema B&B designada espeficamente para PLEMs de gran escala, muy frecuentes en la práctica, donde las formulación tiene un gran número de variables, en general exponencial, y que entonces no pueden ser manejadas de manera explícita.

Dado que el tiempo computacional para resolver la relajación PL del modelo puede ser prohibitivo, este enfoque propone utilizar la relajación PL restringida a un subconjunto de variables que pueda ser manejado en un tiempo computacional factible en la práctica. Dado que la solución obtenida es factible para la relajación PL original, pero no necesariamente óptima, se busca entre las variables no consideradas aquella que deba entrar a la base a fines de mejorar el funcional de la solución.

Dado y^* la solución óptima en el dual del problema restringido, se trata de determinar si existe una variable x tal que $rc_x = c_x - y^*x < 0$, donde y^* es el costo de x en la función objetivo dual, y rc_x denota el costo reducido de x. Si existe tal x, ésta es agregada al conjunto reducido de variables consideradas, y se re-optimiza el problema, mientras que si tal variable no existe, la solución óptima restringida es óptima en la relajación PL que considera todas las variables, por lo que se continúa con el esquema tradicional de B&B. Al proceso de generar este tipo de variables, i.e., columnas en la formulación restringida, se lo conoce como Problema de Generación de Columnas (CG, por sus siglas en inglés).

La resolución de subproblemas de CG y Pricing puede combinarse dentro de algoritmos más generales, intercambiando la resolución de un problema de CG cuando el costo computacional de este sea alto, por un subproblema de Pricing, donde las variables a evaluar son generadas heurísticamente, y en pos de reducir el esfuerzo computacional se sacrifica calidad en las variables que se evalúa agregar. Observar que en este último caso, a diferencia de cuando se resuelve un problema de CG, el hecho de que ninguna de las variables generadas resulte favorable para ser agregada a la formulación, no implica que la solución sea óptima.

En el contexto un algoritmo B&P, es un factor clave la determinación de un algoritmo eficiente para generar nuevas columnas, que permita la resolución de la relajación PL en el marco de tiempos computacionales razonables.

2.4. Búsqueda local

Los algoritmos de búsqueda local son un método heurístico que se basan en lo que probablemente sea el método de optimización más antiguo [49]: *prueba* y *error*.

Dentro de los métodos heurísticos, una opción es construir una solución desde cero, o bien empezar desde una solución conocida (no necesariamente de buena calidad), e iterativamente tratar de mejorarla analizando *soluciones parecidas*. Para este último caso, se suele usar el término *algoritmos de búsqueda local*, que son aquellos que se basan en la idea de partir de una solución inicial, definir un vecindario (i.e., soluciones *parecidas* a la inicial, con algún criterio), y explorarlo en busca de mejores soluciones.

Para definir un algoritmo de búsqueda local es necesario establecer:

1. Una forma de construir una solución inicial.
2. Movimientos (también llamados operadores) para determinar el vecindario a explorar.
3. Un criterio de aceptación y un criterio de parada.

Los movimientos son los encargados de generar soluciones dentro del vecindario, aplicando modificaciones a los atributos de la solución actual (e.g., orden entre los clientes en el contexto de una solución inicial para el VRP). Cada vez que se construye una solución en el vecindario, se la compara contra la solución actual para determinar si es mejor. El criterio de aceptación se utiliza para decidir cuándo una mejor solución es aceptada. Dos criterios muy utilizados en el contexto del VRP son:

- *First-accept:* Propone actualizar inmediatamente la solución actual con la primera solución que mejora.
- *Best-accept:* Antes de actualizar la solución actual, se examinan todas las soluciones en el vecindario determinado por los movimientos, en busca de la mejor de ellas.

Como criterio de parada se suele implementar *seguir buscando mientras la solución mejore*, combinando también cotas de tiempo o cantidad de iteraciones. Si ninguna de las soluciones del vecindario definido por los movimientos es mejor que la solución actual, se dice que se alcanzó un *óptimo local* en dicho vecindario, mientras que se llama, en cambio, *óptimo global* a la mejor solución posible de la instancia del problema.

3. Formulaciones para el VRPPD

A continuación se presenta formalmente la versión de VRPPD uno-a-uno introducida previamente en (1.2), y luego se muestran dos posibles formulaciones de programación lineal entera para el problema.

3.1. Definición del problema

Dado $G = (V, A)$ un digrafo completo, con $V = \{0, 1, \ldots, 2n\}$ el conjunto de nodos y A el conjunto de ejes, donde 0 representa el depósito. Cada eje $(i, j) \in A$ tiene una distancia $d_{ij} \geq 0$ asociada, y existe una restricción de precedencia entre los nodos $i, n + i$, para $i = 1, \ldots, n$.

El objetivo es encontrar un conjunto de a lo sumo k rutas que cubran cada vértice exactamente una vez, con un mínimo costo total.

3.2. Modelos PLEM de la literatura

Muchos problemas pueden ser formulados con modelos PLEM de dos maneras distintas:

- *Formulación Compacta*, como la formulación 3-*index* propuesta por Fisher y Jaikumar [24] para el VRP.
- *Formulación por Extensión*, como las formulaciones del tipo *set-partitioning*, e.g., Balinski y Quandt [5].

Es posible además transformar las formulaciones compactas en formulaciones por extensión a través de técnicas de descomposición. En general, las formulaciones por extensión tienen menos restricciones (filas), pero muchas más variables (columnas) que las formulaciones compactas. Es común que en el marco de ruteo de vehículos estas columnas se refieran a rutas.

Para resolver la relajación LP de formulaciones por extensión, se utilizan técnicas de Pricing y Generación de Columnas.

A continuación se presentan dos posibles formulaciones para el VRPPD1-1, una compacta y otra por extensión.

3.2.1. Formulación Compacta

Antes de presentar el modelo, se introduce la notación necesaria:

- $x_{ij}^k = \begin{cases} 1 & \text{el arco } (i,j) \text{ es atravesado por el vehículo k} \\ 0 & c.c. \end{cases}$.
- $P = \{1, \ldots, n\}$ es el conjunto de nodos de recolección.
- $D = \{n+1, \ldots, 2n\}$ es el conjunto de nodos de entrega.
- Si i es un vértice de recolección, entonces $n+i$ es el vértice de entrega asociado.
- 0 y $2n+1$ denotan el depósito.
- K denota el conjunto de vehículos ($|K|$ es la cantidad máxima de vehículos).
- c_{ij}^k es el costo por utilizar el arco (i,j) con el vehículo k.

Dado que en el contexto del VRPPD1-1 se cuenta con nodos (clientes) apareados en una relación de precedencia uno-a-uno, se introducen también las variables de tiempo B_i^k como una forma de modelar esta situación, utilizando la siguiente notación:

- t_{ij}^k es el tiempo de viaje desde el vértice i hacia el vértice j con el vehículo k.
- d_i es la duración del servicio en el vértice i.
- B_i^k es el inicio del servicio del vehículo k en el vértice i.

Observar que esta notación es válida tanto en el caso simétrico como asimétrico. En el primero, se tiene que $t_{ij}^k = t_{ji}^k$, $c_{ij}^k = c_{ji}^k$, el arco (i,j) y el arco (j,i) podrían modelarse entonces como un único arco.

De esta manera, el modelo básico resulta de adaptar la formulación cúbica para el VRP propuesta por Cordeau et al. [15] para el VRPTW.

$$\text{mín} \sum_{k \in K} \sum_{(i,j) \in A} c_{ij}^k x_{ij}^k \tag{3.1}$$

$$\text{subject to} \sum_{k \in K} \sum_{j:(i,j) \in A} x_{ij}^k = 1 \qquad i \in P \cup D \tag{3.2}$$

$$\sum_{j:(0,j) \in A} x_{0j}^k = 1 \qquad k \in K \tag{3.3}$$

$$\sum_{i:(i,2n+1) \in A} x_{i,2n+1}^k = 1 \qquad k \in K \tag{3.4}$$

$$\sum_{i:(i,j) \in A} x_{ij}^k - \sum_{i:(j,i) \in A} x_{ji}^k = 0 \qquad j \in P \cup D,\ k \in K \tag{3.5}$$

$$x_{ij}^k = 1 \Rightarrow B_j^k \geq (B_i^k + d_i + t_{ij}^k) \qquad (i,j) \in A,\ k \in K \tag{3.6}$$

$$\sum_{j:(i,j) \in A} x_{ij}^k - \sum_{j:(n+i,j) \in A} x_{n+i,j}^k = 0 \qquad i \in P\ k \in K \tag{3.7}$$

$$B_i^k \leq B_{n+i}^k \qquad i \in P\ k \in K \tag{3.8}$$

$$x_{ij}^k \in \{0,1\} \tag{3.9}$$

La función objetivo (3.1) minimiza el costo total de ruteo. Las restricción (3.2) exige que cada vértice sea visitado exactamente una vez. Las igualdades (3.3) y (3.4) garantizan que cada vehículo empieza y regresa al depósito al final de su ruta, contemplando el caso en que un vehículo no sea utilizado. La conservación del flujo se mantiene a través de la restricción (3.5), mientras que la eliminación de subtours se hace a través de variables de tiempo en (3.6), dado que $(t_{ij}^k + d_i) > 0$ para todo $(i,j) \in A$. Es posible linealizar (3.6) utilizando una formulación *big M*. La restricción (3.7) exige que origen y destino de un mismo pedido (dado por la relación de precedencia entre un vértice de entrega y el de recolección asociado), sean visitados por el mismo vehículo, mientras que (3.8) garantiza que la entrega solo puede ocurrir después de la recolección.

3.2.2. Formulación por Generación de Columnas

El VRPPD1-1 puede modelarse con una formulación de tipo *set-partitioning*. Para esto, se denota con Ω al conjunto de todas las rutas factibles (i.e., que respetan precedencias y cada vértice es utilizado a lo sumo una vez), a_{ir} es el número de veces que un vértice $i \in P$ es visitado por una ruta r, y c_r es el costo de la ruta r.

Se utilizan además las variables:

$$y_r = \begin{cases} 1 & \text{si } r \text{ es usada en la solución} \\ 0 & c.c. \end{cases} . \tag{3.10}$$

luego, el modelo resulta:

$$\text{mín} \sum_{r \in \Omega} c_r y_r \tag{3.11}$$

$$\text{subject to} \sum_{r \in \Omega} a_{ir} y_r = 1 \qquad i \in P \tag{3.12}$$

$$\sum_{r \in \Omega} y_r \leq k \tag{3.13}$$

$$y_r \in \{0, 1\} \qquad r \in \Omega \tag{3.14}$$

4. Modelo de Reubicación

4.1. Introducción

La versión clásica del VRP, así como muchas de sus variantes conocidas han sido ampliamente estudiadas en la literatura, tanto con enfoques exactos como (meta) heurísticos. En las últimas décadas, los avances obtenidos en la resolución de PLEMs generales permitió el uso de este tipo de técnicas para resolver sub-problemas de optimización ya sea dentro de la misma resolución de un PLEM, o por ejemplo para explorar grandes vecindarios en algoritmos de búsqueda local.

Siguiendo esta idea, una posibilidad propuesta en la literatura es aplicar un esquema de refinamiento heurístico basado en la resolución de un *modelo de reubicación* (RM, por sus siglas en inglés). Este esquema es un caso particular de búsqueda local basada en el paradigma de destrucción y reparación, en donde a partir de una solución inicial factible z_0 se aplican heurísticas de mejora. El Algoritmo 4.1 muestra el esquema general de refinamiento.

Algoritmo 4.1 Esquema general de refinamiento

Entrada: z_0 solución factible inicial.

1. *Extraer* un conjunto de nodos de las rutas de z_0.
2. *Construir secuencias* con los nodos extraídos y *re-insertarlas* buscando mejorar la calidad de la solución actual.

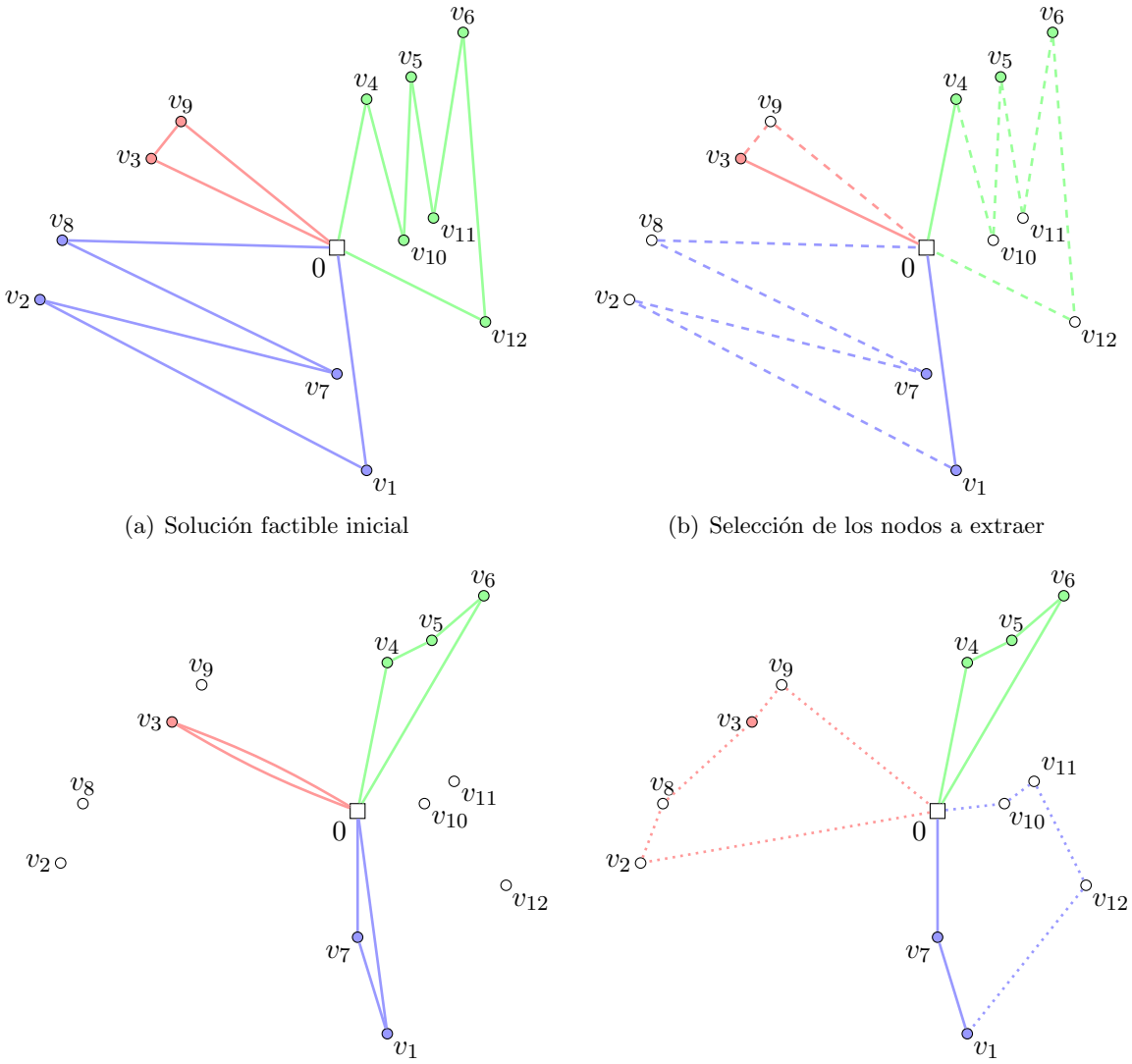
El esquema no especifica cuestiones como: ¿Qué nodos deben extraerse?, ¿Qué secuencias se deben utilizar en la re-inserción? ó ¿Cómo decidir dónde re-ubicar las secuencias?.

En este capítulo se consideran diferentes criterios para remover nodos, y la resolución (heurística) de un modelo PLE como modelo de re-ubicación para decidir cómo insertar las secuencias generadas.

En el siguiente ejemplo se observan las fases involucradas en la heurística de mejora dentro del esquema propuesto. En la Figura 4.1(a) se muestra una solución factible de 3 rutas para el VRP en una instancia euclídea con 12 clientes: $\{v_1, \ldots, v_{12}\}$, mientras que 0 denota el depósito.

En la etapa de extracción, los clientes son seleccionados con algún criterio de remoción. En la Figura 4.1(b) se puede ver cómo se seleccionan los nodos en posiciones pares (representandos con círculos blancos), mientras que el resto de los nodos quedan fijos. En este caso el conjunto de nodos a extraer es $\mathcal{F} = \{v_2, v_8, v_9, v_{10}, v_{11}, v_{12}\}$.

La Figura 4.1(c) muestra la *solución restringida* $z_0(\mathcal{F})$ resultante de "recortar" los ejes asociados a los nodos extraídos. En esta solución restringida, cada eje es un potencial *punto*



(a) Solución factible inicial

(b) Selección de los nodos a extraer

(c) Solución restringida (infactible)

(d) Solución de re-ubicación

Figura 4.1.: Un ejemplo de extraccción y re-ubicación de nodos para el VRP.

de inserción para los nodos (o *secuencias* de nodos) que fueron removidos. Se denota con $\mathcal{I}$ al conjunto de puntos de inserción, que en particular en la Figura 4.1(c) queda determinado como $\mathcal{I} = \mathcal{I}(z_0, \mathcal{F}) = \{(v_3, 0), (0, v_3), (0, v_4), (v_4, v_5), (v_5, v_6), (v_6, 0), (0, v_1), (v_1, v_7), (v_7, 0)\}$.

El vecindario de todas las soluciones que pueden obtenerse mediante la re-ubicación de los nodos en $\mathcal{F}$ se denota como $N(z_0, \mathcal{F})$, y la idea será explorarlo en busca de mejoras para z_0. Por ejemplo, la Figura 4.1(d) muestra la solución resultante de la re-ubicación, en donde v_9 fue re-ubicado en su posición original dentro de la misma ruta, mientras que el resto de los nodos fueron asignados a rutas distintas de las que fueron extraídos: v_2 y v_8 fueron re-ubicados a través de la secuencia (v_2, v_8) en el punto de inserción $(0, v_3)$; v_{10}, v_{11} y v_{12} fueron re-ubicados mediante

la secuencia (v_{10}, v_{11}, v_{12}) en el punto de inserción $(0, v_1)$.

Se utizará un nombre especial para distinguir los puntos de inserción que permiten recuperar la solución original z_0. Por ejemplo, en la Figura 4.1(c) se dice que $i = (v_3, 0) \in \mathcal{I}$ es el *punto de inserción pivote* para v_9, puesto que para recuperar la solución z_0 de la Figura 4.1(a) es necesario re-ubicar v_9 en i. De la misma manera, $i_8 = (v_1, 0)$ es el punto de inserción pivote para v_8, $i_2 = (v_1, v_7)$ es el punto de inserción pivote para v_2, etc.

La noción de *punto de inserción pivote* se generaliza para secuencias de nodos, donde $i_s = (a, b)$ es el punto de inserción pivote para una secuencia s de nodos en $\mathcal{F}$, si para recuperar la solución z_0 es necesario re-ubicar s en i_s. Además, se dice en este caso que s es una *secuencia básica*.

Esta idea fue aplicada con éxito en esquemas meta-heurísticos para el VRP y algunas de sus variantes, obteniendo soluciones de buena calidad en instancias de gran tamaño, poniendo en evidencia el potencial de la técnica, e incluso mejorando para algunas instancias la mejor solución conocida. En la siguiente sección se hace una revisión de la literatura relacionada.

4.2. Revisión de la literatura

En 1981, Sarvanov y Doroshko [53] proponen un método de refinamiento para el Problema del Viajante de Comercio (TSP, [32]), que sería propuesto también de manera independiente por Gutin [31]. Dado un tour inicial T para el TSP, se aplica un procedimiento de mejora basado en: (1) todos los nodos en posiciones pares (o impares) son removidos dejando "huecos" vacíos; (2) los nodos removidos se re-ubican en los "huecos" vacíos mediante un problema de asignación de suma mínima (ver [10]), que se puede resolver en tiempo polinomial.

Más allá de las propiedades teóricas, esta metodología sugiere un esquema sobre el cual se pueden construir heurísticas basadas en extracción-reubicación para el TSP (o problemas relacionados) cada vez más complejas. En esta dirección, Punnen [50] propone una variante del método en donde se consideran secuencias (y no solo nodos) para extraer y re-ubicar. Sin embargo, resultados preliminares (no publicados) reportados en [33] sugieren que el método no es bueno en la práctica, probablemente debido al rígido esquema de remoción y reubicación utilizado.

De Franceschi et al. [18] proponen ideas similares para el VRP conjeturando que la presencia de muchas rutas con restricciones es una buena estructura para estos esquemas de remoción-reubicación. Reemplazan el problema de asignación (resoluble en tiempo polinomial) por un modelo teóricamente $\mathcal{NP}$-Hard pero que puede ser resuelto efectivamente en la práctica (para instancias de tamaño limitado), y que ofrece más flexibilidad para la re-ubicación. A este modelo lo llaman *Reallocation Model* (RM), y lo abordan con una formulación PLEM. Dicha formulación tiene un número de variables exponencial en la cantidad de nodos extraídos, por lo que la solución de la relajación PL es manejada de manera heurística, dentro de un *ciclo de pricing* donde un gran número de variables son generadas, y agregadas al modelo solo si su costo reducido está por debajo de un umbral previamente fijado.

El RM es un punto crucial dentro del esquema de refinamiento que proponen De Franceschi et al. [18] en el contexto del VRP con restricciones de Capacidad y Distancia (DCVRP, ver 1.1). Se basa en una política de remoción de nodos más sofisticada, conjugándola con un algoritmo (SERR) de construcción de un gran número de secuencias con los nodos extraídos, utilizando el RM para re-ubicar dichas secuencias teniendo en cuenta las restricciones de distancia y capacidad, asegurando que cada nodo removido pertenezca a exactamente una secuencia re-ubicada. Para la resolución del modelo utilizan principalmente un paquete de propósito general.

En [18] se estudia este esquema sobre instancias de la literatura para el CVRP (con costos simétricos/asimétricos, con/sin restricciones de distancia) en dos clases de experimentos: (1) sobre soluciones iniciales (no necesariamente factibles) muy lejos del óptimo; (2) sobre soluciones factibles de mucha calidad (generalmente sobre la mejor solución reportada en la literatura). Los experimentos sugieren que SERR es muy efectivo mejorando la solución inicial incluso cuando ésta es de calidad, si bien en general el costo computacional es notable, y proponen para evitar esto, utilizar una version más rápida de SERR sacrificando la calidad de las mejoras.

Más tarde, Toth y Tramontani [55] estudian este esquema nuevamente para el DCVRP. Dado que el RM tiene un número de variables exponencial en la cantidad de nodos extraídos, investigan la estructura del vecindario definido por el modelo PLE, y el Problema de Generación de Columnas (CG) asociado a la relajación LP de la formulación utilizada para explorar dicho vecindario. Muestran que el problema CG es $\mathcal{NP}$-Hard, y proponen un enfoque de dos etapas: (1) se reduce (heurísticamente) el vecindario; (2) se explora el vecindario mediante la resolución del problema PLE con una solución (heurística) del problema CG asociado a la relajación LP.

Experimentos sobre 50 instancias de CVRP de la literatura (con/sin restricciones de distancia) sugieren que el método tiene potencial para mejorar soluciones del CVRP conocidas, incluso cuando estas son de buena calidad.

Este enfoque también fue estudiado en otros contextos. Salari et al. [52] investigan esta metodología en el marco del VRP abierto (OVRP, 1.1), en donde los vehículos no están obligados a regresar al depósito. En este caso, la solución es destruida aleatoriamente (i.e., se remueven nodos de manera aleatoria) y reparada resolviendo un modelo PLE de re-ubicación. Nuevamente, los resultados computacionales sobre 24 instancias benchmark sugieren que el esquema tiene mucho potencial mejorando soluciones de calidad obtenidas a partir de las técnicas meta-heurísticas más efectivas para el OVRP, incluso (en 10 instancias) frente a las mejores soluciones conocidas en la literatura.

Por último, Naji-Azimi, Salari y Toth [46] usan ideas similares para soluciones del m-Ring-Star Problem con restricciones de Capacidad (CmRSP). Proponen un algorimo VNS [34] basado en el modelo de re-ubicación, y consiguen mejoras para soluciones que los métodos de búsqueda local clásicos no pueden mejorar. Resultados sobre 138 instancias benchmark sugieren una gran superioridad del método frente a las heurísticas y métodos exactos para el CmRSP en la literatura.

En las siguientes secciones, se introducen las motivaciones e ideas originales de De Franceschi et al. [18] y Toth-Tramontani [55] sobre las que se basa este trabajo, y se detallan adaptaciones, modificaciones y aportes necesarios para aplicar esta metodología en el contexto del VRP con Pickups y Deliveries uno-a-uno.

4.3. Algoritmos de búsqueda local

Para implementar el esquema de destrucción y reparación presentado en las secciones anteriores, una opción es utilizar algoritmos de búsqueda local basados en la resolución de un modelo de re-ubicación. Dicho modelo es central en estos algoritmos, que construyen alrededor de este un marco de exploración del vecindario que el modelo determina, y que luego es posible ajustar para explotar atributos particulares del problema.

Se recuerda que el esquema de remoción y re-ubicación de nodos propuesto se basa en la idea de partir de una solución inicial z_0, seleccionar nodos con algún criterio, remover dichos nodos, y generar secuencias con estos para luego decidir cómo re-ubicarlas mediante la resolución de un modelo PLE (modelo de re-ubicación), donde se toma la desición de recuperar, en el peor de los casos, la solución original.

A continuación se detalla el esquema básico de dos algoritmos: el algoritmo SERR (siglas en inglés de *Selection, Extraction, Recombination and Reallocation*) de De Franceschi et al. [18], y el algoritmo de Toth y Tramontani [55]. El primero es una generalización para el DCVRP del trabajo de Sarvanov y Doroshko [53] para el TSP, en donde se explora un vecindario particular determinado por un criterio de remoción de nodos fijo (basado únicamente en la paridad de las posiciones). En el segundo, Toth y Tramontani investigan el problema de CG asociado a la relajación PL del RM de De Franceschi et al. [18], y proponen explorar una generalización del vecindario definido en [18] con un método de dos fases.

4.3.1. Algoritmo SERR

De Franceschi et al. [18] proponen el Algoritmo 4.2 que permite generar un gran número de secuencias nuevas con los nodos extraídos de las rutas de una solución para el DCVRP, para luego decidir como re-ubicarlas mediante la resolución (heurística) de un problema PLE.

4.3.2. LSA de Toth-Tramontani

Basados en las ideas anteriores, Toth y Tramontani [55] proponen un algoritmo similar. Remarcan la importancia de $\mathcal{F}$ (el conjunto de nodos extraídos) al determinar el vecindario $N(z_0, \mathcal{F})$, observando que malas elecciones de $\mathcal{F}$ pueden conducir a vecindarios malos donde no haya mejoras para z_0. Se pone el foco entonces en explorar diferentes vecindarios utilizando distintos criterios para elegir $\mathcal{F}$.

Dado que $N(z_0, \mathcal{F})$ puede ser muy grande y por lo tanto no explorable en un tiempo computacional razonable, se propone un algoritmo donde en cada iteración se determine un vecindario reducido $N(z_0, \mathcal{F}, \lambda) \subseteq N(z_0, \mathcal{F})$. En este vecindario reducido, λ un parámetro a fijar que impone una restricción sobre el costo de inserción de los nodos $v \in \mathcal{F}$, vistos como secuencias, en los puntos de inserción $i \in \mathcal{I}$ de la solución restringida. Se propone también la utilización de un modelo de re-ubicación reducido (RRM), para luego explorarlo parcialmente.

Dada una solución z_0 para el DCVRP (tomada de la literatura o construída heurísticamente), proponen el Algoritmo 4.3 de búsqueda local.

Algoritmo 4.2 SERR: *Selection, Extraction, Recombination and Reallocation* (De Franceschi et al. [18])

1. Inicialización:
   2. Aplicar una heurística inicial muy rápida para encontrar una primera solución del DCVRP (posiblemente infactible).
3. Selección:
   4. Aplicar un criterio de selección de nodos. El conjunto de nodos puede no ser consecutivo. Cualquier subconjunto de nodos es válido.
5. Extracción:
   6. Extraer los nodos seleccionados en el paso anterior, y construir la solución restringida del DCVRP. Todos los ejes de la solución restringida se agregan al conjunto $\mathcal{I}$ de puntos de inserción.
7. Recombinación:
   8. Guardar las *secuencias básicas* extraídas en un pool de secuencias $\mathcal{S}_P$.
   9. Utilizando técnicas heurísticas, derivar nuevas secuencias de entre los nodos extraídos, y agregarlas a $\mathcal{S}_P$.
10. Reubicación:
    11. Construir el modelo de re-ubicación (Reallocation Model) para decidir en qué puntos de inserción $i \in \mathcal{I}$ se deben re-ubicar las secuencias $s \in \mathcal{S}_P$ para obtener una mejor solución factible.
    12. Resolver el RM heurísticamente con la ayuda de un paquete de propósito general.
    13. Si se mejora la solución actual, actualizarla.
    14. Aplicar 3-OPT [51] para tratar de mejorar aún más la solución.
15. Terminación:
    16. Si en las últimas n iteraciones se mejora la solución, repetir desde **3.**. Caso contrario, terminar.

4.4. Heurística inicial

La solución inicial z_0 es muy importante para el esquema de destrucción/reparación propuesto, y a continuación se destacan algunas observaciones al respecto.

Utilizar soluciones z_0 de mucha calidad, ya sean obtenidas con otras técnicas o la mejor solución conocida en la literatura, ofrecen un margen de mejora muy chico para el modelo de re-ubicación, mientras que es de esperar que soluciones de baja calidad permitan encontrar en dicho modelo fácilmente asignaciones que produzcan mejores soluciones que z_0.

Resolver un modelo de re-ubicación, en el marco de un algoritmo de búsqueda local, tiene un costo computacional más elevado que el de operadores heurísticos simples (e.g., 2-opt). Se debe debe tener en cuenta entonces que el costo computacional de utilizar directamente un modelo de re-ubicación sobre soluciones de muy baja calidad podría no justificarse cuando sea posible aún aplicar operadores que demanden muy poco tiempo, y que permitan mejorar la calidad de

Algoritmo 4.3 LSA:*Local Search Algorithm* (Toth-Tramontani [55])
Entrada: λ

1. Inicialización:

2. Construir una lista Θ de todos los criterios de selección disponibles.

3. Selección:

4. Aplicar el siguiente criterio de selección en Θ para determinar el conjunto $\mathcal{F}$ de nodos extraídos.

5. Extracción:

6. Extraer los nodos seleccionados en el paso anterior, y construir la solución restringida del DCVRP.

7. Reducción:

8. Determinar el vecindario reducido $N(z_0, \mathcal{F}, \lambda) \subseteq N(z_0, \mathcal{F})$.

9. Construir la relajación LP del RRM con un conjunto inicial de variables.

10. Construcción:

11. Agregar al RRM un conjunto de variables prometedoras, generadas utilizando técnicas de pricing y generación de columnas.

12. Reubicación:

13. "Congelar" el conjunto actual de variables y agregar restricciones de integralidad.

14. Resolver a través de un MIP solver de propósito general.

15. Si la solución mejoró, actualizarla, aplicar una heurística de mejora (e.g., 3-OPT [51]) y repetir desde **1.**

16. Terminación:

17. Si la lista Θ está vacía, terminar. Caso contrario, repetir desde **3.**.

la solución z_0.

Por otro lado, encontrar una solución factible podría ser un problema muy difícil. Si se considera por ejemplo el DCVRP, encontrar una solución factible inicial es un problema $\mathcal{NP}$-Hard, i.e., a menos que $\mathcal{P} = \mathcal{NP}$, no existe un algoritmo para construir una solución factible (sin importar la calidad) en una instancia arbitraria en tiempo polinomial. Se considera entonces en este caso la posibilidad de inicializar el algoritmo con una solución no necesariamente factible (e.g., en el contexto del DCVRP, porque la restricción de distancia total es violada por algunas rutas). En la literatura, frente a esta situación y para manejar los casos de infactibilidad, De Franceschi et al. [18] proponen una modificación del algoritmo SERR presentado en 4.2, donde las restricciones de capacidad y distancia son utilizadas introduciendo variables slack con una penalización muy alta. Más aún, De Franceschi et al. consideran la posibilidad de empezar de una solución parcial, i.e. sin todos los nodos, y usar el algoritmo SERR para ubicar los nodos no utilizados, y obtener una solución factible completa.

En la literatura, aún en el marco del DCVRP, De Franceschi et al. [18] utiliza soluciones iniciales obtenidas por el método clásico de 2 fases de Fisher-Jaikumar [24], y también soluciones

obtenidas a través del método Sweep similar al propuesto por Gillett-Miller [26]. Heurísticamente divide a los clientes en clusters según el ángulo respecto del depósito para tratar de evitar exceder la capacidad de los vehículos. Cuando los clusters son constuídos con éxito, se generan rutas aplicando la heurística nearest-neighbor [32] seguido de una fase de 3-opt. Con este algoritmo típicamente consiguen rutas que satisfacen las restricciones de distancia (más detalle en [18]).

Toth y Tramontani [55] utilizan para el mismo problema el algoritmo Granular Tabu Search propuesto por Toth y Vigo [57], que es autocontenido (i.e., no requiere proveer una solución inicial), y con el cual obtienen soluciones de mucha calidad en algunos casos, pero también soluciones muy lejos del óptimo en otros, haciendo posible experimentar los métodos en ambas situaciones.

En el contexto de otro problema, para el OVRP, Salari et al. [52] aplican este esquema con soluciones iniciales tomadas de la literatura, (en muchos casos la mejor solución conocida), con la intención de encontrar mejoras. Esta alternativa es viable siempre y cuando existan instancias de benchmark para el problema considerado.

4.5. Criterios de selección de nodos

El criterio utilizado para seleccionar el conjunto de nodos a extraer es un punto crucial para determinar el vecindario a explorar por el algoritmo de búsqueda local basado en el modelo de re-ubicación.

En el trabajo de Sarvanov [53], el esquema de extracción-reubicación se propone para el TSP, con un vecindario construído a partir de un criterio de remoción *muy rígido* basado en la paridad de los nodos, donde dada una solución para el TSP, todos los nodos en posiciones pares o impares son removidos.

Este esquema es muy estricto en el sentido de que, si bien es posible determinar aleatoriamente la paridad de las posiciones a remover, no ofrece posibilidad a extracciones locales, o remoción de nodos consecutivos. El esquema podría no ser único, sino una composición de distintos esquemas orientados para buscar mejoras de alguna forma específica, i.e., podría aplicarse inicialmente un criterio muy general, para soluciones de poca calidad, mientras que en soluciones muy buenas, el criterio de remoción esté localizado en explorar determinado atributo de la solución.

En particular, De Franceschi et al. [18] propone 3 criterios de selección de nodos:

- Random-Alternate (RA):
  Similar al propuesto por Sarvanov y Doroshko [53] para el Problema del Viajante de Comercio, se selecciona en un conjunto aleatorio de rutas todos los nodos en posiciones pares, mientras que en el conjunto restante de rutas se seleccionan todos los nodos en posiciones impares. La paridad de las posiciones se determina recorriendo cada ruta en alguna dirección aleatoria.

- Scattered (SCT):
  Cada nodo tiene una probabilidad uniforme del 50 % de ser extraído. A diferencia del anterior, este esquema permite la extracción de nodos consecutivos.

- Neighborhood (NGH):
  Se pone el foco en un nodo semilla v^* y se remueven los nodos v con una probabilidad inversamente proporcional a la distancia c_{vv^*} de v a v^*.
  Para determinar el nodo semilla, se asigna un puntaje a todos los nodos (proporcional al número de nodos cercanos) y se construye una lista decreciente por puntaje, para que en cada iteración de NGH se seleccione el siguiente nodo en la lista (de forma circular) para jugar el rol de nodo semilla v^*.

En [18] se menciona que los esquemas RA y SCT son más apropiados para mejorar soluciones iniciales, mientras que NGH maneja mejor las soluciones obtenidas después de algunas iteraciones del algoritmo de búsqueda local. Por esta razón, en la implementación de SERR se alterna entre los esquemas aplicando una cantidad de veces RA, luego otra cantidad de vecces SCT, y finalmente se utiliza NGH para las iteraciones restantes. En caso de que se encuentre una mejor solución, este esquema de aplicaciones combinadas de criterios de remoción se repite desde el principio, volviendo a empezar por la aplicación de RA.

Toth y Tramontani [55] utilizan ideas similares para construir $\mathcal{F}$ basados en resultados experimentales que sugieren que los criterios aleatorios funcionan en general mejor que los criterios determinísticos. Proponen una versión modificada de RA, SCT y NGH. En la adaptación de RA, para cada ruta se determina aleatoriamente quitar todos los nodos en posiciones pares o impares. Generalizan SCT como SCT(p) de manera que cada nodo tiene una probabilidad p de ser extraído (p es un parámetro fijo). Con respecto a NGH, utilizan una versión NGH$'$ en donde el puntaje asociado a un nodo v en la lista para elegir v^* está determinado por la distancia promedio de los 10 nodos más cercanos al nodo v, y en donde N se ordena de forma creciente.

Nuevamente, resultados experimentales sugieren el uso de RA y SCT(p) para soluciones en iteraciones iniciales del algoritmo propuesto (ver 4.3), mientras que NGH$'$ es apropiado para mejorar soluciones de buena calidad.

4.6. Formulaciones del RM

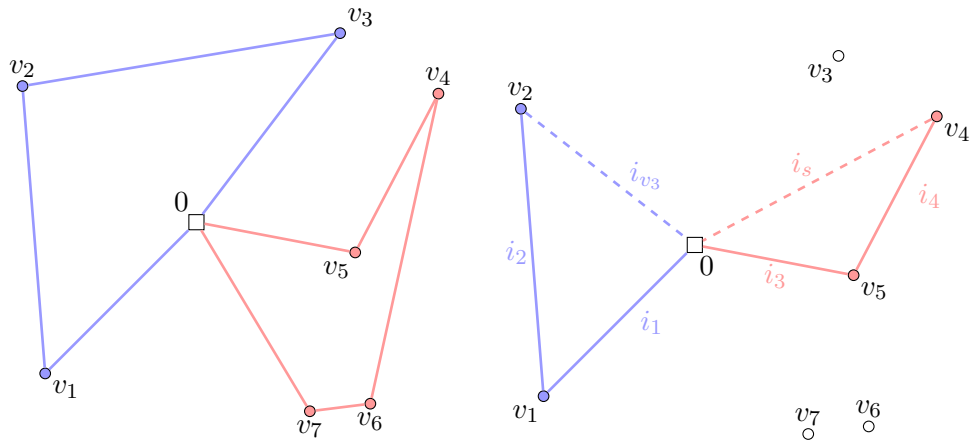
A continuación se formalizan algunas definiciones y se presenta la formulación para el RM propuesta por De Franceschi et al. [18] en el contexto del DCVRP, que se utiliza para motivar la formulación del RM para el VRPPD uno-a-uno.

Se denota con $\mathcal{Z}$ al conjunto de todas las soluciones factibles para el DCVRP.

Sea $z_0 \in \mathcal{Z}$ una solución factible, y el conjunto de nodos $\mathcal{F} \subseteq V\backslash\{0\}$, (donde 0 denota el depósito), se recuerda que $z_0(\mathcal{F})$ denota la solución restringida, obtenida de z_0 a partir de la extracción de todos los nodos $v \in \mathcal{F}$.

Se recuerda que $\mathcal{I} = \mathcal{I}(z_0, \mathcal{F})$ denota el conjunto de todos los ejes en $z_0(\mathcal{F})$, y se define $\mathcal{S} = \mathcal{S}(\mathcal{F})$ como el conjunto de todas las secuencias que pueden construirse combinando los nodos de $\mathcal{F}$.

Se dice que cada eje $i \in \mathcal{I}$ es un potencial *punto de inserción* para cada *secuencia* $s \in \mathcal{S}$.



(a) Una solución factible inicial z_0 (b) Solución restringida $z_0(\mathcal{F})$

$$\mathcal{F} = \{v_3, v_6, v_7\}$$
$$\mathcal{I} = \{i_{v_3}, i_s, i_1, i_2, i_3, i_4\}$$
$$\mathcal{S}(\mathcal{F}) = \{(v_3), (v_6), (v_7), (v_3, v_6), (v_3, v_7), (v_7, v_6), (v_3, v_6, v_7), (v_3, v_7, v_6), (v_6, v_3, v_7)\}$$

(c) Descripción por extensión de $\mathcal{I}$, $\mathcal{F}$ y $\mathcal{S}(\mathcal{F})$ para z_0 de la Figura 4.2(a)

Figura 4.2.: Un ejemplo de $\mathcal{F}$, $\mathcal{I}$ y $\mathcal{S}(\mathcal{F})$ para una solución del DCVRP.

Formalmente, las secuencias básicas son aquellas secuencias s que pertenecen a alguna ruta en la solución original z_0, y no existe otra secuencia en z_0 que contenga a s., mientras que el punto de inserción pivote i_s es el (único) punto de inserción desde donde fue extraída $s \in \mathcal{S}_B$.

Se recuerda también que para cada $\ddagger \in \mathcal{Z}$ y $\mathcal{F} \subseteq V \backslash \{0\}$ se denota con $N(z_0, \mathcal{F})$ al conjunto de todas las soluciones factibles $z_0 \in \mathcal{Z}$ que pueden obtenerse a través de la re-ubicación de todos los nodos $v \in \mathcal{F}$ mediante secuencias, en la solución restringida $z_0(\mathcal{F})$.

En general, $N(z_0, \mathcal{F})$ es un vecindario *exponencial* para la solución z_0, que puede ser visto como una extensión del vecindario propuesto originalmente por Sarvanov y Doroshko [53], e independientemente por Gutin [31] para el TSP (ver detalles en [18]). Se tiene que $N(z_0, \mathcal{F})$ depende fuertemente de $\mathcal{F}$ y en particular:

- $N(z_0, \emptyset) = \{z_0\}$, i.e. no extraer nodos produce un vecindario en donde solo está la solución original z_0.
- $N(z_0, V \backslash \{0\}) = \mathcal{Z}$ para todo z_0 en $\mathcal{Z}$, i.e. extraer todos los nodos cliente determina el vecindario de todas las soluciones posibles para la instancia de DCVRP en cuestión,

haciendo que al metodología pierda sentido ya que se destruye por completo la solución inicial.

Se dice que un punto de inserción $i = (a, b) \in \mathcal{I}$ aloja los nodos $\{v_j \in \mathcal{F} : j = 1, \ldots, h\}$ a través de la secuencia $s = (v_1, \ldots, v_h) \in \mathcal{S}$ si el eje (a, b) en la solución restringida es reemplazado por los ejes $(a, v_1), (v_1, v_2), \ldots, (v_h, b)$ en la nueva solución factible.

Por ejemplo, en la Figura 4.2(b), si se re-ubica (v_6, v_7) en $i_3 \in \mathcal{I}$, se dice que i_3 aloja a $v_6, v_7 \in \mathcal{F}$, y la re-ubicación se haría reemplazando el eje $(0, v_5)$ en 4.2(b), por los ejes $(0, v_6), (v_6, v_7), (v_7, v_5)$.

Se agrega también en 4.2(c) una descripción completa de todos los caminos en $\mathcal{F}$, i.e. $\mathcal{S}(\mathcal{F})$.

Se recuerda que para asegurar la factibilidad (en el sentido del DCVRP) de la solución de re-ubicación de nodos, se asocia secuencias $s \in \mathcal{S}$ a puntos de inserción $i \in \mathcal{I}$ de forma tal que todo nodo $v \in \mathcal{F}$ sea cubierto exactamente una vez por las secuencias s asociadas a puntos de inserción (i.e., todos los clientes son visitados exactamente una vez), y que además cada punto de inserción $i \in \mathcal{I}$ tenga asociado a lo sumo a una secuencia $s \in \mathcal{S}$.

En el ejemplo de la Figura 4.2(b), re-ubicar solo (v_6, v_7) en i_3, por ejemplo, no generaría una solución factible de DCVRP ya que v_3 no sería visitado (más allá de si la re-ubicación mejora o no respecto de z_0 en 4.2(a)) por ningún vehículo. Por otro lado, en el mismo ejemplo, tampoco sería factible re-ubicar $s_1 = (v_6, v_7)$ y $s_2 = (v_3)$ en un mismo punto de inserción i_3, dado que no está bien definida la asignación: si debe ser $v_5 \to s_1 \to s_2 \to 0$, o debe ser $v_5 \to s_2 \to s_1 \to 0$. Incluso la cantidad de posibilidades es mayor si se intenta asignar más de 2 secuencias al mismo punto de inserción. Por esta razón, una buena solución es restringir la re-ubicación con a lo sumo una secuencia por punto de inserción. Observar que no se pierden soluciones de re-ubicación, dado que en los casos donde se desee asignar más de una secuencia para $i \in \mathcal{I}$, siempre es posible construir otra secuencia que las contenga, e.g. se puede considerar $s_3 = s_1 s_2$ y la re-ubicación s_3 en i_3.

Para toda $s \in \mathcal{S}$ se define $c(s)$ como la suma de los costos de los ejes en s, y $\mathcal{R}$ como el conjunto de rutas en $z_0(\mathcal{F})$. Se define también $V(s)$ como el conjunto de nodos de una secuencia $s \in \mathcal{S}$, y $\mathcal{S}(v) = \{s \in \mathcal{S} : v \in V(s)\}$ como el conjunto de secuencias que contienen el nodo $v \in \mathcal{F}$.

En la Figura 4.2(c), se tiene por ejemplo que si $s = (v_6, v_7)$ entonces $V(s) = \{v_6, v_7\}$, mientras que $S(v_7) = \{(v_7), (v_3, v_7), (v_6, v_7), (v_3, v_6, v_7), (v_6, v_3, v_7), (v_6, v_7, v_3)\}$.

Para cada punto de inserción $i = (a, b) \in \mathcal{I}$ y para cada secuencia $s = (v_1, \ldots, v_h) \in \mathcal{S}$ se define γ_{si} como el costo extra por asignar la secuencia s al punto de inserción i, teniendo en cuenta la mejor orientación posible, i.e. $\gamma_{si} := c(s) - c_{a,b} + \text{mín}\{c_{a,v_1} + c_{v_h,b}, c_{a,v_h} + c_{v_1,b}\}$, considerando el *mínimo* entre $c_{a,v_1} + c_{v_h,b}$ y $c_{a,v_h} + c_{v_1,b}$ para tener en cuenta la mejor forma de orientar la asignación de s a i.

En el ejemplo 4.3 se muestran formas de asignar la secuencia (v_6, v_7) al punto de inserción i_3 de la Figura 4.2(b), teniendo en cuenta las dos orientaciones posibles.

Si bien las orientaciones de la secuencia (v_6, v_7) pueden ser vistas como dos secuencias distintas, $s_1 = (v_6, v_7)$ y $s_2 = (v_7, v_6)$, es posible definir las secuencias de forma que s_1 y s_2 sean

indistinguibles, aprovechando la definición del costo de inserción y definiendo la re-ubicación para que siempre se realice con la mejor orientación posible (en el sentido del costo de inserción). Esto impacta sobre el tamaño del conjunto de secuencias $\mathcal{S}$ que se debe considerar explícitamente.

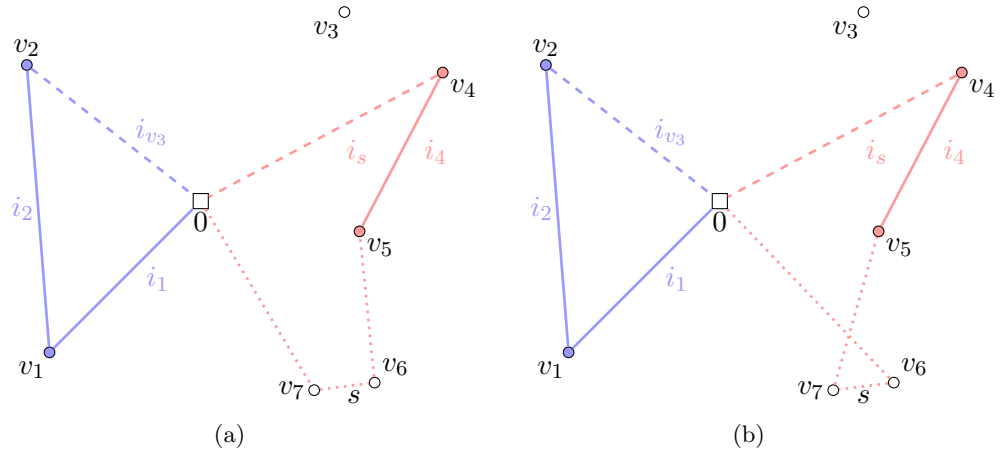


Figura 4.3.: Un ejemplo de re-ubicaciones posibles en el punto de inserción $(0, v_5) \in \mathcal{I}$ en la solución restringida de la Figura 4.2(b)

Con la notación anterior, el modelo de re-ubicación (Reallocation Model) se basa en las siguientes variables de decisión:

$$x_{si} = \begin{cases} 1 & \text{la secuencia } s \in \mathcal{S} \text{ es asignada al punto de inserción } i \in \mathcal{I} \\ 0 & \text{en caso contrario} \end{cases}.$$

y el modelo se formula como:

$$\text{mín} \sum_{s \in \mathcal{S}} \sum_{i \in \mathcal{I}} \gamma_{si} x_{si} \tag{4.1}$$

s.t.

$$\sum_{s \in \mathcal{S}(v)} \sum_{i \in \mathcal{I}} x_{si} = 1 \qquad v \in \mathcal{F} \tag{4.2}$$

$$\sum_{s \in \mathcal{S}} x_{si} \leq 1 \qquad i \in \mathcal{I} \tag{4.3}$$

$$x_{si} \in \{0, 1\} \qquad s \in \mathcal{S}, i \in \mathcal{I} \tag{4.4}$$

Nuevamente, una solución para el RM determina una asignación de las secuencias $s \in \mathcal{S}$ a los puntos de inserción $i \in \mathcal{I}$ en la solución restringida $z_0(\mathcal{F})$. Para que dicha asignación sea una solución factible del VRP, se agrega la restricción (4.2), exigiendo que cada nodo removido $v \in \mathcal{F}$ sea asignado (a través de una secuencia $s \in \mathcal{S}(v)$) a exactamente un único punto de

inserción $i \in \mathcal{I}$. Se agrega también la restricción (4.3) para que cada punto de inserción aloje a lo sumo una secuencia $s \in \mathcal{S}$.

En el contexto de la variante del VRP con restricciones de distancia y capacidad (DCVRP), De Franceschi et al. [18] proponen una versión del RM que agrega restricciones adicionales, utilizando la siguiente notación:

- Dada $s \in \mathcal{S}$ se denota con $q(s)$ a la demanda de la secuencia s, definida como la suma de las demandas q_v asociadas a los nodos $v \in V(s)$, i.e. $q(s) = \sum_{v \in V(s)} q_v$.
- Dada una ruta $r \in \mathcal{R}$ se denota con $\tilde{d}(r)$ a la distancia total de r en $z_0(\mathcal{F})$, i.e. $\tilde{d}(r) = \sum_{(i,j) \in A(r)} d_{i,j}$
- Dada una ruta $r \in \mathcal{R}$ se denota con $\tilde{q}(r)$ a la demanda total de r en $z_0(\mathcal{F})$, i.e. $\tilde{q}(r) = \sum_{v \in V(r)} q_v$.

Las restricciones adicionales son:

$$\sum_{s \in \mathcal{S}} \sum_{i \in \mathcal{I}(r)} q(s)\, x_{si} \leq C - \tilde{q}(r) \quad r \in \mathcal{R} \tag{4.5}$$

$$\sum_{s \in \mathcal{S}} \sum_{i \in \mathcal{I}(r)} \gamma_{si}\, x_{si} \leq D - \tilde{d}(r) \quad r \in \mathcal{R} \tag{4.6}$$

Vale observar que esta formulación del modelo tiene un gran número de variables (exponencial en $|\mathcal{F}|$), por lo cual es muy costoso computacionalmente manipularlo en la práctica. De Franceschi et al. [18] manejan de manera heurística la relajación PL del mismo.

Esta versión del modelo es también estudiada por Toth y Tramontani [55] buscando disminuir el costo computacional que requiere, y refinada en una versión llamada *modelo de re-ubicación reducido* (Reduced Reallocation Model), que se presenta a continuación.

4.7. Reducción del vecindario

Como ya se mencionó, $N(z_0, \mathcal{F})$ define un vecindario exponencial para la solución z_0, que en el peor de los casos cuando $\mathcal{F} = V \backslash \{0\}$, resulta $N(z_0, \mathcal{F}) = \mathcal{Z}$.

Es por esto que tiene sentido tratar de explotar propiedades del problema para definir criterios que permitan reducir el vecindario, y en consecuencia, el tiempo computacional requerido para explorarlo. Asímismo, para que la reducción tenga sentido se debe tener cuidado en no descartar soluciones *buenas*, buscando un equilibrio entre reducir para ganar efectividad en la exploración, y mantener la calidad del vecindario.

Una forma de reducir el vecindario es aprovechando la estructura del RM. En el contexto del DCVRP, para cada secuencia $s \in \mathcal{S}$ y para cada punto de inserción i, se dice que s es factible para i si s puede ser asociada a i sin violar restricciones de capacidad y distancia para la ruta r_i que contiene a i. Es posible estudiar la relación entre las secuencias $s \in \mathcal{S}$ y los puntos

de inserción $i \in \mathcal{I}$ para tratar de generar variables x_{si} de manera inteligente, reduciendo el costo computacional de resolver el RM, y en consecuencia explorando de manera más eficiente $N(z_0, \mathcal{F})$. Recordar que en el algoritmo de búsqueda local se intenta generar nuevas variables x_{si} considerando todo punto de interción $i \in \mathcal{I}$ y toda secuencia $s \in \mathcal{S}$. Sin embargo, el costo de inserción γ_{si} puede ser muy alto, o incluso muchas secuencias s pueden ser infactibles para i, y entonces sería conveniente evitarlo.

Siguiendo esta línea, Toth y Tramontani [55] proponen remover a priori todas las secuencias infactibles junto con aquellas que sean *malas* (aquellas con un costo de inserción alto).

A tal fin, se construye para cada $i \in \mathcal{I}$ un conjunto reducido de nodos $\mathcal{F}_i \subseteq \mathcal{F}$, y se decide que todas las secuencias generadas para i sean secuencias con nodos en $\mathcal{F}_i$, (i.e., $s \in \mathcal{S}_i = \{s \in \mathcal{S} : V(s) \subseteq \mathcal{F}_i\}$).

Observar que si $v \notin \mathcal{F}_i$, entonces para cualquier secuencia s que contenga a v, se tiene que $s \notin \mathcal{S}_i$, haciendo que nunca se genere s asociada a i, impactando directamente en el vecindario $N(z_0, \mathcal{F})$ dejando afuera todas las soluciones que consideren la asignación de s a i, y significando una reducción considerable en dicho vecindario y en el modelo.

Nuevamente, para que la reducción tenga sentido, el criterio con el que se construye $\mathcal{F}_i$ debe definirse con cuidado para no perder potenciales mejoras para z_0 en $N(z_0, \mathcal{F})$.

Basados en esta idea, Toth y Tramontani [55] proponen el modelo de re-ubicación reducido (RRM, por sus siglas en inglés), utilizando la siguiente notación:

- Dada $r \in \mathcal{R}$ se denota $\mathcal{I}(r)$ al conjunto de puntos de inserción de r.
- Para cada $s \in \mathcal{S}$ se define $\mathcal{I}_s$ como el conjunto de puntos de inserción factibles para s, i.e. $\{i \in \mathcal{I} : s \in \mathcal{S}_i\}$.

De esta manera, para cada $s \in \mathcal{S}$, $r \in \mathcal{R}$, se tiene $\mathcal{I}_s(r) = \mathcal{I}_s \cap \mathcal{I}(r)$, y de forma similar al RM, el RRM se formula como:

$$\text{mín} \sum_{s \in \mathcal{S}} \sum_{i \in \mathcal{I}_s} \gamma_{si} x_{si} \tag{4.7}$$

$$\text{sujeto a} \sum_{s \in \mathcal{S}(v)} \sum_{i \in \mathcal{I}_s} x_{si} = 1 \qquad v \in \mathcal{F} \tag{4.8}$$

$$\sum_{s \in \mathcal{S}_i} x_{si} \leq 1 \qquad i \in \mathcal{I} \tag{4.9}$$

$$x_{si} \in \{0, 1\} \qquad s \in \mathcal{S}, i \in \mathcal{I}_s \tag{4.10}$$

con las restricciones de distancia y capacidad máxima definidas como:

$$\sum_{s\in\mathcal{S}}\sum_{i\in\mathcal{I}_s(r)} q(s)x_{si} \leq C - \tilde{q}(r) \quad r \in \mathcal{R} \tag{4.11}$$

$$\sum_{s\in\mathcal{S}}\sum_{i\in\mathcal{I}_s(r)} \gamma_{si}x_{si} \leq D - \tilde{d}(r) \quad r \in \mathcal{R} \tag{4.12}$$

donde a diferencia del RM las restricciones se adaptan considerando que las variables x_{si} están definidas para secuencias s que son *buenas* para el punto de inserción i, i.e. $s \in \mathcal{S}_i$.

Con respecto a la construcción de $\mathcal{F}_i$, Toth y Tramontani proponen un criterio basado en la factibilidad de los nodos $v \in \mathcal{F}$ vistos como secuencias unitarias. Definen la noción de factibilidad para un nodo $v \in \mathcal{F}$ respecto de un punto de inserción $i \in \mathcal{I}$, de forma tal que v es factible para i si la secuencia (v) es factible para i, y denotan $\mathcal{F}_i^* = \{v \in \mathcal{F} : v \text{ es factible para } i\}$ y $\mathcal{I}_v^* = \{i \in \mathcal{I} : v \in \mathcal{F}_i^*\}$.

Toth y Tramontani [55] proponen una reducción de $N(z_0, \mathcal{F})$ basados en la siguiente aserción heurística: *si un nodo v tiene que ser alojado por una ruta $r \in \mathcal{R}$, entonces probablemente se asocie v a uno de los puntos de inserción $i \in \mathcal{I}(r)$ más cercanos a v.*

De esta manera, la reducción debe satisfacer para todo $v \in \mathcal{F}$:

- v debe estar asociado a su pivote i_v (así es posible recuperar la solución z_0).
- v debe estar asociado al menos a un punto de inserción $i \in \mathcal{I}(r)$ para cada $r \in \mathcal{R}$.
- Si el costo de inserción de la secuencia (v) al punto de inserción i es "muy alto", entonces se remueve v de $\mathcal{F}_i$ (de esta manera se está removiendo toda $s \in \mathcal{S}(v)$ de $\mathcal{S}_i$).

Para cada $v \in \mathcal{F}$ y para cada $i \in \mathcal{I}_v^*$, se considera γ_{vi} como el costo de inserción de la secuencia (v) al punto de inserción i. Se define para cada ruta $r \in \mathcal{R}$ y para cada $v \in \mathcal{F}$ el costo promedio de inserción de la secuencia (v) a la ruta r como δ_{vr}, computado como:

$$\delta_{vr} = \frac{\sum_{i\in\mathcal{I}(r)\cap\mathcal{I}_v^*} \gamma_{vi}}{|\mathcal{I}(r) \cap \mathcal{I}_v^*|} \tag{4.13}$$

La idea es utilizar el costo promedio para construir heurísticamente $\mathcal{F}_i$ de la siguiente forma:

$$\mathcal{F}_i = \{v \in \mathcal{F}_i^* : \gamma_{vi} \leq \lambda\, \delta_{vr_i}\} \cup \{v \in \mathcal{F} : i = i_v\} \tag{4.14}$$

donde λ es un parámetro a fijar experimentalmente. Toth y Tramontani denotan bajo este criterio la reducción del vecindario como $N(z_0, \mathcal{F}, \lambda)$. Se tiene entonces que siempre $N(z_0, \mathcal{F}, \lambda) \subseteq N(z_0, \mathcal{F})$, y que mientras mayor sea λ, menor será la reducción, (i.e., $\lim_{\lambda\to\infty} N(z_0, \mathcal{F}, \lambda) = N(z_0, \mathcal{F})$).

4.8. Esquemas de Pricing

Dado que el número de variables puede resultar muy grande, es necesario abordar la resolución mediante técnicas de generación de columnas a partir de un modelo con un número reducido de variables que aseguren factibilidad. Cuando ya no se generan nuevas variables, se agregan restricciones de integralidad y se resuelve el modelo mediante un paquete de propósito general.

4.8.1. De Franceschi et al.

En el contexto del DCVRP, De Franceschi et al. [18] proponen generar inicialmente un conjunto de variables de forma conservadora para que sea fácil computar la relajación LP del RM, denotada $\mathcal{C}$(RM). Luego se generan más y más variables dentro de un ciclo de pricing en donde se resuelve $\mathcal{C}$(RM), y para un conjunto de variables x_{si} ($s \in \mathcal{S}$, $i \in \mathcal{I}$) construídas heurísticamente, se evalúa rc_{si} (el costo reducido) utilizando $\tilde{\pi}$, i.e. el óptimo dual de la relajación lineal. Si rc_{si} está por debajo de un umbral RC_{max} entonces se agrega la variable x_{si}, y se repite el proceso.

Dentro de este esquema, propone un método basado en dos fases.

En la primera fase se inicializa un pool de secuencias básicas $\mathcal{S}_B$ y se asocia cada $s \in \mathcal{S}_B$ al punto de inserción pivote i_s correspondiente. Esto permite asegurar la factibilidad del modelo y que la solución actual pueda reconstruirse simplemente asociando las secuencias básicas a su pivote.

La segunda fase corresponde al ciclo de pricing, en donde en cada iteración se calcula $\tilde{\pi}$, se considera cada $i \in \mathcal{I}$ y se construyen secuencias convenientes. Para esto, desarrollan un esquema de generación de secuencias controlado por los siguientes parámetros:

- L_{max}, que denota la longitud máxima de secuencias a construir.
- N_{min}, que denota la cantidad mínima de variables x_{si} que son agregadas al modelo para $i \in \mathcal{I}$, de longitud $|s|$, con menor rc_{si}.
- N_{max}, que dado $i \in \mathcal{I}$ denota la cantidad máxima de variables x_{si} que son agregadas al modelo para $i \in \mathcal{I}$, de longitud $|s|$. La diferencia con respecto a N_{min} es que una vez que se agregaron las N_{min} variables de menor r_{si} para un $i \in \mathcal{I}$, y una longitud de secuencia $|s|$, solo se seguirán agregando variables en caso de que rc_{si} esté por debajo del umbral RC_{max}.
- δ, el cual junto con RC_{max} determinan un criterio heurístico de aceptación de una variable x_{si} al modelo.

y aplican el algoritmo que se muestra a continuación:

Algoritmo 4.4 SERR: Pricing loop

Entrada: i: punto de inserción, rc^*: el costo reducido más chico generado hasta el momento, $\tilde{\pi}$: el óptimo dual de la relajación lineal, δ, N_{min}, N_{max}, L_{max}, RC_{max}.

1. Inicializar: $L := 0$, $S := \{<>\}$
2. $L := L + 1$
3. Para cada $s \in S$,
   4. Para cada $v \in \mathcal{F}$ tal que $v \notin s$,
      5. Generar todas las secuencias obtenidas de insertar, en cualquier posición, el nodo v en s
6. Para cada s obtenida en **5.**,
   7. Considerar la variable x_{si} y evaluar el costo reducido $rc_{si}(\tilde{\pi})$.
   8. Si $rc_{si}(\tilde{\pi}) < rc^*$, actualizar $rc^* := rc_{si}(\tilde{\pi})$.
9. $S = \emptyset$
10. Insertar en S las N_{min} secuencias s con menor $rc_{si}(\tilde{\pi})$, junto con las secuencias s tal que $rc_{si}(\tilde{\pi}) \leq \text{máx}\{RC_{max}, \delta rc^*\}$, insertando a lo sumo N_{max} secuencias. Las variables x_{si} correspondientes son agregadas a al modelo actual, si bien no se re-optimiza aún.
11. Si las secuencias insertadas en S tienen longitud menor que L_{max}, repetir desde **2.**. Caso contrario, considerar el siguiente punto de inserción.

Cuando el último punto de inserción es considerado, se recalcula $\mathcal{C}(\text{RM})$ y se repite. Los valores L_{max}, N_{min}, N_{max}, RC_{max} y δ son parámetros fijados experimentalmente.

En la segunda fase, De Franceschi et al. [18] proponen agregar heurísticamente un conjunto de variables adicionales x_{si} prometedoras. Consideran todas las secuencias s de longitud 2, y para cada $i \in \mathcal{I}$ que verifique $rc_{si}(\tilde{\pi}) \leq RC_{max}$, se agrega la variable x_{si} al modelo. También, solo para aquellas secuencias adicionales s que son agregadas (a través de alguna variable x_{si}, se consideran las extensiones s' de longitud 3 construídas a partir de s, insertando nodos $v \in \mathcal{F} \backslash V(s)$, y se agrega la variable $x_{s'i}$.

El proceso se repite hasta que no se agreguen nuevas variables al modelo. Se utilizan técnicas de *hashing* para evitar generar variables duplicadas.

4.8.2. Toth-Tramontani

Considerando la reducción del vecindario $N(z_0, \mathcal{F}, \lambda)$ para el DCVRP (ver Sección 4.7), que podría ser explorado resolviendo hasta optimalidad el modelo (4.7) (incluyendo las restricciones (4.11) y (4.12)), Toth y Tramontani [55] proponen hacer la exploración de manera parcial, nuevamente, por restricciones de tiempo computacional.

Se sugiere separar la generación de variables en dos etapas, distribuyendo de manera distinta el esfuerzo dedicado a generar nuevas variables en cada una. En la primera etapa se pone el foco en producir muy rápido una cantidad razonable de variables, aún cuando la calidad de estas no

sean de muy buena calidad. Ya en la segunda etapa, el modelo cuenta con todas las variables iniciales, y todas aquellas que se agregaron en la primera etapa.

Se propone un esquema basado en dos etapas: (1) *fast-pricing*; (2) generación de columnas.

Se define $\mathcal{V}_P$ y $\mathcal{S}_P$ como pools que contienen, respectivamente, todas las variables y secuencias generadas hasta el momento. Las variables en $\mathcal{V}_P$ son las variables presentes en el modelo, mientras que $\mathcal{S}_P$ contiene las secuencias que se consideran en cada iteración de *pricing* para generar una nueva variable x_{si}. Con esta notación, la construcción del vecindario reducido está dada por el siguiente algoritmo:

Algoritmo 4.5

1. (Inicialización)

2. $\mathcal{S}_P$=$\mathcal{S}_B$

3. $\mathcal{V}_P$=$\{x_{si} : s \in \mathcal{S}_B, i = i_s\}$

4. Para cada $i \in \mathcal{I}$,

5. Construir un número reducido de secuencias nuevas $s \in \mathcal{S}_i$ que se ajusten bien a i (tal que γ_{si} sea bajo), y agregar estas secuencias a $\mathcal{S}_P$ y las variables correspondientes a $\mathcal{V}_P$.

6. (Fast Pricing)

7. $\mathcal{S}_P = \mathcal{S}_P \cup \{s \in \mathcal{S} : |s| = 2\}$

8. Resolver $\mathcal{C}$(RRM)

9. Para cada $i \in \mathcal{I}$,

10. Para cada $s \in \mathcal{S}_P \cap \mathcal{S}_i$,

11. Si $rc_{si} \leq RC_{max}$ entonces agregar x_{si} a $\mathcal{V}_P$

12. Si se agregó al menos una variable a $\mathcal{V}_P$, repetir desde **8.**

13. (Generación de Columnas)

14. Para cada $i \in \mathcal{I}$,

15. Resolver el Problema de Generación de Columnas asociado a i.

16. Si se encontró al menos una variable, agregar todas a $\mathcal{V}_P$, resolver $\mathcal{C}$(RRM) y repetir desde **14.**

Después de **1.** es posible asegurar que existe una solución factible para el RRM (la solución z_0), y se pueden utilizar los costos reducidos para evaluar la "calidad" de las variables. El paso de fast pricing en **6.** pretende reducir el costo computacional requerido para el paso de generación de columnas. Finalmente, en **13.** se pretende buscar variables de "buena calidad" resolviendo (heurísticamente) el Problema de Generación de Columnas asociado a i (ver [55] para más detalle). Dicho problema es un punto crucial en la construcción del vecindario, y el que más tiempo consume, si bien produce variables muy buenas incluso en soluciones de mucha calidad. Aún así, resultados experimentales muestran que no son menos importantes las etapas de Inicialización (**1.**) y Pricing (**6.**), que construyen una buena estructura inicial para el RRM.

Nuevamente, se utilizan técnicas de *hashing* para evitar generar variables duplicadas.

5. Algoritmo de Búsqueda Local para el VRPPD uno-a-uno

En este capítulo se presenta un esquema de refinamiento heurístico basado en la resolución de un modelo de re-ubicación (RM) para la variante VRPPD1-1 en base a las ideas introducidas en el capítulo anterior.

Como ya se mencionó, el esquema básico consta de dos fases: (1) *extracción de nodos*, donde un conjunto de nodos son seleccionados con algún criterio y removidos de la solución actual; (2) *re-inserción de nodos*, donde se re-ubican los nodos extraídos para construir una nueva solución, garantizando que esta es al menos tan buena como la original.

Dado que VRPPD1-1 se caracteriza por restricciones de precedencia, los esquemas propuestos en la literatura para VRP no son aplicables de forma directa. En las siguientes secciones se presentan las dificultades principales y adaptaciones realizadas, junto con nuevas propuestas para extender el framework que originalmente utiliza De Franceschi et al. [18] en el marco del DCVRP, y se aprovechan también algunas de las ideas que sugieren Toth y Tramontani [55].

Para introducir algunos conceptos se considera el siguiente ejemplo. En la Figura 5.1(a) se muestra una solución factible para un problema en una instancia euclidiana con 12 nodos (clientes), en donde se distinguen nodos de recolección $\{p_1, p_2, p_3, p_4, p_5, p_6\}$ y nodos de entrega $\{d_1, d_2, d_3, d_4, d_5, d_6\}$, mientras que se denota con 0 al depósito. La relación entre recolecciones y entregas exige que p_i preceda d_i, para $i \in \{1, \ldots, 6\}$. Observar que la restricción de precedencia obliga en general a que cuando p_i esté ubicado en la ruta r, también d_i esté ubicado en la ruta r, en particular después que p_i.

Se aplica ahora la fase de extracción. Siguiendo las ideas aplicadas al VRP, la Figura 5.1(b) muestra (con círculos blancos) el resultado de seleccionar los nodos en posiciones pares. En este caso el conjunto de nodos a extraer es $\mathcal{F} = \{p_2, d_2, d_3, d_4, d_5, d_6\}$.

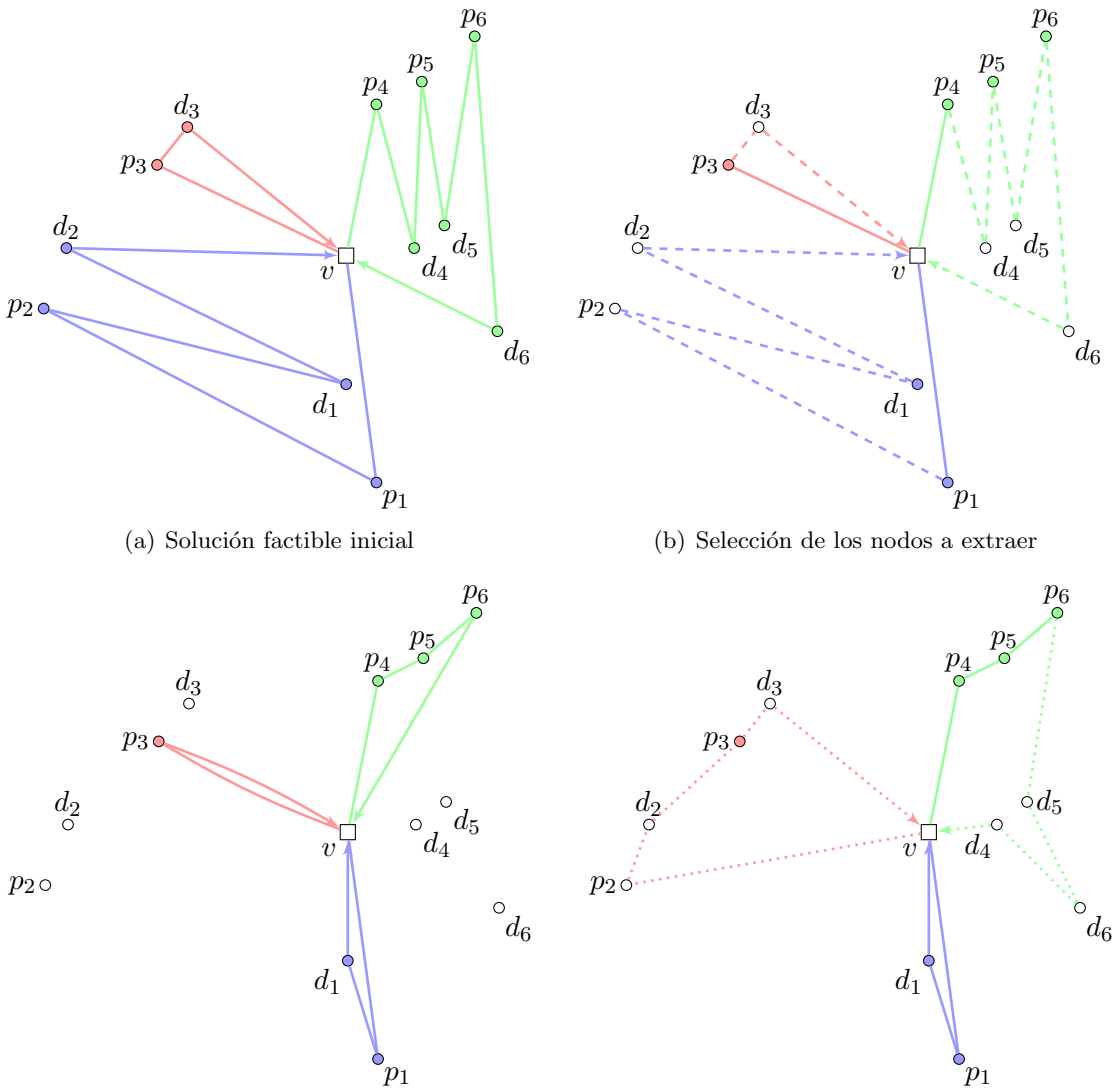
La Figura 5.1(c) muestra la *solución restringida* $z_0(\mathcal{F})$. Se recuerda que ésta resulta de "recortar" los ejes asociados a los nodos en $\mathcal{F}$, determinando los potenciales puntos de inserción $\mathcal{I} = \mathcal{I}(z_0, \mathcal{F}) = \{(0, p_3), (p_3, v), (0, p_4), (p_4, p_5), (p_5, p_6), (p_6, v), (0, p_1), (p_1, d_1), (d_1, 0)\}$.

Aplicada la remoción, dada la distinción entre clientes de recolección y clientes de entrega, y la relación de precedencia entre estos, para un par (p_i, d_i) arbitrario es posible distinguir 3 casos:

(a) Se extrae el vértice p_i de recolecccción sin su entrega d_i asociada.

(b) Se extrae el vértice de entrega d_i sin su recolección p_i asociada.

(c) Se extrae el par (p_i, d_i) simultáneamente.



(a) Solución factible inicial

(b) Selección de los nodos a extraer

(c) Solución restringida (infactible)

(d) Solución de re-ubicación

Figura 5.1.: Un ejemplo de extraccción y re-ubicación de nodos para el VRPPD 1-1.

Tanto en el caso (a) como en el caso (b), será infactible re-ubicar al vértice extraído en una ruta distinta de aquella desde donde fue removido. En cambio, en el caso (c) se tiene libertad para cambiar de ruta al vértice de recolección y entrega mediante la re-ubicación del par (p_i, d_i), si bien deben ser asignados dentro de una misma ruta.

Otra observación respecto del caso (a) y (b), es que el nodo removido solo puede ser re-ubicado, en su ruta original, en posiciones que respeten la precedencia. Esto impone además un sentido sobre la ruta, donde a diferencia del VRP, todas las rutas son ahora factibles o infactibles dependiendo qué sentido se les asigne. En el marco del VRP (sin recolecciones y entregas), si se asume que las distancias son simétricas, en principio no hace falta que las rutas tengan alguna

dirección en particular.

En 5.1(d), se puede ver cómo d_3 es asignado a su ruta original dado que p_3 no fue removido, y como en particular, este tuvo que ser re-ubicado en el único punto de inserción que está después de p_3, que es el eje (p_3, v).

Finalmente, la Figura 5.1(d) muestra la solución resultante de la re-ubicación, en donde d_3 fue re-ubicado en su posición original, d_4, d_5 y d_6 se mantuvieron en la ruta original, pero cambiaron de posición, mientras que p_2 y d_2 se re-ubicaron en una ruta distinta a aquella desde la que fueron extraídos.

Este ejemplo sencillo muestra que la fase de extracción de nodos no es tan flexible dado el impacto que tiene en la fase de re-inserción. Por lo tanto es conveniente adaptar ambas fases al problema particular que se intenta resolver.

En las próximas secciones se introducen y analizan alternativas.

5.1. LSA para el VRPPD 1-1

Siguiendo los lineamientos de las ideas presentadas en el capítulo anterior, se propone un algoritmo original de refinamiento que consta principalmente de 4 etapas:

Para implementar las ideas del esquema introducido al inicio de este capítulo, considerando los algoritmos (4.2) y (4.3), se propone el siguiente algoritmo de refinamiento basado en búsqueda local, que consta principalmente de 4 etapas:

1. Una etapa de inicialización en la cual se construye una solución factible para el problema, de manera muy rápida.

2. Una fase de refinamiento basado en operadores de búsqueda local, donde la intención es aumentar considerablemente la calidad de la solución inicial z_0 a un costo computacional relativamente bajo. El proceso se repite hasta que se alcanza un óptimo local para el vecindario de los operadores.

3. Una segunda fase de refinamiento que se realiza a través de la exploración parcial de la reducción del vecindario $N(z_0, \mathcal{F})$ mediante la resolución (heurística) del RRM. Es la etapa más importante del algoritmo en la cual se pretende mejorar aún más la calidad de la solución obtenida por los operadores de la segunda fase. Utilizando una lista Θ de criterios de remoción de nodos, se aplica durante t unidades de tiempo el paradigma de destrucción/reparación. Se inicializa el modelo con un subconjunto de variables iniciales, considerando la reducción del vecindario, a través de la utilizanción de $\mathcal{S}_i$ en la construcción de secuencias s para i. En el siguiente paso, se ingresa a una adapación del ciclo de pricing de De Franceschi et al. que incorpora también la noción de reducción de vecindario. La idea es generar nuevas variables de manera inteligente, restringiendo el esfuerzo computacional con una cota en la cantidad de iteraciones, mientras se mejore la relajación. Finalmente, se agregan las restricciones de integralidad sobre las variables, y se resuelve el problema mediante un paquete de resolución de PLEMs de propósito general. Dado que siempre se consideran las variables x_{si} básicas, i.e., tal que s es secuencia básica con i el punto

de inserción pivote, se consigue una nueva solución que es al menos tan buena como la original. En caso de que esta nueva solución sea de mejor calidad, se actualiza la solución actual, y se reinicia la lista de criterios Θ, comenzando desde el primer criterio en la lista en la siguiente iteración de destrucción/reparación. Esta fase termina cuando o bien se aplicaron todos los criterios en Θ, o se alcanza un límite de tiempo para la fase de refinamiento basado en PLE.

Algoritmo 5.1 VRPPD1-1-LSA:*Algoritmo de búsqueda local para el VRPPD1-1*

1. *Inicialización:*

2. Construir heurísticamente una solución factible inicial.

3. *Refinamiento inicial basado en operadores de búsqueda local:*

4. Definir una lista Φ de aplicación de los operadores básicos de búsqueda local.

5. Mientras z_0 cambie, aplicar cada $\phi \in \Phi$ actualizando z_0, i.e., $z_0 = \phi(z_0)$

6. *Refinamiento basado en PLE:*

7. Definir una lista Θ de criterios de remoción de nodos.

8. Durante t unidades de tiempo,

9. Aplicar el siguiente criterio en Θ para construir $\mathcal{F}$.

10. Extraer los nodos seleccionados en el paso anterior, y construir la solución restringida del VRPPD1-1. Todos los ejes de la solución restringida se agregan al conjunto $\mathcal{I}$ de puntos de inserción.

11. Guardar las secuencias básicas extraídas en un pool de secuencias $\mathcal{S}_P$, junto con todas las secuencias unitarias.

12. Construir un pool de variables $\mathcal{V}_P$ tal que para cada $i \in \mathcal{I}$, se agrega a $\mathcal{V}_P$ el $\eta\,\%$ de variables x_{si} con menor costo de inserción γ_{si}, donde $s \in \mathcal{S}_P \cap \mathcal{S}_i$. Además, se agregan a $\mathcal{V}_P$ todas las variables x_{si} tal que s es una secuencia básica, con i el punto de inserción pivote para s.

13. Construir la relajación PL del RRM con $\mathcal{V}_P$ como conjunto de variables iniciales.

14. Mientras $C(RRM)$ mejora, y $\#\text{iters} \leq \tau$,

15. Agregar al RRM un conjunto de variables prometedoras, generadas utilizando técnicas de pricing, considerando una reducción para el vecindario $N(z_0, \mathcal{F})$.

16. Resolver $C(RRM)$

17. Agregar restricciones de integralidad al RRM, y resolver a través de un paquete de propósito general.

18. Si se encontró una mejor solución, resetear Θ. En caso de que se agoten los criterios en Θ, terminar.

En las siguientes secciones se introducen detalles sobre cada una de las fases de este algoritmo, alternativas consideradas, dificultades y adaptaciones respecto del esquema original exhibido en el capítulo anterior.

5.2. Heurística inicial

Encontrar una solución factible para la versión del VRPPD1-1 introducida en el Capítulo 3, es un problema en la clase de complejidad $\mathcal{P}$, que puede ser resuelto muy fácilmente de diversas formas.

La Figura 5.2 muestra dos soluciones factibles para una instancia relativamente chica con 16 clientes. Ambas soluciones utilizan una única ruta (o único vehículo). La solución de la Figura 5.2(a) se obtiene recorriendo de manera creciente, en primer lugar todos los clientes de recolección, y luego todos los clientes entrega, garantizando así que p_i y d_i están en la misma ruta, y que además siempre p_i precede a d_i. Por otro lado, la solución de la Figura 5.2(b) se construye visitando los clientes en orden creciente, con una variante en donde para asegurar factibilidad, cada vez que se visita un vértice de recolección, inmediatamente después se visita el vértice entrega asociado.

Si bien es fácil en cualquiera de los procedimientos asegurar la factibilidad, la calidad de esta clase de soluciones puede ser (y en la práctica probablemente lo sea) muy pobre. Es por esto que se decide reforzar el esquema utilizando las ideas propuestas por Nanry et al. [47] para el VRPTW. Se utiliza inicialmente un procedimiento basado en el paradigma goloso, insertando pares de clientes, y luego se pasa a la fase de refinamiento a través de operadores de búsqueda local para mejorar la calidad de la solución inicial obtenida.

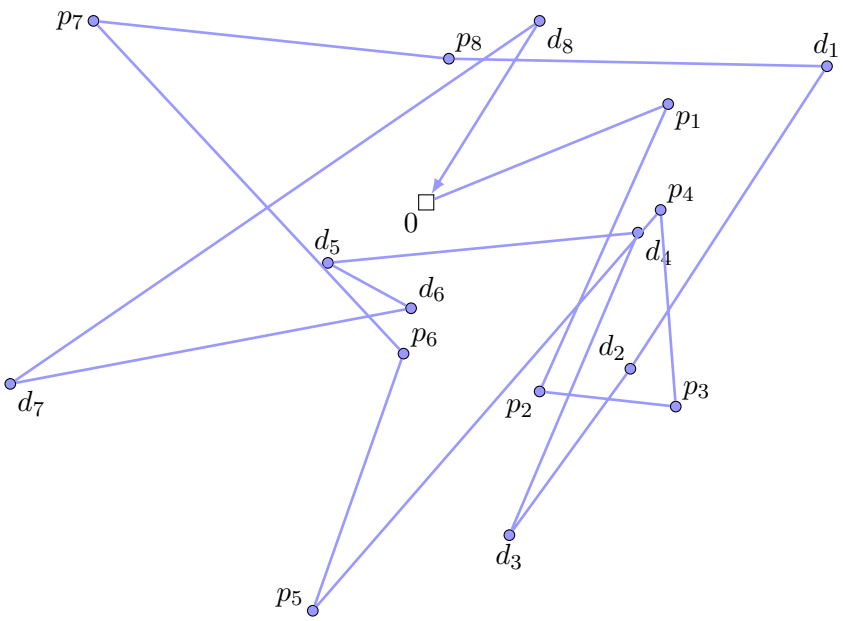
5.2.1. Solución Inicial Factible

Para construir la solución inicial se trabaja con las rutas parciales asociadas a pares de clientes relacionados, es decir, dado un par (p_i, d_i), la ruta parcial asociada se define como $r_i := (0, p_i)(p_i, d_i)(d_i, v)$. La idea es construir un conjunto inicial de este tipo de rutas, que son factibles respecto de la precedencia, eligiendo las de menor costo, para luego extenderlas utilizando un criterio goloso en el costo en extensión.

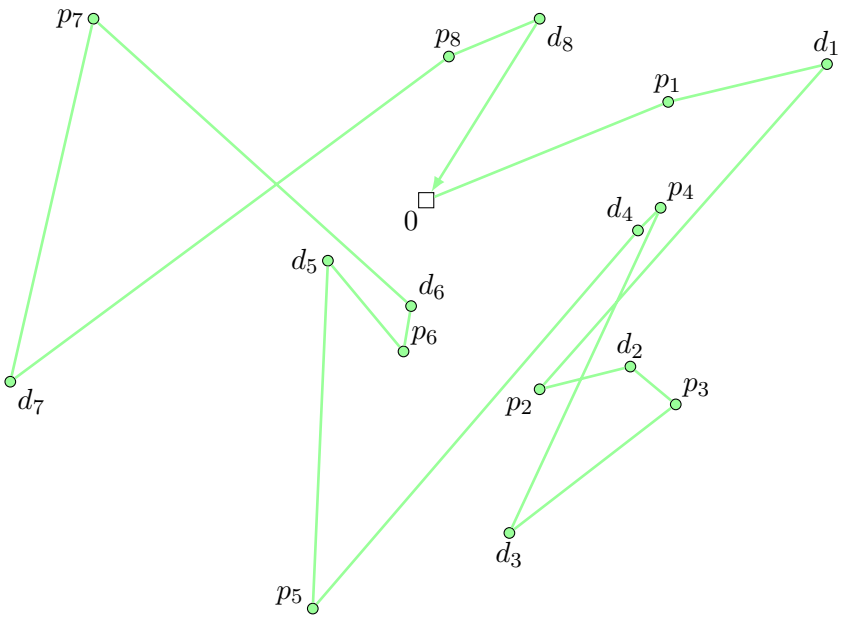
Formalmente, dada una ruta factible $r = (0, v_1) \dots (v_h, v)$, donde v denota el depósito, es posible extender r con (p_i, d_i) para construir $r' = (0, v_1) \dots (v_h, p_i)(p_i, d_i)(d_i, v)$.
Se define también el costo de una ruta r como $c(r) := \sum_{(i,j)\in r} c_{ij}$.

Con esta notación, se propone un algoritmo para construir una solución factible inicial basado en la idea de elegir inicialmente las k rutas r_i de menor costo $c(r_i)$, y fijarlas como *rutas parciales* del problema. Luego, se extienden dichas rutas parciales con los pares (p_j, d_j) restantes, i.e. que no pertenecen a ninguna ruta parcial, insertando cada uno al final de la ruta donde genere menor costo de inserción. Se finaliza con k rutas en donde cada cliente pertenece a exactamente una ruta, y éstas satisfacen las precedencias.

El ejemplo de la Figura 5.3 muestra el resultado de aplicar el Algoritmo 5.2 sobre la instancia utilizada en la Figura 5.2.



(a) Costo total: 72,321840



(b) Costo total: 56,624046

Figura 5.2.: Soluciones factibles para una instancia euclideana del VRPPD1-1 donde 0 denota el depósito, $k = 4$, los nodos de pick-up son $\{p_1, \ldots, p_8\}$, los nodos delivery son $\{d_1, \ldots, d_8\}$, y p_i debe preceder a d_i.

Algoritmo 5.2 Algoritmo para construir una solución factible inicial

Entrada: $V = P \cup D$, $k \geq 1$
Salida: Solución factible de k rutas para el VRPPD1-1

1. *Inicialización:*
   2. Para cada par (p_i, d_i),
      3. Calcular el costo de la ruta parcial r_i.
   4. Elegir las k rutas de menor costo y fijarlas como *rutas parciales*.
5. *Inserción:*
   6. Para cada par $e_i =: (p_i, d_i)$ que no pertence a una *ruta parcial*,
      7. Para cada *ruta parcial* r,
         8. calcular el costo de extender r con e_i.
      9. Utilizar e_i para extender la ruta r^* donde se genera menor costo de extensión, entendido este como $c(r') - c(r^*)$, donde r' es la ruta extendida.

5.2.2. Operadores de búsqueda local

Inmediatamente después de que se construyó la solución inicial factible, se aplican operadores heurísticos para tratar de mejorar la calidad de la misma. A continuación se introduce el esquema general de refinamiento:

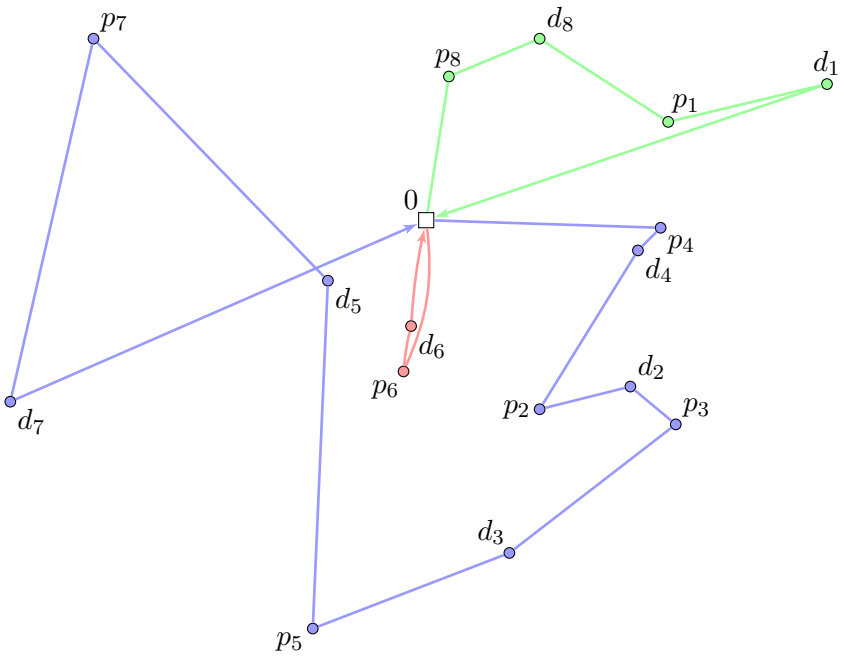
Algoritmo 5.3 Búsqueda Local con Operadores Heurísticos

Entrada: z_0 solución factible para el VRPPD1-1

1. Definir una lista Φ de operadores heurísticos.
2. Para cada $\phi \in \Phi$,
   3. Aplicar $\phi(z_0)$.
4. Si luego de aplicar todos los operadores en Φ, la solución cambió, repetir desde **2.**. Caso contrario, terminar.

La construcción de Φ determina el vecindario a explorar para encontrar mejores soluciones. Al finalizar el algoritmo 5.3, se obtiene una solución que es el óptimo local para el vecindario definido por la aplicación de Φ.

Basados en la propuesta de Nanry et al. [47], se proponen 3 adaptaciones de los operadores de búsqueda local: Single Paired Insertion (SPI), Swapping Between Routes (SBR), y Within Route Insertion (WRI). A continuación se presenta cada uno de ellos.



(a) Costo total: $50,613831$

Figura 5.3.: Solución obtenida con el Algoritmo 5.2 para la instancia de la Figura 5.2.

Single Paired Insertion (SPI)

El operador SPI consiste en buscar refinar una solución factible del VRPPD1-1 a través de la remoción e inserción de un par (p_i, d_i), desde una ruta hacia otra distinta, buscando realizar la inserción de la mejor forma posible, respetando siempre la precedencia de p_i sobre d_i.

Con esta idea, se explora el vecindario de todos los movimientos de remoción/inserción posibles sobre una solución factible $z \in \mathcal{Z}$, teniendo en cuenta el efecto en la calidad de la solución que produciría cada movimiento. Finalmente, si algún movimiento conduce a mejoras, se aplica el que resulte más conveniente en términos de la función objetivo.

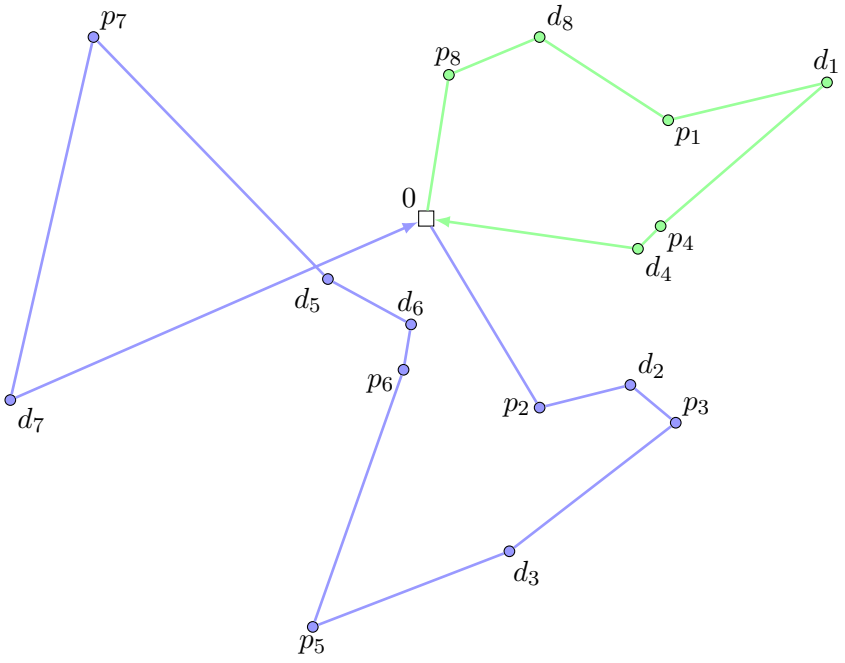
La complejidad de aplicar cada movimiento SPI es $\mathcal{O}((2n)^3) = \mathcal{O}(n^3)$, donde $2n = |P \cup D|$.

Algoritmo 5.4 SPI MOVE

Entrada: z solución factible para el VRPPD1-1

1. Para cada $p_i \in P$,
   2. Para cada ruta r que no contenga a p_i,
      3. Para cada par de posiciones l, m en r, $l \leq m$,
         4. Calcular el costo resultante de mover p_i y d_i desde la ruta actual, hasta r, insertando p_i en la posición l, y d_i en la posición m.
5. Si todos los movimientos considerados en **4.** empeoran la solución actual, terminar. Caso contrario, actualizar la solución actual aplicando el movimiento que resulte más conveniente para la función objetivo, y repetir desde **1.**

En la Figura 5.4 se puede apreciar el impacto de aplicar el operador SPI, sobre la instancia presentada en la Figura 5.3, observando una mejora del 11,26 % aproximadamente.



(a) Costo total: 44,913485

Figura 5.4.: Solución de aplicar el Algoritmo SPI sobre la solución inicial de la Figura 5.3.

Swapping pairs Between Routes (SBR)

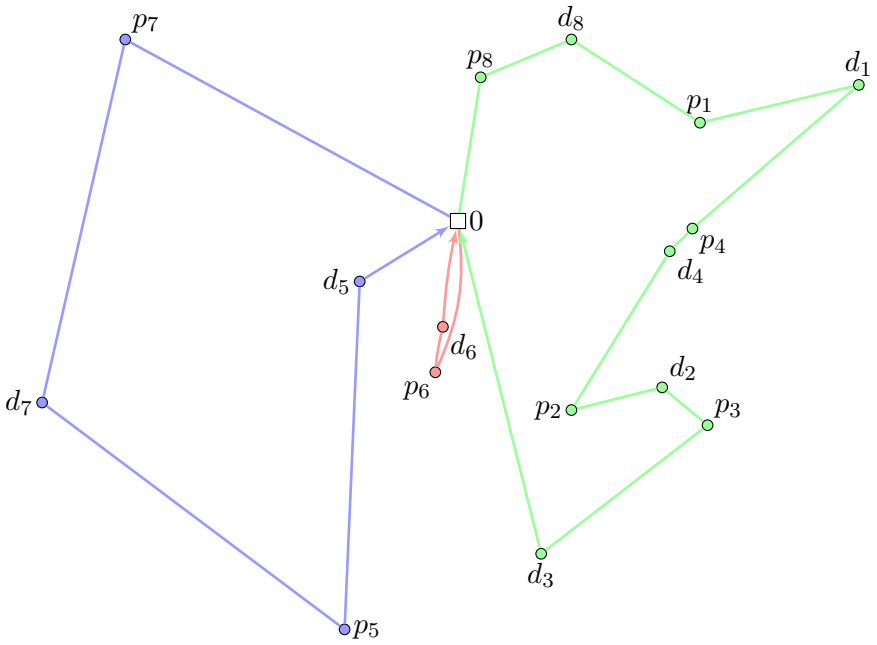
A diferencia de SPI, el operador SBR explora el vecindario definido por intercambios (*swaps*) entre pares de clientes de recolección y entrega relacionados, alojados en rutas distintas, donde además se tiene en cuenta la mejor forma de re-ubicar los pares en cuestión, dentro de cada

una de las rutas. En el algoritmo 5.5 se formaliza esta idea, donde la complejidad de aplicar un movimiento, i.e., un intercambio que mejore la calidad de la solución, es $\mathcal{O}(n^4)$ donde $2n = |P \cup D|$.

Algoritmo 5.5 SBR MOVE

Entrada: z solución factible para el VRPPD1-1

1. Para cada par (p_i, d_i) y (p_j, d_j) en rutas distintas r_i y r_j respectivamente,
   2. Calcular el costo resultante de intercambiar las rutas de (p_i, d_i) y (p_j, d_j), ubicando de la mejor forma posible, p_i y d_i en la ruta r_j, mientras que p_j y d_j se ubican en la ruta r_i de la mejor forma posible también, siempre satisfaciendo la restricción de precedencia.
3. Si todos los movimientos considerados en **2.** empeoran la solución, terminar. Caso contrario, aplicar el movimiento que resulte más conveniente para la función objetivo, y repetir desde **1.**



(a) Costo total: 47,653777

Figura 5.5.: Solución de aplicar el Algoritmo SBR 5.5 sobre la solución inicial de la Figura 5.3.

De manera análoga al ejemplo presentado para el caso anterior, en la Figura 5.5 se muestra el resultado de aplicar el operador SBR, sobre la instancia presentada en la Figura 5.3, observando una mejora aproximada del 5,8 %, menor en este caso particular que la conseguida por el movimiento SPI.

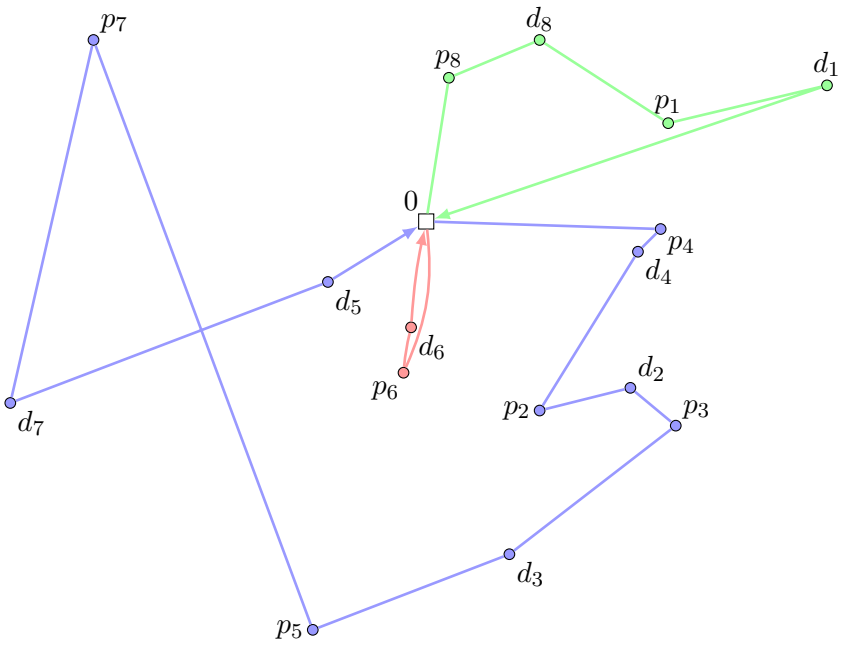
Within Route Insertion (WRI)

El operador WRI es un operador de refinamiento individual de rutas. Para cada ruta se exploran todos los deplazamientos posibles que satisfacen la restricción de precedencia entre clientes relacionados. La complejidad de aplicar un movimiento es $\mathcal{O}(n^3)$ donde $2n = |P \cup D|$ y al igual que SBR.

Algoritmo 5.6 WRI MOVE

Entrada: z solución factible para el VRPPD1-1

1. Para cada ruta r,
   2. Para cada par $(p_i, d_i) \in \tilde{r}$,
      3. Para cada par de posiciones l, m en $\tilde{r}$ tal que $l \leq m$,
         4. Calcular el costo de remover p_i y d_i para insertarlos en la posición l y m respectivamente.
5. Si todos los movimientos considerados en **4.** empeoran la solución, terminar. Caso contrario, aplicar el movimiento que resulte más conveniente para la función objetivo, y repetir desde **2.**



(a) Costo total: 49,895851

Figura 5.6.: Solución de aplicar el Algoritmo WRI 5.6 sobre la solución inicial de la Figura 5.3.

Al aplicar este operador para refinar la solución exhibida por la Figura 5.3, se consigue una mejora mucho menor que aquella obtenida por los operadores SPI y WRI, siendo apenas del 1,4 % aproximadamente.

5.3. Criterios de selección de nodos

Al comienzo del capítulo se señalan los posibles escenarios que se pueden presentar al remover nodos, debido a la relación de precedencia. Para evitar manejar el proceso posterior de inserción se consideran 3 criterios:

1. RA, aplicado únicamente sobre nodos de recolección, con la salvedad de que si p_i es removido entonces también se remueve el nodo de recolección d_i asociado.
2. SCT(p), con la misma consideración que el criterio anterior.
3. NGH, utilizando como semilla cada uno de los nodos en $V = P \cup D$, removiendo siempre de a pares asociados.

La decisión de remover simultáneamente los pares de clientes asociados está fundamentada en el hecho de que al remover un conjunto $\mathcal{F}$ de nodos, el vecindario de soluciones contiene al menos todas las soluciones que se pueden obtener removiendo cualquier subconjunto de $\mathcal{F}$. Luego, el vecindario definido por los esquemas de extracción de nodos que admitan la remoción de un vértice sin su par asociado, está siempre contenido en el vecindario donde todas las extracciones se efectúan de a pares.

Observación: $N(z_0, \mathcal{G}) \subseteq N(z_0, \mathcal{F})$ para todo $\mathcal{G} \subseteq \mathcal{F}$ conjunto de nodos extraídos de una solución factible z_0 para el VRPPD1-1.

Cabe señalar que la extracción simultánea es una decisión muy importante en el modelado de las restricciones de precedencia en el RM, como se verá más adelante, donde se asume fuertemente que $p_i \in P \cap \mathcal{F}$ si y sólo si $d_i \in D \cap \mathcal{F}$.

5.4. Una formulación del RM para el VRPPD 1-1

Al trabajar con la formulación del RM introducida en el capítulo anterior, dado que los clientes son discriminados por clientes de recolección y clientes de entrega, si se denota $V(s)$ al conjunto de nodos de una secuencia $s \in \mathcal{S}(\mathcal{F})$, pueden distinguirse los siguientes casos:

1. $V(s)$ contiene clientes de recolección y clientes de entrega, con al menos un par recolección-entrega asociado.
2. $V(s)$ contiene únicamente clientes de recolección (ó clientes de entrega)
3. $V(s)$ contiene clientes de recolección y entrega, pero no pares asociados (i.e., nunca se cumple $p_i, d_i \in V(s)$ simultáneamente para algún $i = 1, \ldots, n$).

Para analizar cada uno de estos casos, se introduce la siguiente nomenclatura: dada una secuencia s de nodos en $\mathcal{F}$ y un nodo $p_i \in V(s)$, se dice que p_i es *recolección exclusiva* si $d_i \notin V(s)$, i.e., d_i (la entrega asociada a p_i) no está en s. Análogamente, se definen también los nodos *entrega exclusiva* dentro de una secuencia s. Se dice que s es *recolección-exclusiva* si $V(s)$ contiene únicamente nodos de recolección, mientras que es *entrega-exclusiva* si solo contiene nodos de entrega. Si contiene nodos de recolecciones y entregas exclusivas, se dice que es *exclusiva*. Una secuencia será *inclusiva* si sólo contiene pares de nodos asociados.

En el primer caso, donde existen un nodo recolección p_i y el nodo entrega d_i asociado, se debe cumplir que p_i preceda a d_i. Una secuencia s que contenga una entrega d_i, donde la recolección p_i asociada esté después que d_i, es infactible independientemente del punto de inserción con el que se considere asignarla, y por lo tanto utilizar la variable x_{si} correspondiente permite generar soluciones infactibles para el problema. Se re-define $\mathcal{S}$ entonces como el conjunto de caminos en $\mathcal{F}$ que respetan precedencias. Observar que la re-definición de $\mathcal{S}$ no obliga a que las secuencias alojen únicamente pares de recolección y entrega simultáneamente, sino que se admite la posibilidad de que las secuencias alojen recolecciones sin sus respectivas entregas, así como también que alojen entregas sin sus recolecciones asociadas.

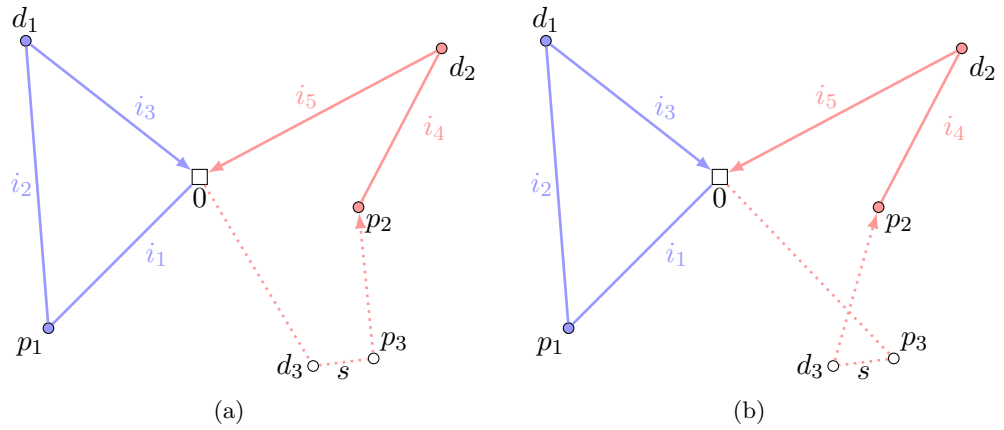


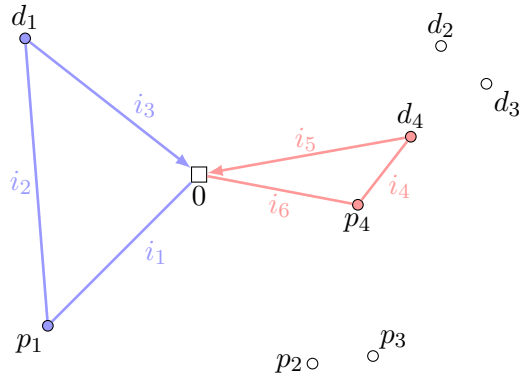
Figura 5.7.: Ejemplos de re-ubicación (uno factible y otro infactible) para la secuencia s en el punto de inserción $(v, p_2) \in \mathcal{I}$, donde las orientaciones de s son indistinguibles.

Por otro lado, a diferencia del esquema aplicado al VRP, el hecho de que para una secuencia $s \in \mathcal{S}$ exista un par $\{p_i, d_i\} \subseteq V(s)$ determina la orientación en la asignación de la secuencia en un punto de inserción. Se considera el ejemplo 5.7, en el cual se muestra una solución restringida para una instancia de VRPPD1-1 donde la precedencia exige que p_i preceda a d_i (para $i \in \{1, 2, 3\}$). Asignar s al punto de inserción $(0, p_2)$ como muestra la Figura 5.7(a) genera una ruta infactible, dado que el nodo de recolección p_3 es visitado después que el nodo de entrega d_3. En cambio, la asignación puede realizarse como se muestra en la Figura 5.7(b), donde no se generan conflictos de precedencia.

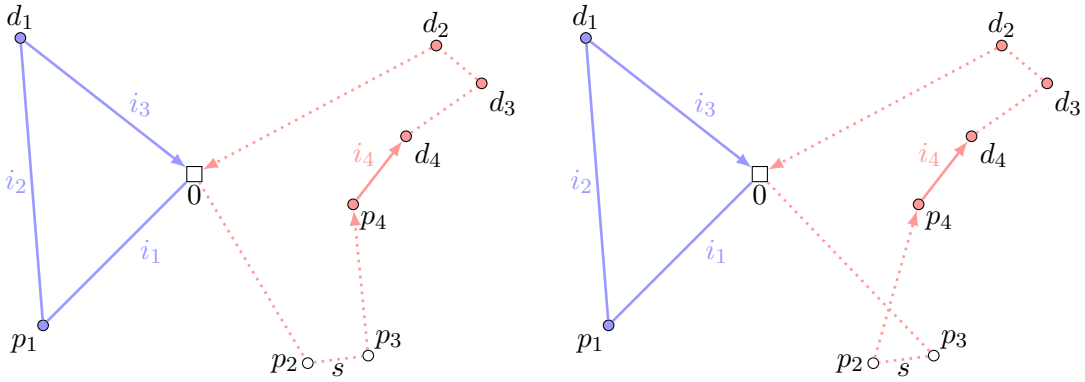
Sin embargo, para secuencias del tipo 2, a las que se llaman recolección-exclusivas (o entrega-exclusivas), la situación es diferente. En este caso es nuevamente posible considerar ambas orientaciones al asignar la secuencia a un punto de inserción. En el ejemplo de la Figura 5.8 se muestra esta situación para una secuencia $s = (p_2, p_3)$ recolección-exclusiva. De la misma forma, para las secuencias de tipo 3, i.e. aquellas secuencias que sean exclusivas en general, también es posible considerar ambas orientaciones al asignarlas.

Como consecuencia de estas observaciones, es necesario adaptar la definición de γ_{si}, que denota el costo de inserción de la secuencia $s \in \mathcal{S}$ en el punto de inserción $i \in \mathcal{I}$.

Una alternativa es definir γ_{si} según si s es exclusiva o no, y mantener un esquema parecido al utilizado para γ_{si} en el RM de capítulo anterior. Sin embargo, se decide considerar todas



(a) Solución restringida $z_0(\mathcal{F})$ con $\mathcal{F} = \{p_2, p_3, d_2, d_3\}$



(b) (c)

Figura 5.8.: Ejemplos de re-ubicaciones factibles para la secuencia s en el punto de inserción $(0, p_4) \in \mathcal{I}$, donde las orientaciones de s son indistinguibles.

las secuencias con una única orientación, y de esta manera, en el ejemplo de la Figura 5.8, las dos orientaciones de s son consideradas secuencias distintas, i.e. $s_1 = (p_2, p_3)$ es distinta de la secuencia $s_2 = (p_3, p_2)$. Si bien esto implica aumentar el número de variables consideradas, el impacto en la práctica es menor dado que la construcción de variables se resuelve heurísticamente con un esquema de generación de columnas. En consecuencia, para $s = (v_1, \ldots, v_h) \in \mathcal{S}$ y $i = (a, b)$ se re-define:

$$\gamma_{si} := c(s) - c_{ab} + c_{av_1} + c_{v_h b}$$

Por otro lado, así como en el caso del DCVRP se reflejan en el modelo las restricciones de distancia y capacidad agregando las Ecuaciones (4.5) y (4.6) a la formulación, se propone reflejar en el RM las restricciones de precedencia 1-1.

Se observa que las secuencias $s \in \mathcal{S}$ tal que $V(s)$ contiene nodos de entrega (o recolección) exclusiva, son las que generan incompatibilidades al re-ubicar, mientras que secuencias inclusivas podrán ser asignadas en cualquier punto de inserción.

En el ejemplo 5.8(a), la secuencia inclusiva (p_2, d_2) puede ser re-ubicada en cualquier punto de inserción $i \in \{i_1, \ldots, i_6\}$. En cambio, la secuencia recolección-exclusiva (p_2, p_3) debería ser asignada con cuidado, ya que si d_2 o d_3 fuesen asignados dentro de una secuencia al punto de inserción i_6 o i_4, entonces (p_2, p_3) no podría asignarse a i_5, ya que estaría generando incompatibilidades con los nodos d_2 y d_3 que quedarían posicionados antes que sus correspondientes recolecciones. Más aún, ninguna secuencia $s \in \mathcal{S}$ que contenga nodos de recolección exclusiva podría ser asignada a un punto de inserción de las características de i_5, que es el último en su ruta, puesto que no habría forma de que dicho nodo de recolección exclusiva preceda a su entrega asociada. Esta última particularidad será explotada en detalle en la Sección 5.6.1.

Cuando se consideran pares de secuencias con nodos asociados, (i.e., $s_1, s_2 \in \mathcal{S}$ tal que $p_i \in V(s_1)$ y $d_i \in V(s_2)$) también se debe tener cuidado respecto de la asignación. Desde ya, asignar s_1 y s_2 a puntos de inserción en rutas distintas de la solución restringida, siempre genera soluciones infactibles, dado que p_i y d_i estarían en rutas distintas. Por ejemplo, en la Figura 5.8(a) si la secuencia (p_2, p_3) es asignada al punto de inserción i_2, entonces no es factible asignar (d_2, d_3) al punto de inserción i_5, y tampoco a i_4 ó i_6 por pertenecer a una ruta distinta de la de i_2.

Se propone entonces, en el marco del VRPPD1-1, agregar al RM las siguientes 2 nuevas restricciones:

- *Factibilidad inter-ruta:*

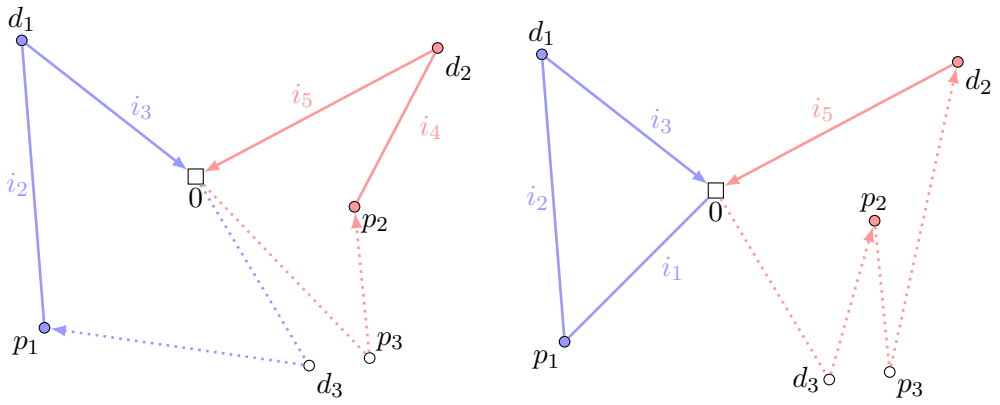
$$x_{s_1,i} + x_{s_2,j} \leq 1 \qquad \begin{array}{r} \exists\, v_k \in P \cap \mathcal{F},\ s_1 \in \mathcal{S}(v_k),\ s_2 \in \mathcal{S}(v_{n+k}) \\ i \in \mathcal{I}(r_1),\ j \in \mathcal{I}(r_2),\ r_1 \neq r_2,\ r_1, r_2 \in \mathcal{R} \end{array} \tag{5.1}$$

  Dadas dos rutas r_1 y r_2 ($r_1 \neq r_2$), se quiere evitar que pares asociados de recolección y entrega sean alojados por rutas distintas. Es decir, si $p_i \in P \cap \mathcal{F}$ y $d_i \in D \cap \mathcal{F}$, se quiere que las secuencias s_1 que contienen a p_i y las secuencias s_2 que contengan a d_i no puedan ser alojadas simultáneamente por puntos de inserción de distintas rutas. En el ejemplo de la figuta 5.9(a), las secuencias (p_3) (recolección-exclusiva) y (d_3) (entrega-exclusiva) son asignadas a puntos de inserción ($(0, p_1)$ y $(0, p_2)$ respectivamente) ubicados en distintas rutas de la solución restringida.

- *Factibilidad intra-ruta:*

$$x_{s_2,i} + x_{s_1,j} \leq 1 \quad \begin{array}{r} \exists\, v_k \in P \cap \mathcal{F},\ s_1 \in \mathcal{S}(v_k),\ s_2 \in \mathcal{S}(v_{n+k}) \\ i < j,\ i, j \in \mathcal{I}(r),\ r \in \mathcal{R} \end{array} \tag{5.2}$$

  Dentro de una misma ruta r, se quiere que los pares asociados de entrega y recolección respeten la precedencia (i.e., $p_i \in P \cap \mathcal{F}$ esté antes que $d_i \in D \cap \mathcal{F}$ en la ruta r). Por lo tanto, las secuencias s_1 que contienen a p_i y las secuencias s_2 que contengan a d_i no deben poder asociarse simultáneamente a un punto de inserción j e i (ambos de r) respectivamente, donde i está antes que j. La situación de infactibilidad para este caso se representa en la Figura 5.9(b), donde si bien (p_3) y (d_3) son asignadas dentro de la misma ruta en la solución restringida, no respetan la precedencia de sus nodos exclusivos.



(a) Infactibilidad inter-ruta (b) Infactibilidad intra-ruta

Figura 5.9.: Ejemplos de infactibilidades inter-ruta e intra-ruta al asignar las secuencias (p_3) y (d_3) con nodos exclusivos asociados.

5.5. Sobre la reducción del vecindario

Con el objetivo de reducir el vecindario $N(z_0, \mathcal{F})$, se analiza un criterio heurístico para tratar de no considerar a priori las secuencias con un costo de inserción alto y de esta manera reducir al mismo tiempo el esfuerzo computacional requerido para explorarlo.

Como se menciona en (4.7), de manera análoga se construyen $\mathcal{F}_i$, y $\mathcal{S}_i$ teniendo en cuenta ahora la re-definición de $\mathcal{S}$ que involucra las precedencias, utilizando esta relación para decidir si un par de nodos p_k, d_k deben ser considerados para un punto de inserción i dado. La construcción de $\mathcal{F}_i$ presentada en la ecuación (4.14) se adapta de forma de que si un nodo de recolección satisface el criterio heurístico para ser considerado en el punto de inserción i, entonces el nodo de recolección asociado también es considerado en i, sin importar el costo de inserción. Se pretende lo mismo en caso de que sea el nodo de recolección el que satisfaga el criterio.

Formalmente, $\mathcal{F}_i$ se construye como:

$$\mathcal{F}_i := \{p_k, d_k \in \mathcal{F} : \gamma_{p_k i} \leq \lambda\, \delta_{p_k r_i} \vee \gamma_{d_k i} \leq \lambda\, \delta_{d_k r_i}\} \cup \{v \in \mathcal{F} : i = i_v\}$$

De esta manera, si se denota con $\mathcal{S}_i(v)$ al conjunto de secuencias factibles para el punto de inserción $i \in \mathcal{I}$ que contienen a $v \in \mathcal{F}$ (i.e. $\mathcal{S}_i(v) := \{s \in \mathcal{S} : V(s) \subseteq \mathcal{F}_i \wedge v \in V(s)\}$, la exploración del vecindario $N(z_0, \mathcal{F}, \lambda)$ se realiza a través de una versión adaptada del RRM introducido en (4.7), pero con las siguientes restricciones adicionales:

$$x_{s_1,i} + x_{s_2,j} \leq 1 \quad \exists\, v_k \in P \cap \mathcal{F},\ s_1 \in \mathcal{S}_i(v_k),\ s_2 \in \mathcal{S}_j(v_{n+k}) \\ i \in \mathcal{I}(r_1),\ j \in \mathcal{I}(r_2),\ r_1 \neq r_2,\ r_1, r_2 \in \mathcal{R} \tag{5.3}$$

$$x_{s_2,i} + x_{s_1,j} \leq 1 \quad \exists\, v_k \in P \cap \mathcal{F},\ s_1 \in \mathcal{S}_j(v_k),\ s_2 \in \mathcal{S}_i(v_{n+k}) \\ i < j,\ i, j \in \mathcal{I}(r),\ r \in \mathcal{R} \tag{5.4}$$

5.6. Esquemas de Pricing

Para resolver el RM, se inicializa conservadoramente con un conjuto de variables chico que asegure factibilidad, y se resuelve luego la relajación lineal para agregar más variables usando técnicas de pricing, evitando sobrecargar el modelo con un número excesivo de éstas, teniendo en cuenta las restricciones de tiempo computacional.

Para generar variables prometedoras es conveniente explotar características particulares de VRPPD1-1. A continuación se detallan adaptaciones basadas en los esquemas de pricing de De Franceschi et al. [18] y Toth-Tramontani [55], y se propone un nuevo algoritmo de pricing basado en estas ideas.

Se recuerda que el esquema original de De Franceschi et al. [18] consta de dos fases: (1) la fase de inicialización, donde se agregan secuencias básicas para garantizar la factibilidad del modelo; (2) la fase denominada ciclo de pricing, donde para cada punto de inserción $i \in \mathcal{I}$ se considera la información dual de la relajación lineal del modelo construido hasta el momento y se generan secuencias $s \in \mathcal{S}$ para agregar las variables x_{si} en caso de que sea conveniente.

La adaptación básica sobre este esquema se produce en la segunda fase: la generación de secuencias debe tener en cuenta las restricciones de precedencia, i.e. toda secuencia que contenga pares de clientes de recolección-entrega asociados debe verificar que el nodo de recolección preceda al nodo de entrega. Esta es en realidad una adaptación obligatoria luego de la re-definición $\mathcal{S}$ introducida en las secciones anteriores. Observar una vez más que podría suceder que la secuencia tenga nodos exclusivos de recolección o entrega, como se describe en el inicio de este capítulo.

Por otro lado, en la formulación del RM (o incluso RRM) para el DCVRP presentada en (4.1), todas las restricciones del problema están presentes desde que se inicializa el modelo. Existe la posibilidad, en el caso del RRM, de que al aplicar la reducción de vecindario y construir $\mathcal{F}_i$, éste sea tal que $\mathcal{F}_i = \emptyset$ para algún i y que entonces alguna restricción del estilo $\sum_{s \in \mathcal{S}_i} x_{si} \leq 1$ (ver 4.9) no esté presente inicialmente. Esta situación es manejable agregando para cada $i \in \mathcal{I}$ al menos una secuencia s al conjunto $\mathcal{S}_i$. De esta manera, todas las restricciones están fijas y la información de las variables duales en C(RRM) es completa, haciendo posible computar los costos reducidos de forma precisa.

A diferencia de esto, en la formulación del RM para el VRPPD1-1, las restricciones de precedencias presentadas en (5.4) no están todas presentes al momento de inicializar el modelo,

y no es posible manejarlas de forma similar al caso en que $\mathcal{F}_i = \emptyset$. Lo que se tiene es una situación particular que se suele dar en el contexto de formulaciones PLE con una cantidad exponencial de variables, en donde tanto filas como columnas deben ser generadas sobre demanda.

Como consecuencia, una solución primal podría no ser factible debido a incompatibilidades entre secuencias asociadas a distintas variables que son utilizadas en la solución primal, cuyas restricciones de factibilidad (inter-ruta o intra-ruta) aún no se han agregado al modelo.

Sin embargo, esta situación puede ser bien manejada: cada vez que se desea agregar una variable x_{si} al modelo, se deben considerar todas las incompatibilidades dadas por las restricciones de precedencia (5.4), entre las variables presentes en el modelo y la variable x_{si}, agregando junto con esta última, también, todas las restricciones de precedencia que sean necesarias.

Con este esquema, la ausencia de algunas filas en el modelo incorpora dificultades al intentar calcular los costos reducidos asociados a las variables x_{si}, e impacta directamente en la generación de variables, en donde el cálculo de los costos reducidos no es preciso, o ni siquiera puede ser computado correctamente, debido a que no se tiene información dual asociada a las restricciones que no están presentes.

Esta situación en donde el número de filas del modelo PLE (con un conjunto reducido de variables) depende del número de columnas, se conoce como problema CDR (siglas de *Column-Dependent-Rows*).

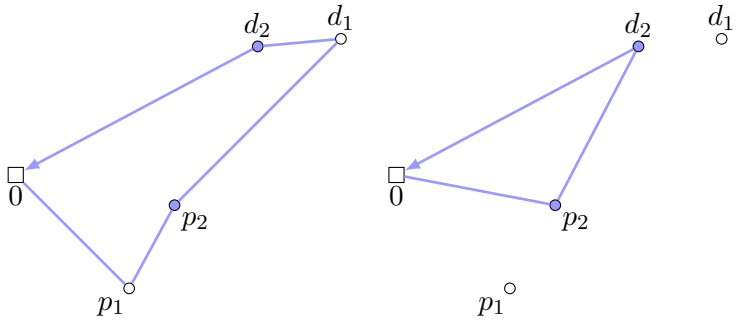
La literatura sobre los problemas CDR es limitada [61, 4, 60, 21]. Para el *Cutting Stock Problem* (CSP), Zak [61] resuelve el problema CDR heurísticamente, restringiendo el sub-problema de Generación de Columnas, consiguiendo soluciones sub-óptimas para el CSP y terminando prematuramente el algoritmo de pricing. Avella et al. [4] utiliza una formulación set-packing para el *Time-Constrained Routing Problem* (TCRP) y propone una heurística basada en la generación de columnas y filas simultáneamente. Define un criterio de parada para el problema de GC utilizando la información dual de las restricciones que sí están presentes en el modelo. Sin embargo, Muter et al. [45] demuestran que dicho criterio de parada no estima apropiadamente las variables duales de las restricciones ausentes conduciendo a soluciones sub-óptimas. Finalmente, nuevamente Muter et al. [44], propone un esquema de propósito general para diseñar algoritmos de generación simultánea de colunas y filas para problemas CDR.

Frente a esta situación, una opción es trabajar con un enfoque de generación simultánea de columnas y filas, con filas dependendientes de las Columnas (ver *Simultaneous Column-and-Row Generation with Column-Dependent Rows* [44]). Sin embargo, se descarta esta opción dada, una vez más, la restricción de tiempo computacional que implica utilizar el esquema propuesto por Muter et al.[44] en este contexto, y la prioridad de encontrar la mayor cantidad de variables prometedoras con menor costo reducido posible.

Una alternativa sería estudiar cómo varía la relajación C(RRM) cada vez que se agrega una nueva columna y las respectivas filas, tomando alguna decisión en base a esto para determinar con algún criterio si la variable debe ser efectivamente añadida al modelo. Nuevamente, este esquema podría ser costoso en el contexto ya mencionado, ya que implicaría resolver la relajación PL para cada potencial variable a agregar al modelo.

Otra alternativa es considerar una nueva reducción del vecindario. Recordar que se asume que siempre son removidos pares p_i, d_i ($p_i \in P$, $d_i \in D$) de la solución z_0 (ver Sección 5.3). Si además se asume que, si $p_i, d_i \in \mathcal{F}$ entonces deben estar en la misma secuencia $s \in \mathcal{S}$, es posible quitar las restricciones de precedencia del modelo de reubicación, y luego todas las restricciones estarían presentes originalmente en el modelo.

Bajo esta última alternativa, se considera la situación en el Ejemplo 5.10. Se selecciona el conjunto de nodos $\mathcal{F} = \{p_1, d_1\}$, y se extraen de la solución de la Figura 5.10(a) para construir la solución restringida $z_0(\mathcal{F})$ como se muestra en la Figura 5.10(b).



(a) Solución original z_0 (b) Solución restringida $z_0(\mathcal{F})$

Figura 5.10.: Ejemplo donde no es posible asegurar que la solución del RRM (o incluso RM) sea al menos tan buena como la solución original z_0, asumiendo que para secuencia $s \in \mathcal{S}$, se tiene que $p_i \in V(s)$ si y solo si $d_i \in V(s)$.

Se tiene entonces que toda secuencia $s \in \mathcal{S}$ deberá contener a p_1 y d_1 simultáneamente, donde si bien siempre existe una solución factible para el modelo de reubicación, no es posible asegurar que sea de mejor calidad que z_0 (incluso podría ser peor).

Se considera finalmente trabajar con una aproximación de los costos reducidos, utilizando la información dual que sí está disponible, calculando el costo reducido rc_{si}, asociado a la variable x_{si}, heurísticamente.

Formalmente, se computa una aproximaxión denotada como $\hat{rc}_{si}$ para rc_{si} dada por:

$$\hat{rc}_{si} := \gamma_{si} - \sum_{v \in V(s)} \tilde{\pi}_v^1 - \tilde{\pi}_i^2 \tag{5.5}$$

Se asume $\hat{rc}_{si} \approx rc_{si}$, aprovechando la información de las restricciones que están fijas en el modelo independientemente del conjunto de variables iniciales, donde las variables duales optimas $\tilde{\pi}_v^1$ y $\tilde{\pi}_i^2$ corresponden a las restricciones (4.8) y (4.9) respectivamente.

Con las adaptaciones mencionadas, el algoritmo resulta:

Algoritmo 5.7 LSA-VRPPD 1-1: Ciclo de Pricing

Entrada: i: punto de inserción, $\hat{rc}^*$: el costo reducido más chico generado hasta el momento, $\tilde{\pi}$: el óptimo dual de la relajación lineal, δ, N_{min}, N_{max}, L_{max}, RC_{max}.

1. Inicializar: $L := 0$, $S := \{<>\}$
2. $L := L + 1$
3. Para cada $s \in S$,
   4. Para cada $v \in \mathcal{F}$ tal que $v \notin s$,
      5. Generar todas las secuencias obtenidas de insertar, en posiciones que no violen precedencias, el nodo v en s
6. Para cada s obtenida en **5.**,
   7. Considerar la variable x_{si} y evaluar la aproximación $\hat{rc}_{si}(\tilde{\pi})$ del costo reducido correspondiente.
   8. Si $\hat{rc}_{si}(\tilde{\pi}) < \hat{rc}^*$, actualizar $\hat{rc}^* := \hat{rc}_{si}(\tilde{\pi})$.
9. $S = \emptyset$
10. Insertar las N_{min} secuencias s con menor $\hat{rc}_{si}(\tilde{\pi})$, junto con las secuencias s tal que $\hat{rc}_{si}(\tilde{\pi}) \leq \text{máx}\{RC_{max}, \delta\hat{rc}^*\}$, insertando a lo sumo N_{max} secuencias
11. Si las secuencias insertadas en S tienen longitud menor que L_{max}, repetir desde **2.**. Caso contrario, considerar el siguiente punto de inserción.

Nuevamente, los valores L_{max}, N_{min}, N_{max}, RC_{max} y δ son parámetros a fijar experimentalmente. A diferencia del esquema original, en esta versión adaptada para el VRPPD1-1 no se considera agregar secuencias adicionales durante la segunda fase.

Se utilizan técnicas de hashing para evitar generar variables duplicadas, y en la práctica el algoritmo se aplica mientras mejore la relajación, junto con una cota máxima en la cantidad de aplicaciones, y una tolerancia de iteraciones máximas sin mejora.

5.6.1. Nuevos esquemas de Pricing

Si bien los esquemas de pricing introducidos en las secciones anteriores ya incluyen adaptaciones al contexto del VRPPD 1-1, se agregan nuevos esquemas basados en estas ideas, teniendo en cuenta las dificultades presentadas.

Tanto en el esquema de De Franceschi et al. [18] como en el de Toth-Tramontani [55], se utiliza un umbral denotado como RC_{max}, que es un parámetro fijado experimentalmente. Al momento de elegir una variable x_{si} para agregar al modelo, se utiliza una estimación para el costo reducido rc_{si} y un criterio de aceptación que depende entre otras cosas del RC_{max}, para decidir si la variable debe o no debe ser agregada al RM (ó RRM, según corresponda).

Ajustar este parámetro implica gran cantidad de experimentos sobre un conjunto de instancias definido, sobre todo cuando el algoritmo de pricing depende de muchos parámetros que pueden afectar el desempeño del valor de RC_{max}.

Se propone entonces utilizar un umbral dinámico, que se ajuste a la instancia del problema a medida que se agregan variables al modelo. A dicho umbral se lo denomina $dynamicRC_{max}$, y se aplica de forma tal que dada una variable x_{si} con costo reducido estimado por $\hat{rc}_{si}$, esta se agregue al modelo en caso de que $\hat{rc}_{si} \leq$ máx{ $\delta \hat{rc}^*$, $dynamicRC_{max}$}, donde al igual que en la versión original, δ es un parámetro fijado experimentalmetne, mientras que $\hat{rc}^*$ denota el costo reducido (aproximado) más chico encontrado.

Para definir $dynamicRC_{max}$ (DRC_{max}), una posibilidad es utilizar la información proporcionada por la estimación del costo reducido $\hat{rc}_{si}$ de aquellas variables que están presentes en el modelo, asumiendo que agregar una variable x_{si} a este, implica que x_{si} es de buena calidad, y que debe entonces influir en el valor de DRC_{max} para tratar de generar variables con costo reducido (estimado) similar a $\hat{rc}_{si}$.

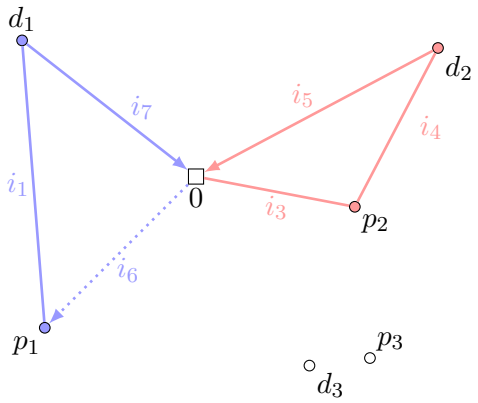
En particular, en la adaptación del esquema de De Franceschi et al., se inicializa $DRC_{max} = 0$ en cada iteración del ciclo de pricing, y se ajusta promediando el valor de $\hat{rc}_{si}$ de cada variable x_{si} que se agrega al modelo.

Esta última adaptación incrementa el impacto de los parámetros N_{min} y N_{max} utilizados en el esquema de De Franceschi et al. Se recuerda que dada una secuencia $s \in \mathcal{S}$ y un punto de inserción $i \in \mathcal{I}$, las N_{min} variables x_{si} con menor costo reducido siempre son agregadas al modelo (independientemente del criterio de aceptación), impactando fuertemente en el valor de $dynamicRC_{max}$. Luego, incrementar demasiado N_{min} podría producir ajustes sobre el umbral dinámico que no sean representativos de variables de buena calidad, y se estaría siendo muy flexible el criterio de aceptación. Es posible utilizar valores chicos de N_{min} donde pocas variables impacten en el valor de DRC_{max}, mientras que a un gran número de variables restantes (dependiendo del valor de N_{max}) se les permitiría ser evaluadas por el criterio de aceptación para eventualmente ingresar al modelo. Para determinar una buena configuración de N_{min} y N_{max} al utilizar DRC_{max} se realiza un análisis empírico cuyos resultados son discutidos en el siguiente capítulo.

Por otro lado, buscando explotar aún más la estructura del problema, durante las etapas de generación de secuencias es posible utilizar información sobre las variables ya generadas para determinar (heurísticamente) si es conveniente generar una nueva variable.

Se considera el ejemplo de la Figura 5.11. Se muestra la solución restringida $z_0(\mathcal{F})$ para una instancia de 6 clientes. El punto de inserción $i_6 \in \mathcal{I}$, en línea punteada, es el pivote de la secuencia básica (p_3, d_3). En el pool de variables $\mathcal{V}_P$ se tiene la variable $x_{(p_3,d_3)i_6}$ que permite reconstruir la solución original, junto con algunas variables generadas heurísticamente.

El ejemplo representa el estado del modelo de re-ubicación durante la fase de pricing, y se está evaluando si se debe agregar la variable $x_{(d_3),i_4}$ al modelo. Considerar la asignación de (d_3) a i_4 solo sería factible si p_3 fuese utilizada en algún punto de inserción ubicado en una posición anterior a i_4, dentro de la misma ruta. Como no se tiene en el modelo ninguna variable cuya secuencia contenga a p_4, y haga posible la asignación de esta a i_3, el ingreso de $x_{(d_3),i_4}$ puede postergarse o bien no realizase. Dado que para decidir si una variable ingresa a $\mathcal{V}_P$ se cuenta con un criterio de aceptación que depende del estado actual de la relajación PL, postergar su ingreso podría generar la situación de que, al momento de agregar la variable, esta ya no satisfaga el criterio de aceptación, lo cual (heurísticamente) denotaría que ya no es conveniente incluirla.



(a) Solución restringida $z_0(\mathcal{F})$

$$\mathcal{F} = \{p_3, d_3\}$$
$$\mathcal{I} = \{i_1, \ldots, i_7\}$$
$$\mathcal{V}_P = \{x_{(p_3,d_3)i_6}, x_{(p_3)i_5}, x_{(d_3)i_5}, x_{(d_3)i_4}\}$$

(b) Descripción por extensión de $\mathcal{I}$, $\mathcal{F}$ y $\mathcal{V}_P$

Figura 5.11.: Ejemplo donde no es conveniente generar la variable $x_{(d_3)i_4}$ (es infactible asignar (d_3) en i_4)

Otra opción en este escenario, es rechazar la inclusión de este tipo de variables, y eventualmente considerarla si vuelve a cumplir con el criterio de aceptación, y la asignación que representa genera una solución factible para el problema.

Por esta razón, en general, se decide no agregar al modelo variables x_{si} tal que s sea entrega-exclusiva, donde al menos una recolección asociada no fue contemplada aún dentro de la secuencia de alguna variable $x_{s'j}$ con j punto de inserción ubicado en una posición anterior a i, dentro de la misma ruta. La idea de este criterio es servir como filtro de variables durante la etapa de generación de secuencias para un punto de inserción dado, para evitar incluir en el modelo potenciales asignaciones infactibles.

6. Resultados computacionales

El objetivo de este capítulo es reportar la experiencia computacional de nuestro algoritmo, considerando diferentes parámetros y combinando las distintas propuestas analizadas en los capítulos anteriores. Para esto, se desarrolla un esquema de experimentación basado en 3 fases que se presenta en las siguientes secciones. Al evaluar empíricamente el desempeño del algoritmo que se propone, hay varias cuestiones que se deben tener en cuenta:

1. **Sobre el análisis de resultados:**
   Es necesario definir un algoritmo básico respecto al cual se pueda comparar las adaptaciones propuestas, analizar si tiene sentido aplicar el esquema, si consigue soluciones de *buena* calidad, etc.

   Para comparar los resultados se decide entonces diseñar un esquema de experimentación basado en fases: *Definición de configuración inicial*, *Análisis de alternativas para el algoritmo*, y *Comparación global*. En primer lugar se busca definir una versión básica del Algoritmo 5.1 propuesto para el VRPPD1-1. En la segunda etapa se analizan junto con este las adaptaciones y mejoras propuestas basadas en los esquemas originales de De Franceschi et at. [18] y Toth y Tramontani [55]. Finalmente, se realiza una comparación global del mejor esquema que implementa nuestra versión adaptada de las ideas de Toth y Tramontani [55], junto con la versión básica, y el mejor esquema que implementa la nueva reducción que se propone. Este esquema de experimentación se detalla en las siguientes secciones.

2. **Instancias del problema:**
   Dado que el VRPPD1-1 es una versión simplificada del VRPPD, no existen instancias benchmark. Por esta razón, se utilizaron adaptaciones de las siguientes instancias tomadas de la literatura:

   - VRPLIB: Se consideraron 15 instancias (donde $|V|$ varía entre 22 y 241), originalmente propuestas para el DCVRP [59].

   - CMT: Se tuvieron en cuenta 7 de las 14 instancias con costo real propuestas por Christofides y Eilon [12], y Christofides, Mingozzi y Toth [13], originalmente para el CVRP y DCVRP. Vale aclarar que el resto de las instancias CMT no consideradas son variaciones de aquellas que sí se utilizan, en parámetros que no están presentes en el VRPPD1-1, (e.g., tiempo máximo por ruta, tiempo máximo de carga, etc.).

3. **Entorno de pruebas:**
   Todos los experimentos se realizaron con un Intel Core i7, con 16Gb RAM, ejecutando en Ubuntu 12.04 LTS, utilizando C++ y el paquete de propósito general ILOG Cplex 12.4.

6.1. Esquema de experimentación

6.1.1. Definición de configuración inicial

El objetivo de esta etapa es establecer una versión básica del algoritmo que permita estudiar posteriormente el impacto de las mejoras propuestas. Para esto, se ajustó la configuración de parámetros del algoritmo buscando mejorar su rendimiento. Dada la gran cantidad de parámetros presentes, para facilitar la tarea del lector, a continuación se realiza una enumeración de todos ellos, recordando la tarea que desempeña cada uno.

- Φ : Lista de operadores de búsqueda local para aplicar luego de la solución inicial factible.
- Θ : Lista de criterios de remoción de nodos.
- p : Probabilidad de extraer un nodo al aplicar el criterio SCT.
- t : Tiempo máximo de refinamiento heurístico aplicando el esquema de extracción/re-inserción con el modelo de re-ubicación.
- η : Porcentaje de variables iniciales x_{si} con menor costo de inserción γ_{si}.
- τ : Cantidad máxima de re-optimizaciones de $C(RM)$ (ó RRM según corresponda) en cada iteración del esquema de refinamiento de destrucción/reparación.
- δ : Proporción de $\tilde{rc}*$, el costo reducido estimado más chico encontrado desde la última re-optimización de $C(RM)$ (ó RRM), utilizado en el criterio de aceptación de x_{si} al modelo actual.
- RC_{max} : umbral aplicado dentro del criterio de aceptación de x_{si} al modelo actual. Cuando se utiliza este umbral en su versión dinámica, se denota DRC_{max}.
- L_{max} : Longitud máxima de las secuencias generadas durante la fase de pricing.
- N_{min} : Cantidad mínima de variables x_{si} que son agregadas al modelo para $i \in \mathcal{I}$, de longitud $|s|$, con menor rc_{si}.
- N_{max}, Dado $i \in \mathcal{I}$ denota la cantidad máxima de variables x_{si} que son agregadas al modelo para $i \in \mathcal{I}$, de longitud $|s|$. La diferencia con respecto a N_{min} es que una vez que se agregaron las N_{min} variables de menor r_{si} para un $i \in \mathcal{I}$, y una longitud de secuencia $|s|$, solo se seguirán agregando variables en caso de que rc_{si} esté por debajo del umbral RC_{max}.

Por restricciones de tiempo computacional para experimentar con todas las posibles combinaciones de parámetros, se analizó empíricamente y en diferentes etapas el comportamiento de algunos de ellos, reduciendo la cantidad de opciones para probar. Para esto, se fijan los parámetros no involucrados en el ciclo de pricing, y consideran distintos valores para el resto, partiendo de los valores iniciales los propuestos por De Franceschi et al. [18], según se indica a continuación. Vale aclarar que en esta fase no se considera el filtro de secuencias introducido en la Sección 5.6.1, y tampoco las reducciones para el vecindario $N(z_0, \mathcal{F})$, que se mencionan en la Sección 4.7.

Con respecto a los criterios de remoción de nodos, en el marco del Algoritmo 5.1, se fijó $p = 0{,}5$ y se consideró $\Theta = \{RA(3), SCT(p,3), NGH^{'}(n)\}$, i.e, 3 iteraciones de RA, 3 de SCT(p) y finalmente NGH$^{'}$, poniendo el foco en mejoras globales en las primeras iteraciones, y concentrándose en mejoras locales en iteraciones avanzadas.

Para la heurística inicial se utilizó el Algoritmo 5.2 junto con la lista $\Phi = \{SPI, SBR, WRI\}$ de operadores de búsqueda local. Cabe destacar que SPI es el que experimentalmente consigue mayores mejoras, marcando una notable diferencia entre el porcentaje de mejora obtenido al aplicarlo, respecto del resto. Es importante destacar que para cada uno de los operadores existen soluciones donde solo uno de ellos consigue mejoras, mientras que los otros no pueden refinar la solución. Por esto último, se utiliza una combinación de todos ellos para definir la lista Φ de aplicación de operadores en la etapa 2 de nuestro algoritmo, y que se prioriza en primer lugar la aplicación de SPI buscando mejorar rápidamente la solución factible básica obtenida con el Algoritmo 5.2. Para facilitar la lectura, se denotará a la solución obtenida luego de aplicar los operadores de búsqueda local como $\mathcal{SI}$.

En esta etapa, el Algoritmo 5.1 se corrió siempre con $t = 1800$ segundos, $\tau = 200$, y $\eta = 25$, i.e. se aplican iteraciones de destrucción/reparación durante a lo sumo 30 minutos, utilizando el 25 % de variables iniciales x_{si} con menor costo de inserción γ_{si}, para cada $i \in \mathcal{I}$.

Se definió la siguiente configuración básica de parámetros para el ciclo de pricing, similar a la utilizada en [18]: $N_{min} = 5$, con la salvedad de que al momento de evaluar secuencias s unitarias (i.e., $|s| = 1$), se fija $N_{min} = 10$, con el objetivo de considerar una mayor cantidad de secuencias unitarias en el pool $\mathcal{S}_P$ que luego serán extendidas, $N_{max} = 10$, $L_{max} = 5$, $\delta = 3$, $RC_{max} = 10$.

Al igual que De Franceschi et al. [18], en la primera aplicación del ciclo de pricing, cuando aún no se resolvió ninguna relajación PL, se fija $\tilde{\pi} = 0$ y $RC_{max} = +\infty$. También, al considerar secuencias unitarias $s \in \mathcal{S}$, se utiliza una estimación heurística del costo reducido, computada como $rc_{si} = c_{av} + c_{vb}$, donde $i = (a, b) \in \mathcal{I}_s$. Esta última decisión está basada en el resultado empírico que se menciona en [18], sobre el hecho de que los costos reducidos obtenidos con la relajación PL para secuencias unitarias no estiman correctamente la efectividad de dichas secuencias.

Finalmente, partiendo de dicha configuración, se experimentó con diferentes valores para los parámetros del ciclo de pricing, en el siguiente orden:

1. $L_{max} = 3, 5, 7$
2. $RC_{max} = 1, 10, 50, 100, 200$ y DRC_{max}
3. $N_{min} = 1, 5, 10$
4. $N_{max} = N_{min} + 5, +10, +15$

En cada una de las etapas, al avanzar a la siguiente se tiene en cuenta el resultado de la anterior, i.e., si dentro de la etapa 1, $L_{max} = 3$ mostró buenos resultados respecto de los otros valores, entonces al experimentar con RC_{max} en la etapa 2 se fija $L_{max} = 3$. Los resultados generales se describen a continuación, si bien en esta etapa no se hace referencia explícita a las tablas de los resultados, éstas se anexan en el Apéndice A.

En primer lugar, con respecto a la longitud máxima de secuencia considerada en el ciclo de pricing, los resultados muestran a $L_{max} = 7$ con el peor rendimiento, mientras que $L_{max} = 3$ y 5 ofrecen un promedio de mejora por instancia similar. Sin embargo, al comparar estos últimos instancia por instancia, se observa que $L_{max} = 3$ consigue un mayor porcentaje de mejora sobre $\mathcal{SI}$ respecto de $L_{max} = 5$, superándolo en 9/22 instancias, mientras que $L_{max} = 5$ consigue mejores resultados solo en 6/22. Este resultado sugiere que incrementar L_{max} deteriora el porcentaje de mejora. Este resultado respalda el hecho de que secuencias de mayor longitud en el ciclo de pricing generan una mayor cantidad de incompatibilidades con secuencias de variables en el modelo, impactando en una gran cantidad de restricciones de precedencia que, dado que se estima heurísticamente $rc_{si} \approx \tilde{rc}_{si}$, no son tenidas en cuenta al computar el costo reducido. De esta manera, $\tilde{rc}_{si}$ no refleja realmente la efectividad se considerar x_{si}, y se termina agregando variables que no son realmente convenientes en el modelo. Sumado a esto, incrementar L_{max} también impactó en el tiempo de cómputo, donde $L_{max} = 5$ utilizó un 7,25 % más de tiempo que $L_{max} = 3$, y $L_{max} = 7$ necesitó un 16,5 % más. Es de esperarse que generar secuciencias más largas durante el ciclo de pricing impacte directamente en el tiempo requerido. Finalmente, se decide fijar $L_{max} = 3$ para los experimentos siguientes.

En segundo lugar, se analizó el impacto de variar $RC_{max} = 1, 10, 50, 100, 200$ en el criterio de aceptación de variables en el modelo. Recordar que x_{si} es incluida en el modelo en caso de que $\tilde{rc}_{si} \leq \text{máx}\{\ \delta\hat{rc}^*,\ RC_{max}\}$, donde $\delta\hat{rc}^*$ denota una proporción del menor costo reducido encontrado hasta el momento, desde la última vez que se calculó $\mathcal{C}(RM)$. Si bien este es un parámetro que está ligado al conjunto de instancias considerado, se espera que utilizar RC_{max} muy chico podría conducir a un criterio de aceptación muy exigente, mientras que valores grandes de RC_{max} dejarían ingresar practicamente cualquier variable al modelo. Los resultados mostraron el mejor comportamiento para $RC_{max} = 50$, lo cual se ajusta con lo dicho anteriormente.

Siguiendo con el criterio de aceptación, se compararon los resultados obtenidos al variar RC_{max} contra los de utilizar el umbral dinámico DRC_{max}. Se implementaron para esto dos estrategias para aceptar la variable x_{si}:

- $\tilde{rc}_{si} \leq \text{máx}\{\ \delta\hat{rc}^*,\ DRC_{max}\}$, manteniendo un esquema parecido al original de De Franceschi et al. [18].
- $\tilde{rc}_{si} \leq DRC_{max}$, en donde el criterio está completamente manejado por el umbral dinámico.

Los resultados muestran tanto en una comparación global, como uno-a-uno, a los criterios dinámicos con mejor performance que los estáticos. Estos últimos consiguen el mejor porcentaje de mejora promedio por instancia, concentrando en la comparación global la mayor cantidad de instancias con mejor solución obtenida. Entre los criterios dinámicos no se observó una gran diferencia, ambos encuentran la misma solución en 19 de 22 instancias. Esto sugiere que dentro de este tipo de esquemas dinámicos, $\delta\hat{rc}^*$ tiene un impacto menor, y el que determina la aceptación es DRC_{max}. Se decide entonces fijar en los experimentos siguientes $\tilde{rc}_{si} \leq DRC_{max}$ para decidir si se debe o no se debe agregar x_{si} al modelo.

El siguiente paso fue estudiar el comportamiento del ciclo de pricing variando el valor con $N_{min} = 1, 5, 10$, manteniendo en todos los casos $N_{max} = N_{min} + 5$. Se espera que al incrementar N_{min}, se deteriore la performance del pricing, debido a que más variables estarían ingresando al modelo aún cuando no cumplan el criterio de aceptación, y afectarían de esta manera el

valor de DRC_{max} haciendo que alcance valores que generen la admisión de variables de baja efectividad práctica. Los resultados respaldan esta hipótesis, mostrando el mejor comportamiento para $N_{min} = 1$, donde además de conseguir un porcentaje de mejora promedio mucho mayor, reduce un 25 % el tiempo total consumido por $N_{min} = 5$ y 10. Esto último se debe a que, dentro del ciclo de pricing, incrementar la cantidad de variables que ingresa al modelo, aumenta también el número de secuencias en el pool $\mathcal{S}_P$, haciendo que se construya un mayor número de secuencias durante la etapa de generación de secuencias, al extender aquellas que están en $\mathcal{S}_P$.

En la última etapa, se experimentó con algunos valores para el desplazamiento entre N_{min} y N_{max}. Fijando $N_{min} = 5$ se consideró $N_{max} = N_{min} + 5, +10, +15$. Se descubrió que incrementar el desplazamiento deteriora el rendimiento del pricing, donde quizás al permitir a un mayor número de variables ser evaluadas, se admitan demasiados casos en donde $\tilde{rc}_{si}$ no estime realmente la efectividad práctica de considerar x_{si}. Además la secuencia s de estas variables es considerada luego para construir otras variables, al considerar extensiones de s.

Finalmente, dados los resultados expuestos anteriormente, la configuración básica que se decide utilizar en las siguientes etapas del esquema de experimentación involucra la siguiente configuración para el ciclo de pricing: $L_{max} = 3$, $N_{min} = 1$, $N_{max} = N_{min} + 5$, y DRC_{max} inicializado en 0 cada vez que se re-optimiza la relajación PL del RM.

6.1.2. Análisis de alternativas para el algoritmo

En esta etapa se comparan las diferentes versiones del Algoritmo 5.1, que incluyen combinaciones del filtro de secuencias propuesto en la Sección 5.6.1, la reducción original de Toth y Tramontani [55] junto con la adaptación propuesta en la Sección 4.7, partiendo de la configuración definida en la fase anterior.

Cabe aclarar que se fijó $\lambda = 2$ en todos los casos que se aplica la reducción de vecindario $N(z_0, \mathcal{F}, \lambda) \subseteq N(z_0, \mathcal{F})$.

Para facilitar la lectura, se introduce la siguiente notación:

- $\mathcal{B}$ denota la configuración básica construída durante la etapa anterior.
- $r\mathcal{TT}$ denota la reducción original de Toth y Tramontani [55] en el vecindario $N(z_0, \mathcal{F}, \lambda)$.
- $r\mathcal{OR}$ denota la reducción adaptada para $N(z_0, \mathcal{F}, \lambda)$ que explota la relación de precedencia entre clientes, introducida en la Sección 5.5.
- *vi* y *pr* se utilizarán como sufijos para denotar donde es aplicada la reducción. Por ejemplo, $r\mathcal{TT}vi$ significa que la reducción de vecindario $r\mathcal{TT}$ está aplicada solamente sobre el conjunto de variables iniciales, mientras que $r\mathcal{TT}pr$ significa que se aplica dicha reducción sobre las variables iniciales y también durante la fase de pricing.
- $\mathcal{SGF}$ se utilizará para referirnos al filtro de secuencias (Sección 5.6.1).

Si bien lo ideal sería explorar todas las combinaciones posibles de diferentes características, junto con distintos valores para λ y demás parámetros que no fueron estudiados en la etapa de experimentación anterior (e.g., p en el criterio SCT, o η en el porcentaje de variables iniciales), dado el tiempo computacional que esto implica, se realiza un análisis por grupos, diferenciando aquellos que implementan la reducción $r\mathcal{TT}$ por un lado, y los que implementan la reducción $r\mathcal{OR}$ por otro. En cada uno de los grupos, se consideran diferentes versiones, que incluyen la reducción en *vi*, *pr*, y también $\mathcal{SGF}$. Se mantiene la configuración original sobre el resto de los parámetros.

Se realizó un análisis considerando los siguientes grupos:

- *Esquemas* $r\mathcal{TT}$*:* que considera $\mathcal{B}$, $r\mathcal{TT}vi$, $r\mathcal{TT}vi + \mathcal{SGF}$, $r\mathcal{TT}pr$ y $r\mathcal{TT}pr + \mathcal{SGF}$.
- *Esquemas* $r\mathcal{OR}$*:* que involucra $\mathcal{B}$, $r\mathcal{OR}vi$, $r\mathcal{OR}vi + \mathcal{SGF}$, $r\mathcal{OR}pr$ y $r\mathcal{OR}pr + \mathcal{SGF}$.

Los resultados obtenidos con la reducción original de Toth y Tramontani [55] sugieren que aplicarla durante la etapa de pricing deteriora la performance del esquema, perjudicando los porcentajes de mejora por instancia. Esto se justifica desde la teoría dado que el esquema $r\mathcal{TT}$ no tiene en cuenta la relación de precedencia entre clientes, y por lo tanto muchas veces podría suceder que los vecindarios reducidos $N(z_0, \mathcal{F}, \lambda)$ sean muy pobres en mejoras para z_0. Se recuerda que en este esquema, la construcción de $\mathcal{F}_i$ considera individualmente cada nodo según el costo de inserción en $i \in \mathcal{I}$, para decidir (heurísticamente) si debe ser considerado al generar secuencias s para potenciales asignaciones de s en i. En muchos casos, cuando el nodo de recolección p_k está en $\mathcal{F}_i$, podría suceder que la entrega asociada d_k haya sido excluida de $\mathcal{F}_i$, o que ni siquiera esté en $\mathcal{F}_j$ con j un punto de inserción ubicado antes que i, dentro de la misma ruta. Análogamente, podría darse la situación en que p_k está en F_i, pero d_k no, e incluso no esté en $\mathcal{F}'_j$ con j' un punto de inserción ubicado después que i, dentro de la misma ruta.

Por otro lado, las versiones que implementan la reducción mostraron que son capaces de utilizar menos tiempo, aplicando más iteraciones. Cuando la reducción es solo considerada en las variables iniciales (i.e., en las versiones $r\mathcal{TT}vi$), el tiempo promedio por instancia disminuye en 1 % respecto de $\mathcal{B}$, y la cantidad de iteraciones se mantiene similar. Cuando se aplica la reducción en el pricing (i.e., en las versiones $r\mathcal{TT}pr$), el tiempo promedio por instancia desciende un 12 % respecto de $\mathcal{B}$, y se consigue aplicar un 34 % más de iteraciones promedio por instancia. Tratando de aprovechar esta situacíón, buscando mejorar el desempeño del esquema $r\mathcal{TT}pr$, se experimentó con $r\mathcal{TT}pr + \mathcal{SGF}$ y $L_{max} = 5$. El objetivo era evaluar si el tiempo que se ahorraba respecto de $\mathcal{B}$ tenía sentido invertirlo generando secuencias de mayor longitud durante al fase de pricing. Aún así, el esquema no mostró mejoras considerables. Esto está nuevamente justificado con las ideas expuestas en el párrafo anterior.

El filtro en la generación de secuencias (i.e., la versión $\mathcal{SGF}$), tampoco mostró un buen rendimiento en los esquemas $r\mathcal{TT}$, perjudicando muchas veces el promedio de mejora por instancia. En este esquema, se entiende que el filtro no resulte útil ya que en muchos casos es posible que la reducción excluya nodos de recolección de conjuntos $\mathcal{F}_i$, con $i \in \mathcal{I}$, dificultando la generación de secuencias que contengan en forma exclusiva los nodos de entrega asociados. Una vez más, se entiende que esto es consecuencia del criterio individual que se aplica sobre los nodos para determinar la reducción, donde no se tiene en cuenta la relación de precedencia.

En general, la versión $r\mathcal{TT}$ que mostró el mejor desempeño es $r\mathcal{TT}vi$, consiguiendo el mejor promedio de mejora por instancia, y obteniendo el mayor porcentaje de mejora global en 5/22 instancias, contra las 2/22 que consigue $\mathcal{B}$.

Muy diferente es la situación en los esquemas $r\mathcal{OR}$ que utilizan la reducción adaptada que aprovecha la relación de precedencia entre clientes. En estos, aplicar las versiones $r\mathcal{OR}pr$ (i.e., durante la etapa de pricing), incrementa no solo el promedio de mejora por instancia, sino que también la cantidad de instancias en las que se obtiene mejor resultado al comparar instancia por instancia contra esquemas $r\mathcal{OR}vi$. De la misma manera, agregar el filtro de secuencias al experimentar con la versión $r\mathcal{OR}pr+\mathcal{SGF}$ también mostró muy buenos resultados.

Al igual que en los esquemas anteriores, la reducción de vecindario mostró ser capaz de reducir el tiempo promedio consumido en un 8 % respecto de $\mathcal{B}$, aplicando un 22 % más de iteraciones. Es razonable también que, si bien disminuye el tiempo utilizado, lo haga en menor medida que las reducciones $r\mathcal{TT}$, siendo que los conjuntos $\mathcal{F}_i$ construidos con el esquema $r\mathcal{OR}$ son superconjuntos de aquellos obtenidos mediante $r\mathcal{TT}$.

Finalmente, se hicieron pruebas variando algunos de los valores fijados en la etapa de experimentación anterior. La mejor versión fue $r\mathcal{OR}pr + \mathcal{SGF}$, consiguiendo en la comparación global entre esquemas $r\mathcal{OR}$, la mayor cantidad de instancias con mayor mejora.

En la siguiente etapa se realiza una comparación global de las versiones: $\mathcal{B}$, $r\mathcal{TT}vi$ y $r\mathcal{OR}pr+\mathcal{SGF}$, cuyos resultados son expuestos en las Tablas 6.1, 6.2 y 6.3.

6.1.3. Comparación global

En esta sección se realiza una comparación global de las versiones del Algoritmo 5.1 que mostraron mejor desempeño en las etapas anteriores, considerando la versión básica $\mathcal{B}$. Se eligió la mejor versión que implementa la reducción original de Toth y Tramontani [55], y la versión en la que se observó el mejor desempeño implementando la nueva reducción adaptada para el $VRPPD1-1$.

Las Tablas 6.1, 6.2 y 6.3 reportan los resultados obtenidos utilizando $t = 5$ y 30 minutos, y $\lambda = 2$ en los esquemas que implementan reducciones de vecindario. A través de $t = 5mins$ se pretende evaluar el desempeño no solo en el porcentaje de mejora, sino también comparar cuán rápido logran mejorar la calidad de la solución.

Las columnas de las tablas tienen el siguiente significado:

- **Instancia**, indica el nombre de cada instancia, junto con la cantidad de clientes y vehículos disponibles, $2n$ y k respectivamente.
- $\mathcal{SI}$, correspondiente a la solución obtenida luego de aplicar la heurística inicial y los operadores de búsqueda local.
- $\mathcal{B}$, corresondiente al desempeño de la versión básica el Algoritmo 5.1.
- **Mejor** $r\mathcal{TT}$ correspondiente a la versión $r\mathcal{TT}vi$ que presentó mejor rendimiento durante el análisis en la etapa de experimentación anterior.

- **Mejor** $r\mathcal{OR}$ correspondiente a la versión $r\mathcal{OR}pr + \mathcal{SGF}$, seleccionada de la misma forma que $r\mathcal{TT}vi$ durante la etapa de experimentación anterior.

Para cada esquema se reporta el promedio de mejora respecto de la solución $\mathcal{SI}$, indicando el peor, mejor, y caso promedio de 5 corridas en las cuales se utilizaron diferentes semillas iniciales para los esquemas de extracción de nodos. Se reporta también el promedio de tiempo (en segundos) y cantidad de iteraciones en cada instancia. Al final se agrega el promedio por instancia, junto con la varianza del promedio de mejora por instancia.

En aspectos generales, la Tabla 6.2 muestra que en casi todas las intancias los 3 esquemas encuentran mejoras, en el caso promedio, respecto de la solución inicial. En las instancias con muy pocos clientes, este comportamiento es esperado dado que la cantidad de combinaciones posibles es relativamente chica, y es razonable pensar que los operadores de búsqueda local consiguen explorar un gran número de soluciones, proporcionando quizás soluciones óptimas, o dejando muy poco margen para que el esquema de re-ubicación logre obtener mejoras. De todas formas, en ninguno de los casos se tiene garantías de esta situación, y de hecho, la instancia *D023-03* es un contraejemplo para estos casos. También se puede observar que en las instancias chicas, y en solo en algunas instancias de tamaño mediano, no se alcanza a consumir el tiempo máximo de ejercución. Esto es porque el vecindario definido por la lista Θ de criterios de remoción de nodos es consumida completamente sin que el esquema consiga una solución mejor que la actual.

Por otro lado, con respecto a los diferentes esquemas, también se observa en la Tabla 6.2 que $r\mathcal{OR}$ concentra la mayor cantidad de instancias con el promedio de mejora más alto en todos lo casos (i.e., peor, promedio y mejor). Si bien en el caso promedio supera en 7/22 instancias a las 6/22 de $r\mathcal{TT}$ y a las 4/22 de $\mathcal{B}$, vale destacar que al observar el peor caso se puede ver que $r\mathcal{OR}$ consigue mucho mejores resultados obteniendo 9/22 instancias, contra las 2/22 de $r\mathcal{TT}$ y las 3/22 de $\mathcal{B}$. En principio, este resultado sugiere que el esquema $r\mathcal{OR}$ consigue muchas más instancias con mejor resultado que los otros esquemas, pero aún es posiblen que en las instancias en que pierde, lo haga por mucho, o que cuando gane, lo haga apenas superando el desempeño de otros esquemas. En este sentido, si se observa el porcentaje de mejora promedio por instancia, $r\mathcal{TT}$ levemente supera a los otros esquemas, quedando en segundo lugar $r\mathcal{OR}$ y en tercero $\mathcal{B}$. Aún así, la diferencia es pequeña, y en la comparación uno-a-uno entre las instancias, la diferencia la hace el caso *D121-11c*, donde $r\mathcal{TT}$ supera por casi 7 % a los otros esquemas. Se debe tener en cuenta entonces que el promedio de mejora por instancia puede ser mal interpretado en términos del comportamiento del algoritmo. Si este consigue una mejora muy importante en una única instancia, pero funciona con baja calidad en el resto, podría suceder que tenga un promedio elevado. Por otro lado, un esquema que gane en muchas instancias con poco margen, y que pierda por mucho en alguna instancia, podría tener un promedio de mejora por instancia muy bajo. Para hacer más justa la comparación, se observa también la varianza de las diferentes versiones del Algoritmo. Dado que el comportamiento del esquema podría depender mucho de la semilla utilizada para el criterio de remoción, se espera que en general el comportamiento del algoritmo sea consistente. En particular, la versión $r\mathcal{OR}$ se destaca en este aspecto, con una varianza levemente superior al 0,1 % mientras que $r\mathcal{TT}$ y $\mathcal{B}$ tienen una varianza de 1,88 % y 1,68 % respectivamente. Esto último se refleja en el hecho de que en el peor caso $r\mathcal{OR}$ obtiene, en general, el mayor porcentaje de mejora en cada instancia.

En la Tabla 6.1, donde $t = 5$ minutos, el comportamiento es similar, con la salvedad de que

el mejor promedio por instancia lo tiene el esquema $r\mathcal{OR}$, con 5,91 % respecto del 5,62 % de $r\mathcal{TT}$ y el 5,3 % de $\mathcal{B}$. Sumado a esto, la mayor cantidad de instancias con promedio de mejora más alto, también las consigue $r\mathcal{OR}$, tanto en el peor caso y mejor caso, como en el caso promedio. Y finalmente, la varianza sugiere que el esquema $r\mathcal{OR}$ es más estable y que no depende tanto de la semilla inicial. Este resultado es muy favorable para la variante de reducción de vecindario propuesta, y sugiere que incluso en un marco de tiempo más acotado, el desempeño de $r\mathcal{OR}$ es mejor que las otras alternativas.

Por otra parte, también se hicieron comparaciones uno a uno entre los esquemas. En este sentido, se debe tener claro que una comparación global sugiere el esquema, entre los considerados, que concentra los mejores resultados. La comparación uno a uno nos permite sacar conclusiones sólo respecto del comportamiento entre las versiones que se comparan, pudiendo ser el caso que un esquema sea bueno en una comparación global, pero pierda frente a una comparación uno a uno con alguno de los integrantes de la comparación global. Esta situación se da con $r\mathcal{TT}$ y $r\mathcal{OR}$. Se encontró que el esquema $r\mathcal{TT}$ pierde con $\mathcal{B}$ en cantidad de instancias con mejor resultado promedio, ganando $\mathcal{B}$ con 9/22 respecto de las 7/22 de $r\mathcal{TT}$ aún cuando este último lo supera levemente en el caso promedio de mejora. Por otro lado, $r\mathcal{OR}$ supera tanto en cantidad de instancias con mejor desempeño y promedio a $\mathcal{B}$, pero no muestra buenos resultados al comparar uno a uno con $r\mathcal{TT}$, aún cuando $r\mathcal{OR}$ ofrece los mejores resultados en la comparación global.

Teniendo en cuenta el resultado del párrafo anterior, y el hecho de que lo ideal sería conseguir una versión que *gane* tanto en la comparación global como en todas las comparaciones uno-a-uno, se realizó una nueva comparación global utilizando una nueva versión de $r\mathcal{OR}$, que también había había mostrado un buen desempeño en la etapa de experimentación anterior. En la Tabla 6.3 se muestan los resultados de esta nueva comparación, donde $r\mathcal{OR}2$ es una variante de $r\mathcal{OR}$ con $L_{max} = 5$. Esta versión supera a las demás en la cantidad de instancias con mejor resultado, e incluso obtiene un mayor promedio de mejora. Sin embargo, el resultado en las comparaciones uno-a-uno se invirtió respecto de la versión anterior. En este caso $r\mathcal{OR}2$ logra superar el promedio de mejora de $r\mathcal{TT}$, consiguiendo 8,31 % respecto de 8,06 % de $r\mathcal{TT}$, e incluso le gana en cantidad de instancias con mejor porcentaje de mejora. Sin embargo, al comparar $r\mathcal{OR}2$ contra $\mathcal{B}$, $r\mathcal{OR}$ queda en desventaja con una situación similar a la de la comparación uno a uno entre $\mathcal{B}$ y $r\mathcal{TT}$. Se tiene entonces que la nueva versión $r\mathcal{OR}$ domina la comparación global, y si bien no queda claro que sea mejor que $\mathcal{B}$ en el uno a uno ofrece mejores resultados que $r\mathcal{TT}$. Aún a favor de $r\mathcal{OR}2$, la varianza sigue siendo mucho más chica que los $\mathcal{B}$ y $r\mathcal{TT}$, con 0,22 % respecto del 1,87 % de $\mathcal{B}$ y 1,67 % de $r\mathcal{TT}$.

Finalmente, es importante destacar que las versiones $r\mathcal{OR}$ consiguen el mejor desempeño en las comparaciones globales, obteniendo la mayor cantidad de instancias con mayor porcentaje promedio de mejora, con un promedio similar de mejora en general, y con una fuerte ventaja en la varianza, sugiriendo que este esquema es el más consistente en la práctica.

Instancia			Sol. Inicial	Básico					Mejor TT					Mejor OR				
Nombre	2n	k		Peor %	Av. %	Mejor %	Av. ET	AVI	Peor %	Av. %	Mejor %	Av. ET	AVI	Peor %	Av. %	Mejor %	Av. ET	AVI
D022-04g	20	4	285.6037	0.00	0.00	0.00	**0.89**	**26.00**	0.00	0.00	0.00	0.74	**26.00**	0.00	0.00	0.00	0.81	**26.00**
D023-03g	22	3	552.5039	0.00	**1.47**	**1.84**	0.80	**29.00**	0.00	**1.47**	**1.84**	**0.81**	**29.00**	0.00	**1.47**	**1.84**	0.74	**29.00**
D030-03g	28	3	495.047	0.00	**0.16**	**0.81**	**2.57**	36.00	0.00	**0.16**	**0.81**	2.52	36.00	0.00	**0.16**	**0.81**	2.34	**37.00**
D033-04g	32	4	496.1447	0.00	0.00	0.00	**4.65**	**38.00**	0.00	0.00	0.00	4.49	**38.00**	0.00	0.00	0.00	4.02	**38.00**
D051-06c	50	6	539.4449	0.00	0.00	0.00	**22.65**	**56.00**	0.00	0.00	0.00	21.86	**56.00**	0.00	0.00	0.00	19.78	**56.00**
D076-11c	74	11	713.839	7.88	9.52	**10.40**	**180.45**	133.00	7.88	8.96	**10.40**	178.47	132.00	**9.31**	**10.03**	**10.40**	170.74	**174.00**
D101-09c	100	9	1069.5505	2.19	7.04	12.91	**301.85**	97.00	4.58	**10.74**	**17.71**	301.47	108.00	**5.68**	8.53	12.45	301.08	**200.00**
D101-11c	100	11	1163.0662	20.48	31.01	40.97	300.87	137.00	20.48	30.80	37.12	**302.26**	141.00	**32.17**	**37.49**	**45.48**	300.30	**212.00**
D121-11c	120	11	1378.0127	**11.53**	16.07	23.65	**303.59**	94.00	6.58	**18.57**	**35.35**	301.80	100.00	6.05	16.53	21.95	302.84	**143.00**
D151-14b	150	14	1255.8763	0.00	**1.62**	3.90	**307.41**	24.00	0.00	1.61	3.97	305.63	26.00	0.00	1.15	**4.23**	301.80	**55.00**
D151-14c	150	14	1243.0783	0.00	**0.14**	**0.64**	305.92	25.00	0.00	**0.14**	**0.64**	**307.66**	26.00	0.00	0.13	0.29	302.97	**54.00**
D200-18b	198	18	1672.0877	**0.05**	0.82	2.08	**324.82**	9.00	**0.05**	**0.93**	2.15	309.30	9.00	0.02	0.76	**2.22**	305.26	**16.00**
D200-18c	198	18	1382.1755	0.00	0.00	**0.01**	**312.43**	9.00	0.00	0.00	**0.01**	305.86	9.00	0.00	0.00	**0.01**	297.38	**20.00**
D201-05k	200	5	8479.8618	**0.58**	**1.37**	**2.32**	305.29	12.00	0.36	1.30	**2.32**	306.97	**14.00**	0.36	0.94	2.06	**309.84**	**14.00**
D241-10k	240	10	5690.8845	0.00	**0.07**	**0.22**	312.52	5.00	0.00	0.05	**0.22**	329.21	**6.00**	0.00	0.05	**0.22**	**340.90**	**6.00**
100vrpnc12	48	4	948.8734	**3.18**	12.00	17.58	**301.63**	109.00	**3.18**	11.86	**17.81**	301.49	109.00	3.12	**14.59**	**17.81**	301.40	**155.00**
100vrpnc3	74	7	845.9476	0.32	2.04	4.89	**301.92**	109.00	0.32	2.04	4.89	300.58	109.00	**1.02**	**5.75**	**11.10**	300.19	**209.00**
120vrpnc11	98	9	1527.1633	6.59	20.02	36.28	**303.09**	69.00	6.59	**20.09**	**36.54**	302.08	76.00	**15.69**	17.85	26.34	301.36	**118.00**
150vrpnc4	98	9	1355.0136	0.00	3.82	8.81	303.97	26.00	0.00	3.84	8.81	**304.28**	28.00	0.00	**5.59**	**9.48**	300.39	**55.00**
199vrpnc5	118	11	1644.7277	0.02	1.85	**4.51**	**319.46**	9.00	0.02	2.48	4.48	314.09	9.00	**0.09**	**2.54**	3.54	304.94	**15.00**
50vrpnc1	148	14	572.1237	0.00	2.64	6.59	25.11	83.00	0.00	**3.21**	**8.81**	**28.07**	**91.00**	0.00	2.64	6.59	23.34	90.00
75vrpnc2	198	19	682.4024	**2.41**	5.04	**8.19**	**167.09**	**156.00**	**2.41**	**5.48**	7.70	162.76	149.00	**2.41**	3.88	4.86	116.69	147.00
Promedio				2.51	5.30	8.48	**214.04**	59.00	2.38	5.62	**9.16**	213.29	60.00	**3.45**	**5.91**	8.26	209.51	**85.00**
Varianza				1.40					1.79					0.13				

Cuadro 6.1.: Comparación global con $t = 5$ minutos de las versiones $\mathcal{B}$, $r\mathcal{TT}$ y $r\mathcal{OR}$ de Algoritmo 5.1.

Instancia			Sol. Inicial	$\mathcal{B}$					Mejor TT					Mejor OR				
Nombre	2n	k		Peor %	Av. %	Mejor %	Av. ET	AVI	Peor %	Av. %	Mejor %	Av. ET	AVI	Peor %	Av. %	Mejor %	Av. ET	AVI
D022-04g	20	4	285.6037	0.00	0.00	0.00	**0.75**	**26.00**	0.00	0.00	0.00	0.73	**26.00**	0.00	0.00	0.00	0.69	**26.00**
D023-03g	22	3	552.5039	0.00	**1.47**	**1.84**	**0.79**	**29.00**	0.00	**1.47**	**1.84**	0.77	**29.00**	0.00	**1.47**	**1.84**	0.73	**29.00**
D030-03g	28	3	495.047	0.00	**0.16**	**0.81**	**2.51**	**36.00**	0.00	**0.16**	**0.81**	**2.51**	**36.00**	0.00	**0.16**	**0.81**	2.31	35.00
D033-04g	32	4	496.1447	0.00	0.00	0.00	**4.58**	**38.00**	0.00	0.00	0.00	4.48	**38.00**	0.00	0.00	0.00	3.96	36.00
D051-06c	50	6	539.4449	0.00	0.00	0.00	21.92	**56.00**	0.00	0.00	0.00	**22.01**	**56.00**	0.00	0.00	0.00	19.61	50.00
D076-11c	74	11	713.839	7.88	9.52	**10.40**	177.79	132.00	7.88	8.96	**10.40**	**180.83**	132.00	**9.31**	**10.03**	**10.40**	169.78	**137.00**
D101-09c	100	9	1069.5505	3.62	9.72	14.04	862.11	277.00	5.27	**12.88**	**19.82**	**1059.15**	**321.00**	**5.68**	8.78	13.05	568.89	255.00
D101-11c	100	11	1163.0662	21.23	31.73	42.30	802.94	300.00	21.23	31.29	37.12	**862.55**	298.00	**32.91**	**38.42**	**46.81**	847.45	**362.00**
D121-11c	120	11	1378.0127	12.31	24.52	**39.97**	**1798.13**	**388.00**	16.02	**31.11**	39.82	1619.91	380.00	**16.43**	24.53	28.04	1202.48	368.00
D151-14b	150	14	1255.8763	6.03	8.57	11.74	1803.31	153.00	6.11	7.14	8.19	**1805.38**	148.00	**6.97**	**9.30**	**12.70**	1801.57	**271.00**
D151-14c	150	14	1243.0783	0.95	**3.38**	**10.68**	1804.99	153.00	**1.13**	1.96	3.42	**1806.25**	147.00	0.69	1.74	3.33	1781.87	**243.00**
D200-18b	198	18	1672.0877	**3.60**	**5.59**	9.31	1812.96	55.00	2.48	4.85	**9.54**	**1814.91**	53.00	2.11	4.54	6.93	1804.96	**81.00**
D200-18c	198	18	1382.1755	0.00	0.20	0.72	1804.59	61.00	0.00	0.20	0.72	**1812.14**	57.00	**0.15**	**0.97**	**1.36**	1804.12	**96.00**
D201-05k	200	5	8479.8618	1.59	4.20	5.57	**1809.28**	**107.00**	3.60	4.31	**6.42**	1807.74	91.00	**4.05**	**4.82**	6.38	1806.38	77.00
D241-10k	240	10	5690.8845	0.00	0.90	2.67	1815.18	**47.00**	**0.45**	**1.77**	**3.61**	1816.21	38.00	0.00	1.11	**3.61**	**1829.71**	38.00
100vrpnc12	48	4	948.8734	**17.58**	**20.02**	**29.30**	738.59	251.00	17.48	17.89	18.84	**845.18**	**277.00**	3.24	14.84	17.84	599.69	**277.00**
100vrpnc3	74	7	845.9476	0.32	6.51	**12.93**	**721.50**	240.00	0.32	**6.61**	**12.93**	712.82	260.00	**2.12**	5.97	11.10	423.66	**276.00**
120vrpnc11	98	9	1527.1633	6.59	20.39	36.54	**1306.66**	227.00	6.59	20.09	**36.55**	957.93	181.00	**15.74**	**22.16**	36.13	911.88	**314.00**
150vrpnc4	98	9	1355.0136	**5.41**	8.99	11.38	1803.71	164.00	5.31	8.37	11.50	**1803.92**	152.00	5.30	**9.36**	**15.81**	1756.23	**323.00**
199vrpnc5	118	11	1644.7277	**8.01**	**9.67**	11.74	**1817.54**	57.00	6.40	9.46	11.79	1817.18	51.00	6.01	9.10	**12.96**	1808.31	**89.00**
50vrpnc1	148	14	572.1237	0.00	2.64	6.59	25.46	83.00	0.00	**3.21**	**8.81**	**27.88**	**91.00**	0.00	2.64	6.59	23.41	90.00
75vrpnc2	198	19	682.4024	**2.41**	5.04	**8.19**	**168.57**	**151.00**	**2.41**	**5.48**	7.70	163.01	149.00	**2.41**	3.88	4.86	117.97	147.00
Promedio				4.43	7.87	**12.12**	**959.27**	138.00	4.67	**8.06**	11.36	951.98	137.00	**5.14**	7.90	10.93	876.62	**165.00**
Varianza				1.87					1.67					0.11				

Cuadro 6.2.: Comparación global con $t = 30$ minutos de las versiones $\mathcal{B}$, $r\mathcal{TT}$ y $r\mathcal{OR}$ del Algoritmo 5.1.

Instancia			Sol. Inicial	Básico					Mejor TT					Mejor OR				
Nombre	2n	k		Peor %	Av. %	Mejor %	Av. ET	AVI	Peor %	Av. %	Mejor %	Av. ET	AVI	Peor %	Av. %	Mejor %	Av. ET	AVI
D022-04g	20	4	285.6037	0.00	0.00	0.00	0.75	**26.00**	0.00	0.00	0.00	0.73	**26.00**	0.00	0.00	0.00	**1.10**	**26.00**
D023-03g	22	3	552.5039	0.00	**1.47**	**1.84**	0.79	**29.00**	0.00	**1.47**	**1.84**	0.77	**29.00**	0.00	**1.47**	**1.84**	**1.24**	**29.00**
D030-03g	28	3	495.047	0.00	**0.16**	**0.81**	2.51	**36.00**	0.00	**0.16**	**0.81**	2.51	**36.00**	0.00	**0.16**	**0.81**	**4.00**	**36.00**
D033-04g	32	4	496.1447	0.00	0.00	0.00	4.58	**38.00**	0.00	0.00	0.00	4.48	**38.00**	0.00	0.00	0.00	**8.08**	**38.00**
D051-06c	50	6	539.4449	0.00	0.00	0.00	21.92	**56.00**	0.00	0.00	0.00	22.01	**56.00**	0.00	0.00	0.00	**38.18**	**56.00**
D076-11c	74	11	713.839	**7.88**	**9.52**	**10.40**	177.79	132.00	**7.88**	8.96	**10.40**	180.83	132.00	7.37	8.47	**10.40**	**247.38**	**147.00**
D101-09c	100	9	1069.5505	3.62	9.72	14.04	862.11	277.00	**5.27**	**12.88**	**19.82**	**1059.15**	**321.00**	4.58	8.46	13.47	826.63	287.00
D101-11c	100	11	1163.0662	21.23	31.73	**42.30**	802.94	300.00	21.23	31.29	37.12	862.55	298.00	**33.04**	**35.33**	39.20	**1309.06**	**396.00**
D121-11c	120	11	1378.0127	12.31	24.52	**39.97**	**1798.13**	388.00	16.02	31.11	39.82	1619.91	380.00	**27.30**	**33.53**	38.77	1604.56	**463.00**
D151-14b	150	14	1255.8763	6.03	**8.57**	11.74	1803.31	153.00	**6.11**	7.14	8.19	**1805.38**	148.00	5.96	7.94	**11.82**	1800.46	**258.00**
D151-14c	150	14	1243.0783	0.95	**3.38**	**10.68**	1804.99	153.00	**1.13**	1.96	3.42	1806.25	147.00	0.69	1.77	3.64	**1809.94**	**229.00**
D200-18b	198	18	1672.0877	**3.60**	**5.59**	9.31	1812.96	55.00	2.48	4.85	**9.54**	**1814.91**	53.00	1.91	4.90	8.77	1805.98	**79.00**
D200-18c	198	18	1382.1755	0.00	0.20	0.72	1804.59	61.00	0.00	0.20	0.72	**1812.14**	57.00	0.00	**0.50**	**1.46**	1807.03	**89.00**
D201-05k	200	5	8479.8618	1.59	4.20	5.57	1809.28	**107.00**	**3.60**	**4.31**	**6.42**	1807.74	91.00	3.36	3.97	4.34	**1813.71**	61.00
D241-10k	240	10	5690.8845	0.00	0.90	2.67	1815.18	**47.00**	**0.45**	**1.77**	**3.61**	1816.21	38.00	0.00	0.39	0.64	**1819.60**	25.00
100vrpnc12	48	4	948.8734	**17.58**	**20.02**	**29.30**	738.59	251.00	17.48	17.89	18.84	845.18	277.00	17.41	17.90	18.82	**1214.19**	**303.00**
100vrpnc3	74	7	845.9476	0.32	6.51	**12.93**	**721.50**	240.00	0.32	**6.61**	**12.93**	712.82	260.00	**1.76**	6.10	10.63	637.86	**261.00**
120vrpnc11	98	9	1527.1633	6.59	20.39	36.54	1306.66	227.00	6.59	20.09	**36.55**	957.93	181.00	**15.79**	**23.59**	35.96	**1336.24**	**281.00**
150vrpnc4	98	9	1355.0136	**5.41**	8.99	11.38	1803.71	164.00	5.31	8.37	11.50	1803.92	152.00	4.42	**9.42**	**14.88**	**1805.23**	**262.00**
199vrpnc5	118	11	1644.7277	**8.01**	9.67	11.74	**1817.54**	57.00	6.40	9.46	11.79	1817.18	51.00	6.93	**10.04**	**12.89**	1809.67	**78.00**
50vrpnc1	148	14	572.1237	0.00	2.64	6.59	25.46	83.00	0.00	3.21	8.81	27.88	**91.00**	0.00	**4.53**	**9.46**	**43.94**	89.00
75vrpnc2	198	19	682.4024	**2.41**	5.04	**8.19**	168.57	151.00	**2.41**	**5.48**	7.70	163.01	149.00	**2.41**	4.37	4.86	**246.23**	**163.00**
Promedio				4.43	7.87	**12.12**	959.27	138.00	4.67	8.06	11.36	951.98	137.00	**6.04**	**8.31**	11.03	**999.56**	**166.00**
Varianza				1.87					1.67					0.22				

Cuadro 6.3.: Comparación global con $t = 30$ minutos de las versiones $\mathcal{B}$, $r\mathcal{TT}$ y $r\mathcal{OR}$(con $L_{max} = 5$) del Algoritmo 5.1.

7. Conclusiones y trabajo a futuro

En esta tesis se aborda una variante del Problema de Ruteo de Vehículos (VRP) que considera Recolecciones y Entregas (VRPPD) mediante un esquema heurístico de refinamiento basado en Programación Lineal Entera. Se adaptaron las ideas de De Franceschi et al. [18] y Toth y Tramontani [55] aplicadas con éxito en la variante del VRP que incluye restricciones de Distancia y Capacidad (DCVRP), y se propusieron nuevas características para incorporar al esquema tratando de explotar la relación de precedencia entre los clientes de recolección y entrega.

En la primer parte del trabajo se introdujo el VRP junto con la variante VRPPD en su versión más general. Dado que el objetivo de la tesis es analizar el comportamiento del esquema original de De Franceschi et al. [18], junto con los desafíos y dificultades que surjen en este contexto, se decidió utilizar un caso particular del VRPPD clásico. Se relajó la restricción de capacidad sobre los vehículos, y se consideró el caso en que las precedencias entre clientes son uno-a-uno. La motivación está en que luego de una primer etapa en donde se evalúe el comportamiento general del esquema, se pase a una segunda etapa donde se incorporen las restricciones de capacidad. Se dejó en claro que el alcance de esta tesis involucra sólo la primer etapa. En el Capítulo 1 se reflejaron estas ideas.

Durante el Capítulo 2 se resumen las nociones obtenidas a través de un estudio preliminar sobre Programación Lineal Entera en general y las técnicas standard de resolución de formulaciones PLEM. También se agregan nociones de Búsqueda Local. Luego, en el Capítulo 3, se formaliza el VRPPD1-1 y se presentan 2 formulaciones clásicas mediante PLEM, adaptadas de la litetura.

En el Capítulo 4 se introducen, analizan y discuten las ideas originales de De Franceschi et al. [18] y Toth y Tramontani [55]. Se realiza también una revisión de la literatura mencionando otras variantes del VRP donde el esquema fue adaptado con éxito [52, 46, 34].

En el Capítulo 5 se propone un algoritmo de refinamiento heurístico basado en PLE, teniendo en cuenta las ideas del capítulo anterior. Se describen en detalle, para cada una de las etapas del algoritmo propuesto, las dificultades, alternativas consideradas y adaptaciones respecto del esquema original. Se trata de explotar la relación de precedencia entre los clientes para generar variables de manera inteligente, manejando de manera heurística la situación de filas dependiente de las columnas a través de un costo reducido estimado.

Finalmente, en el Capítulo 6 se constrasta el análisis teórico de las etapas anteriores con los resultados empíricos obtenidos de experimentar con diferentes combinaciones de las características analizadas en el capítulo anterior. Se introduce el esquema de experimentación utilizado y se discuten los resultados. Estos muestran ser prometedores y sugieren que la adaptación del esquema al contexto del VRPPD tiene potencial.

Sin embargo, quedan varias líneas de trabajo a futuro:

Heurística Inicial Si bien en el marco de esta tesis, es razonable la combinación de operadores de búsqueda local utilizada para obtener soluciones iniciales, sería muy interesante comparar el desempeño del refinamiento basado en el modelo de re-ubicación frente a soluciones construídas mediante técnicas más complejas.

Restricciones de Capacidad Este es el siguiente paso dentro de la línea de investigación que proponen De Franceschi et al. [18] buscando analizar el comportamiento del modelo de re-ubicación en el marco más general del VRPPD.

Lifting de las restricciones del precedencia Con el objetivo de mejorar la estimación de los costos reducidos, sería muy interesante considerar una versión reforzada de las restricciones de precedencia propuestas dentro del modelo de re-ubicación para el VRPPD1-1.

Criterios dinámicos complejos de evolución para los parámetros Dada la gran cantidad de parámetros, siguiendo la idea del umbral dinámico DRC_{max} propuesto y resultados experimentales muy prometedores, se cree que esta idea podría ser implementada en el resto de los parámetros, tratando de utilizar la información de las variables consideradas de *buena* calidad.

A. Tablas

Instancia			Sol. Inicial	Lmax=3			Lmax=5			Lmax=7		
Nombre	2n	k		% Mejora	Av. ET	AVI	% Mejora	Av. ET	AVI	% Mejora	Av. ET	AVI
D022-04g	20	4	285.6037	0.00	2.49	**26**	0.00	4.69	**26**	0.00	**6.94**	**26**
D023-03g	22	3	552.5039	**1.84**	2.82	**29**	**1.84**	4.70	**29**	**1.84**	**7.05**	**29**
D030-03g	28	3	495.047	0.00	7.65	**34**	0.00	14.98	**34**	0.00	**21.76**	**34**
D033-04g	32	4	496.1447	0.00	14.14	**38**	0.00	33.46	**38**	0.00	**52.39**	**38**
D051-06c	50	6	539.4449	0.00	77.12	**56**	0.00	161.99	**56**	0.00	**260.93**	**56**
D076-11c	74	11	713.839	**9.28**	518.46	**105**	8.31	1171.08	102	7.72	**1808.81**	74
D101-09c	100	9	1069.5505	5.37	1806.02	**139**	**12.17**	1819.12	65	6.78	**1825.88**	36
D101-11c	100	11	1163.0662	**42.68**	**1809.98**	**231**	42.35	1806.37	172	41.77	1805.57	98
D121-11c	120	11	1378.0127	9.19	1801.56	**116**	**21.87**	1799.69	88	5.85	**1827.71**	49
D151-14b	150	14	1255.8763	**1.42**	1817.32	**47**	0.98	1838.88	19	0.00	**1989.48**	9
D151-14c	150	14	1243.0783	1.08	1807.62	**48**	**1.57**	1801.61	21	0.02	**1904.78**	11
D200-18b	198	18	1672.0877	**1.51**	1798.37	**29**	1.32	1895.23	11	0.28	**2016.08**	6
D200-18c	198	18	1382.1755	0.00	1845.14	**15**	0.00	**2002.63**	9	0.00	1945.91	6
D201-05k	200	5	8479.8618	1.96	1806.20	**48**	**4.44**	1810.63	31	3.55	**1887.76**	18
D241-10k	240	10	5690.8845	**0.22**	1796.43	**21**	0.14	1941.04	13	0.14	**2112.65**	11
100vrpnc12	48	4	948.8734	17.55	1803.29	**174**	17.61	1827.88	72	**18.48**	**1832.67**	47
100vrpnc3	74	7	845.9476	**12.68**	1807.10	**190**	11.10	1801.71	78	11.10	**1857.19**	43
120vrpnc11	98	9	1527.1633	15.82	1809.97	**104**	**15.88**	1835.11	50	**15.88**	**1845.17**	34
150vrpnc4	98	9	1355.0136	**10.14**	1817.09	**53**	4.14	**1876.73**	24	3.53	1841.64	12
199vrpnc5	118	11	1644.7277	**10.52**	1828.11	**28**	4.89	1848.96	11	4.57	**1940.85**	6
50vrpnc1	148	14	572.1237	0.00	61.64	54	0.00	136.66	54	**8.81**	**333.41**	**80**
75vrpnc2	198	19	682.4024	**4.86**	513.76	**133**	3.92	1046.19	104	3.92	**1812.47**	93
Promedio				6.64	1206.92	**78**	**6.93**	1294.52	50	6.10	**1406.23**	37

Figura A.1.: Resultados obtenidos con el Algoritmo 5.1 utilizando diferentes valores de L_{max}.

Instancia			Sol. Inicial	$RC_{max} = 1$			$RC_{max} = 10$			$RC_{max} = 50$			$RC_{max} = 100$			$RC_{max} = 200$			$RC^* + DRC_{max}$			DRC_{max}		
Nombre	2n	k		% Mejora	Av. ET	AVI	% Mejora	Av. ET	AVI	% Mejora	Av. ET	AVI	% Mejora	Av. ET	AVI	% Mejora	Av. ET	AVI	% Mejora	Av. ET	AVI	% Mejora	Av. ET	AVI
D022-04g	20	4	285.6037	0.00	2.45	**26**	0.00	2.49	**26**	0.00	2.64	**26**	0.00	**3.09**	**26**	0.00	2.86	**26**	0.00	1.45	**26**	0.00	1.33	**26**
D023-03g	22	3	552.5039	**1.84**	2.81	**29**	**1.84**	2.82	**29**	**1.84**	2.85	**29**	**1.84**	**3.24**	**29**	**1.84**	3.22	**29**	**1.84**	1.34	**29**	**1.84**	1.32	**29**
D030-03g	28	3	495.047	0.00	7.59	**34**	0.00	7.65	**34**	0.00	7.91	**34**	0.00	**9.04**	**34**	0.00	8.52	**34**	0.00	4.45	**34**	0.00	4.37	**34**
D033-04g	32	4	496.1447	0.00	13.76	**38**	0.00	14.14	**38**	0.00	14.37	**38**	0.00	**17.07**	**38**	0.00	16.03	**38**	0.00	8.51	**38**	0.00	8.45	**38**
D051-06c	50	6	539.4449	0.00	73.48	**56**	0.00	77.12	**56**	0.00	78.44	**56**	0.00	**93.70**	**56**	0.00	85.72	**56**	0.00	41.75	**56**	0.00	41.04	**56**
D076-11c	74	11	713.839	**9.28**	590.35	**127**	**9.28**	518.46	105	**9.28**	571.87	105	**9.28**	**660.33**	105	**9.28**	611.38	105	**9.28**	383.99	123	**9.28**	376.09	123
D101-09c	100	9	1069.5505	14.02	1804.41	186	5.37	1806.02	139	14.02	**1814.56**	148	14.02	1805.26	136	14.02	1801.75	142	**15.18**	1807.02	266	**15.18**	1799.85	**268**
D101-11c	100	11	1163.0662	33.41	1801.07	214	**42.68**	**1809.98**	231	32.99	1800.64	209	41.65	1806.98	218	41.68	1805.01	229	33.77	1808.86	316	33.77	1807.83	**318**
D121-11c	120	11	1378.0127	9.19	1819.27	120	9.19	1801.56	116	**27.62**	1810.01	146	**27.62**	**1820.05**	134	**27.62**	1807.66	138	26.40	1799.67	226	26.40	1805.44	**229**
D151-14b	150	14	1255.8763	**2.70**	1818.15	62	1.42	1817.32	47	1.42	1819.63	42	1.33	1807.35	36	1.42	1805.61	39	2.17	**1825.56**	**80**	2.17	1801.88	79
D151-14c	150	14	1243.0783	0.12	1807.97	47	1.08	1807.62	48	1.08	1825.24	44	1.08	**1838.71**	39	1.08	1812.60	41	**1.61**	1815.84	**74**	**1.61**	1813.54	**74**
D200-18b	198	18	1672.0877	2.04	**1841.98**	33	1.51	1798.37	29	2.67	1799.59	25	2.67	1816.37	24	2.67	1826.96	25	**11.31**	1803.83	54	**11.31**	1826.59	**55**
D200-18c	198	18	1382.1755	**0.12**	**1965.08**	16	0.00	1845.14	15	0.00	1817.81	14	0.00	1816.84	13	0.00	1907.85	14	**0.12**	1817.55	**29**	**0.12**	1799.44	**29**
D201-05k	200	5	8479.8618	3.91	1798.61	56	1.96	1806.20	48	**6.75**	**1846.49**	56	1.96	1837.28	45	1.96	1800.91	43	1.96	1799.25	70	1.96	1810.20	**72**
D241-10k	240	10	5690.8845	**0.22**	1799.37	21	**0.22**	1796.43	21	**0.22**	1801.44	20	**0.22**	**1900.13**	20	**0.22**	1892.03	20	**0.22**	1799.67	**30**	**0.22**	1839.57	**30**
100vrpnc12	48	4	948.8734	**17.55**	1800.93	176	**17.55**	**1808.64**	174	**17.55**	1801.49	170	**17.55**	1808.16	167	**17.55**	1801.95	165	**17.55**	1387.36	**205**	**17.55**	1422.84	**205**
100vrpnc3	74	7	845.9476	0.77	1382.93	120	**12.68**	1803.76	189	**12.68**	1804.71	162	**12.68**	1807.28	162	**12.68**	**1807.85**	161	12.26	1801.69	**285**	4.13	1384.96	206
120vrpnc11	98	9	1527.1633	15.82	1809.91	101	15.82	1814.61	104	**35.98**	**1834.95**	130	15.82	1805.99	96	15.82	1814.88	95	27.42	1807.86	**181**	26.01	1816.05	166
150vrpnc4	98	9	1355.0136	4.78	1822.54	56	10.14	1819.92	53	10.09	1802.61	52	10.09	1814.85	52	10.09	**1823.40**	52	**11.03**	1803.76	**97**	**11.03**	1817.82	**97**
199vrpnc5	118	11	1644.7277	9.86	1816.95	32	**10.52**	1829.54	28	8.60	1816.18	24	8.30	1850.17	26	8.30	**1855.93**	26	6.92	1817.08	**40**	4.82	1817.42	24
50vrpnc1	148	14	572.1237	0.00	61.25	**54**	0.00	61.95	**54**	0.00	67.53	**54**	0.00	71.80	**54**	0.00	**72.72**	**54**	0.00	35.08	**54**	0.00	36.22	**54**
75vrpnc2	198	19	682.4024	**7.93**	515.84	137	4.86	516.31	133	4.86	598.55	133	4.86	614.48	133	4.86	**617.02**	133	7.49	325.33	**151**	7.49	342.40	147
Promedio				6.07	1198.03	79	6.64	1207.55	78	**8.53**	1215.43	78	7.77	**1227.64**	75	7.78	1226.45	76	8.48	1168.04	**112**	7.95	1153.39	107

Figura A.2.: Resultados obtenidos con el Algoritmo 5.1 utilizando diferentes valores de RC_{max}.

Instancia			Sol. Inicial	Nmin= 1			Nmin= 5			Nmin=10		
Nombre	2n	k		% Mejora	Av. ET	AVI	% Mejora	Av. ET	AVI	% Mejora	Av. ET	AVI
D022-04g	20	4	285.6037	0.00	0.77	**26**	0.00	**1.33**	**26**	0.00	**1.33**	**26**
D023-03g	22	3	552.5039	**1.84**	0.62	**29**	**1.84**	**1.32**	**29**	**1.84**	**1.32**	**29**
D030-03g	28	3	495.047	0.00	2.28	**34**	0.00	**4.37**	**34**	0.00	**4.37**	**34**
D033-04g	32	4	496.1447	0.00	4.31	**38**	0.00	**8.45**	**38**	0.00	**8.45**	**38**
D051-06c	50	6	539.4449	0.00	19.34	**56**	0.00	**41.04**	**56**	0.00	**41.04**	**56**
D076-11c	74	11	713.839	8.42	130.04	109	**9.28**	**376.09**	**123**	**9.28**	**376.09**	**123**
D101-09c	100	9	1069.5505	14.95	873.60	**329**	**15.18**	**1799.85**	268	**15.18**	**1799.85**	268
D101-11c	100	11	1163.0662	**47.15**	408.08	271	33.77	**1807.83**	**318**	33.77	**1807.83**	**318**
D121-11c	120	11	1378.0127	**28.10**	1313.07	**388**	26.40	**1805.44**	229	26.40	**1805.44**	229
D151-14b	150	14	1255.8763	**2.53**	**1808.31**	**190**	2.17	1801.88	79	2.17	1801.88	79
D151-14c	150	14	1243.0783	**1.97**	1805.97	**183**	1.61	**1813.54**	74	1.61	**1813.54**	74
D200-18b	198	18	1672.0877	3.54	1807.10	**76**	**11.31**	**1826.59**	55	**11.31**	**1826.59**	55
D200-18c	198	18	1382.1755	**0.38**	**1809.20**	**82**	0.12	1799.44	29	0.12	1799.44	29
D201-05k	200	5	8479.8618	**4.81**	1802.97	**144**	1.96	**1810.20**	72	1.96	**1810.20**	72
D241-10k	240	10	5690.8845	**2.16**	1797.69	**68**	0.22	**1839.57**	30	0.22	**1839.57**	30
100vrpnc12	48	4	948.8734	**17.84**	623.48	**258**	17.55	1422.84	205	17.63	**1811.90**	177
100vrpnc3	74	7	845.9476	**12.93**	1016.83	**389**	4.13	**1384.96**	206	0.77	1357.86	114
120vrpnc11	98	9	1527.1633	25.90	1282.42	**252**	**26.01**	**1816.05**	166	25.99	1803.36	102
150vrpnc4	98	9	1355.0136	10.88	1800.01	**230**	**11.03**	**1817.82**	97	4.74	1810.38	50
199vrpnc5	118	11	1644.7277	**11.05**	1811.91	**92**	4.82	1817.42	24	5.85	**1827.00**	30
50vrpnc1	148	14	572.1237	0.00	16.87	**54**	0.00	36.22	**54**	0.00	**57.25**	**54**
75vrpnc2	198	19	682.4024	4.86	112.91	113	**7.49**	342.40	**147**	3.43	**392.41**	100
Promedio				**9.06**	920.35	**155**	7.95	1153.39	107	7.38	**1172.60**	95

Figura A.3.: Resultados obtenidos con el Algoritmo 5.1 utilizando diferentes valores de N_{min}.

Instancia			Sol. Inicial	Nmin (+5)			Nmin (+10)			Nmin (+15)		
Nombre	2n	k		% Mejora	Av. ET	AVI	% Mejora	Av. ET	AVI	% Mejora	Av. ET	AVI
D022-04g	20	4	285.6037	0.00	1.33	**26**	0.00	1.33	**26**	0.00	**1.34**	**26**
D023-03g	22	3	552.5039	**1.84**	1.32	**29**	**1.84**	**1.33**	**29**	**1.84**	**1.33**	**29**
D030-03g	28	3	495.047	0.00	4.37	**34**	0.00	4.76	**34**	0.00	**4.84**	**34**
D033-04g	32	4	496.1447	0.00	8.45	**38**	0.00	8.99	**38**	0.00	**9.28**	**38**
D051-06c	50	6	539.4449	0.00	41.04	**56**	0.00	42.89	**56**	0.00	**44.02**	**56**
D076-11c	74	11	713.839	**9.28**	**376.09**	**123**	**9.28**	320.78	109	**9.28**	340.96	110
D101-09c	100	9	1069.5505	15.18	1799.85	**268**	13.33	1800.27	260	**19.84**	**1803.88**	261
D101-11c	100	11	1163.0662	**33.77**	**1807.83**	**318**	20.55	1365.94	202	20.55	1414.27	202
D121-11c	120	11	1378.0127	**26.40**	1805.44	**229**	26.15	**1815.74**	215	15.94	1808.95	187
D151-14b	150	14	1255.8763	**2.17**	1801.88	**79**	**2.17**	**1822.98**	76	**2.17**	1801.59	72
D151-14c	150	14	1243.0783	**1.61**	1813.54	**74**	**1.61**	1820.27	70	**1.61**	**1827.93**	70
D200-18b	198	18	1672.0877	**11.31**	**1826.59**	**55**	9.92	1821.08	50	9.87	1799.36	50
D200-18c	198	18	1382.1755	**0.12**	1799.44	**29**	**0.12**	**1874.69**	28	**0.12**	1793.24	27
D201-05k	200	5	8479.8618	1.96	1810.20	72	**7.49**	**1857.30**	**85**	3.88	1817.83	73
D241-10k	240	10	5690.8845	**0.22**	1839.57	**30**	**0.22**	**1891.71**	**30**	**0.22**	1875.31	29
100vrpnc12	48	4	948.8734	17.55	1422.84	205	**17.58**	1801.32	**245**	**17.58**	**1808.29**	227
100vrpnc3	74	7	845.9476	4.13	**1384.96**	**206**	10.09	992.90	157	**11.47**	1344.37	203
120vrpnc11	98	9	1527.1633	26.01	**1816.05**	166	26.01	1813.61	158	**35.64**	1814.67	**214**
150vrpnc4	98	9	1355.0136	**11.03**	**1817.82**	**97**	10.62	1809.13	78	10.55	1805.93	74
199vrpnc5	118	11	1644.7277	4.82	1817.42	24	4.91	**1859.61**	33	**6.77**	1807.28	**36**
50vrpnc1	148	14	572.1237	0.00	36.22	**54**	0.00	37.22	**54**	0.00	**38.29**	**54**
75vrpnc2	198	19	682.4024	**7.49**	342.40	147	**7.49**	352.95	147	5.27	**518.38**	**191**
Promedio				**8.33**	1691.64	**157**	8.07	1674.45	145	8.22	**1698.76**	151

Figura A.4.: Resultados obtenidos con el Algoritmo 5.1 utilizando diferentes valores de N_{max}.

Instancia			Sol. Inicial	$\mathcal{B}$			α_1			α_2			α_3			α_4		
Nombre	2n	k		% Mejora	Av. ET	AVI	% Mejora	Av. ET	AVI	% Mejora	Av. ET	AVI	% Mejora	Av. ET	AVI	% Mejora	Av. ET	AVI
D022-04g	20	4	285.6037	0.00	0.75	**26**	0.00	0.73	**26**	0.00	0.65	**26**	0.00	0.63	**26**	0.00	**1.21**	**26**
D023-03g	22	3	552.5039	**1.47**	0.78	**29**	**1.47**	0.76	**29**	**1.47**	0.71	**29**	**1.47**	0.71	**29**	**1.47**	**1.21**	**29**
D030-03g	28	3	495.047	**0.16**	2.45	35	**0.16**	2.47	35	**0.16**	2.24	**36**	**0.16**	2.19	**36**	**0.16**	**3.71**	35
D033-04g	32	4	496.1447	0.00	4.45	**38**	0.00	4.37	**38**	0.00	3.81	**38**	0.00	3.81	**38**	0.00	**7.51**	**38**
D051-06c	50	6	539.4449	0.00	22.12	**56**	0.00	22.19	**56**	0.00	19.86	**56**	0.00	19.31	**56**	0.00	**38.71**	**56**
D076-11c	74	11	713.839	9.52	171.97	132	8.96	175.00	132	**10.03**	168.39	**173**	**10.03**	166.61	**173**	9.11	**308.72**	170
D101-09c	100	9	1069.5505	9.72	865.78	276	**12.88**	963.83	321	9.86	483.08	289	11.10	524.01	316	9.75	**1019.62**	**342**
D101-11c	100	11	1163.0662	31.73	754.39	300	31.29	1001.37	298	27.78	638.04	355	27.87	725.68	366	**34.08**	**1392.71**	**393**
D121-11c	120	11	1378.0127	24.52	**1799.48**	387	**31.11**	1660.77	380	20.31	993.47	405	18.14	1103.08	**417**	20.13	1493.75	352
D151-14b	150	14	1255.8763	8.57	1803.41	153	7.14	**1805.67**	148	**8.94**	1801.37	329	8.89	1803.85	**336**	7.39	1802.36	254
D151-14c	150	14	1243.0783	**3.38**	1802.79	153	1.96	**1808.54**	147	1.74	1784.88	299	1.74	1777.07	**304**	1.77	1803.35	226
D200-18b	198	18	1672.0877	5.59	1814.90	54	4.85	1810.65	52	**6.11**	1814.21	96	6.07	1807.66	**98**	5.46	**1815.62**	81
D200-18c	198	18	1382.1755	0.20	1803.81	61	0.20	**1812.26**	56	**1.01**	1804.52	114	1.00	1801.32	**116**	0.50	1801.04	88
D201-05k	200	5	8479.8618	4.20	1807.79	**107**	4.31	1808.37	90	**5.66**	1806.59	95	4.35	1802.10	106	4.35	**1808.82**	65
D241-10k	240	10	5690.8845	0.90	1809.88	**47**	**1.77**	1814.84	38	1.15	1826.47	43	1.15	1824.55	43	1.41	**1832.87**	25
100vrpnc12	48	4	948.8734	**20.02**	587.88	232	17.89	579.21	265	17.96	587.92	**301**	14.83	562.32	267	15.33	**1061.23**	266
100vrpnc3	74	7	845.9476	6.51	**751.96**	199	**6.61**	744.10	197	5.97	375.34	257	5.97	414.87	**276**	6.10	654.36	261
120vrpnc11	98	9	1527.1633	20.39	1436.94	228	20.09	1010.48	196	19.88	586.93	215	22.15	914.28	327	**29.43**	**1525.99**	**329**
150vrpnc4	98	9	1355.0136	8.99	1801.13	147	8.37	1800.89	149	**12.40**	**1803.60**	318	11.48	1746.27	**336**	9.97	1801.20	274
199vrpnc5	118	11	1644.7277	9.67	**1815.58**	48	9.46	1813.52	50	8.95	1810.64	91	**9.80**	1807.48	**92**	9.77	1814.40	81
50vrpnc1	148	14	572.1237	2.64	35.79	101	3.21	42.00	**114**	2.64	31.81	108	2.64	22.80	90	**3.95**	**49.10**	97
75vrpnc2	198	19	682.4024	5.04	151.49	153	**5.48**	147.61	147	3.88	125.04	151	3.88	115.36	146	4.37	**251.02**	**162**
Promedio				7.87	956.62	134	**8.06**	946.80	135	7.54	839.53	174	7.40	861.18	**182**	7.93	**1013.11**	166

Figura A.5.: Comparación global con $t = 30$ mins. de las versiones $r\mathcal{TT}$ donde:
α_1=$\{r\mathcal{TT}vi$,$L_{max} = 3$, $N_{min} = 1$,$DRC_{max}\}$
α_2=$\{r\mathcal{TT}vi$,$\mathcal{SGF}$,$L_{max} = 3$, $N_{min} = 1$,$DRC_{max}\}$
α_3=$\{r\mathcal{TT}pr$,$\mathcal{SGF}$,$L_{max} = 3$, $N_{min} = 1$,$DRC_{max}\}$
α_4=$\{r\mathcal{TT}pr$,$L_{max} = 5$, $N_{min} = 1$,$DRC_{max}\}$

Instancia			Sol. Inicial	$\mathcal{B}$			β_1			β_2			β_3			β_4			β_5		
Nombre	2n	k		% Mejora	Av. ET	AVI	% Mejora	Av. ET	AVI	% Mejora	Av. ET	AVI	% Mejora	Av. ET	AVI	% Mejora	Av. ET	AVI	% Mejora	Av. ET	AVI
D022-04g	20	4	285.6037	0.00	0.75	**26**	0.00	0.69	**26**	0.00	1.10	**26**	0.00	**1.53**	**26**	0.00	0.69	**26**	0.00	0.75	**26**
D023-03g	22	3	552.5039	**1.47**	0.78	**29**	**1.47**	0.73	29	**1.47**	1.24	29	**1.47**	**1.56**	**29**	**1.47**	0.74	**29**	**1.47**	0.77	**29**
D030-03g	28	3	495.047	**0.16**	2.45	35	**0.16**	2.31	35	**0.16**	4.00	35	**0.16**	**4.87**	35	**0.16**	2.33	**36**	**0.16**	2.46	35
D033-04g	32	4	496.1447	0.00	4.45	**38**	0.00	3.96	35	0.00	8.08	**38**	0.00	**8.53**	**38**	0.00	4.00	**38**	0.00	4.44	**38**
D051-06c	50	6	539.4449	0.00	22.12	**56**	0.00	19.61	50	0.00	38.18	**56**	0.00	**43.62**	**56**	0.00	20.59	**56**	0.00	22.44	**56**
D076-11c	74	11	713.839	9.52	171.97	132	**10.03**	169.78	136	8.47	247.38	141	8.93	**378.72**	153	**10.03**	173.46	**173**	8.96	175.86	132
D101-09c	100	9	1069.5505	9.72	865.78	276	8.78	568.89	255	8.46	826.63	269	**15.21**	**1545.80**	**324**	7.58	520.51	294	13.14	1048.61	299
D101-11c	100	11	1163.0662	31.73	754.39	300	**38.42**	847.45	362	35.33	1309.06	341	35.65	**1575.02**	333	35.53	999.07	**486**	31.13	799.62	288
D121-11c	120	11	1378.0127	24.52	1799.48	387	24.53	1202.48	368	**33.53**	1604.56	**508**	17.73	**1805.37**	285	21.10	1388.71	499	22.85	1803.33	382
D151-14b	150	14	1255.8763	8.57	1803.41	153	**9.30**	1801.57	271	7.94	1800.46	258	5.15	1803.85	127	9.07	**1804.02**	**324**	8.67	1802.48	146
D151-14c	150	14	1243.0783	**3.38**	1802.79	153	1.74	1781.87	243	1.77	1809.94	228	3.24	**1810.20**	138	1.73	1792.93	**293**	3.21	1807.12	146
D200-18b	198	18	1672.0877	**5.59**	1814.90	54	4.54	1804.96	80	4.90	1805.98	79	2.41	**1818.73**	56	5.33	1806.35	**93**	4.14	1810.53	51
D200-18c	198	18	1382.1755	0.20	1803.81	61	**0.97**	1804.12	96	0.50	1807.03	89	0.47	1808.01	54	**0.97**	1800.77	**111**	0.20	**1810.11**	56
D201-05k	200	5	8479.8618	4.20	1807.79	**107**	4.82	1806.38	76	3.97	1813.71	61	**4.89**	**1823.08**	60	4.88	1801.79	92	4.10	1805.47	85
D241-10k	240	10	5690.8845	0.90	1809.88	**47**	1.11	1829.71	37	0.39	1819.60	26	1.09	**1850.93**	24	1.08	1822.30	41	**1.59**	1807.91	37
100vrpnc12	48	4	948.8734	**20.02**	587.88	232	14.84	599.69	277	17.90	1214.19	**303**	14.82	**1345.63**	221	14.84	604.88	273	19.98	583.41	231
100vrpnc3	74	7	845.9476	6.51	751.96	199	5.97	423.66	**276**	6.10	637.86	261	**7.20**	**1276.04**	262	5.97	381.72	257	6.51	742.45	199
120vrpnc11	98	9	1527.1633	20.39	1436.94	228	22.16	911.88	**314**	23.59	1336.24	281	**29.82**	**1776.27**	245	22.10	769.37	247	18.11	1290.75	203
150vrpnc4	98	9	1355.0136	8.99	1801.13	147	9.36	1756.23	**322**	9.42	1805.23	262	8.35	**1811.32**	157	9.54	1803.41	315	**10.20**	1809.07	149
199vrpnc5	118	11	1644.7277	9.67	1815.58	48	9.10	1808.31	**89**	**10.04**	1809.67	78	8.25	1812.72	66	6.47	1804.85	**89**	9.06	**1821.18**	49
50vrpnc1	148	14	572.1237	2.64	35.79	101	2.64	23.41	90	**4.53**	43.94	88	2.97	**52.67**	91	2.64	32.33	108	3.21	41.82	**114**
75vrpnc2	198	19	682.4024	5.04	151.49	153	3.88	117.97	146	4.37	246.23	**162**	4.82	**333.56**	158	3.88	126.49	151	**5.48**	155.36	153
Promedio				7.87	956.62	134	7.90	876.62	164	**8.31**	999.56	164	7.85	**1122.18**	133	7.47	884.61	**183**	7.83	961.18	132

Figura A.6.: Comparación global con $t = 30$ mins. de las versiones $r\mathcal{OR}$ donde:
β_1=$\{r\mathcal{OR}pr$,$L_{max} = 3$, $N_{min} = 1$,$DRC_{max}\}$
β_2=$\{r\mathcal{OR}pr$,$\mathcal{SGF}$,$L_{max} = 5$, $N_{min} = 1$,$DRC_{max}\}$
β_3=$\{r\mathcal{OR}pr$,$\mathcal{SGF}$,$L_{max} = 3$, $N_{min} = 5$,$DRC_{max}\}$
β_4=$\{r\mathcal{OR}vi$,$\mathcal{SGF}$,$L_{max} = 5$, $N_{min} = 1$,$DRC_{max}\}$
β_5=$\{r\mathcal{OR}vi$,$L_{max} = 5$, $N_{min} = 1$,$DRC_{max}\}$

Bibliografía

[1] T. Achterberg. *Constraint Integer Programming.* PhD thesis, 2007.

[2] T. Achterberg. SCIP: solving constraint integer programs. *Mathematical Programming Computation*, 1(1):1–41, Jan. 2009.

[3] T. Achterberg and T. Berthold. Improving the feasibility pump. *Discrete Optimization*, 4(1):77–86, Mar. 2007.

[4] P. Avella, B. D'Auria, and S. Salerno. A LP-based heuristic for a time-constrained routing problem. *European Journal of Operational Research*, 173(1):120–124, Aug. 2006.

[5] M. Balinski and R. Quandt. On an integer program for a delivery problem. *Operations Research*, (April 2014), 1964.

[6] G. Berbeglia, J.-F. Cordeau, I. Gribkovskaia, and G. Laporte. Static pickup and delivery problems: a classification scheme and survey. *Top*, 15(1):1–31, Apr. 2007.

[7] L. Bertacco, M. Fischetti, and A. Lodi. A Feasibility Pump heuristic for general Mixed-Integer Problems. (ii):1–23, 2005.

[8] E. Bixby, M. Fenelon, and Z. Gu. MIP: Theory and practice—closing the gap. *System Modelling and Optimization: Methods, Theory, and Applications*, pages 1–31, 2000.

[9] R. E. Bixby and E. Rothberg. Progress in computational mixed integer programming - a look back from the other side of the tipping point. *Annals OR*, pages 37–41, 2007.

[10] R. Burkard, M. Dell'Amico, and S. Martello. *Assignment Problems.* Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2009.

[11] D. Chen, R. Batson, and Y. Dang. Applied integer programming: modeling and solution. page 8, 2011. Successful Integer Programming Applications.

[12] N. Christofides and S. Eilon. Expected distances in distribution problems. *OR*, 1969.

[13] N. Christofides, A. Mingozzi, and P. Toth. *Combinatorial Optimization.* 1979.

[14] V. Chvátal. *Linear Programming.* Series of books in the mathematical sciences. W. H. Freeman, 1983.

[15] J.-F. Cordeau, G. Desaulniers, J. Desrosiers, M. M. Solomon, and F. Soumis. The vehicle routing problem. chapter VRP with Time Windows, pages 157–193. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2001.

[16] E. Danna, E. Rothberg, and C. L. Pape. Exploring relaxation induced neighborhoods to improve MIP solutions. *Mathematical Programming*, 102(1):71–90, May 2004.

[17] G. Dantzig and J. Ramser. The truck dispatching problem. *Management science*, 1959.

[18] R. De Franceschi, M. Fischetti, and P. Toth. A new ILP-based refinement heuristic for vehicle routing problems. *Mathematical Programming*, 105(2-3):471–499, Nov. 2006.

[19] S. Eilon, N. Christofides, and C. D. T. Watson-Gandy. *Distribution management: mathematical modelling and practical analysis*. 1971.

[20] Fair Isaac Corporation. FICO Xpress Optimization Suite. `http://www.fico.com/en/Products/DMTools/Pages/FICO-Xpress-Optimization-Suite.aspx`.

[21] D. Feillet and M. Gendreau. A note on branch-and-cut-and-price. *Operations Research Letters*, 38(5):346–353, 2010.

[22] M. Fischetti, F. Glover, and A. Lodi. The feasibility pump. *Mathematical Programming*, pages 1–17, 2005.

[23] M. Fischetti and A. Lodi. Local branching. *Mathematical Programming*, 98(1-3):23–47, 2003.

[24] M. Fisher and R. Jaikumar. A generalized assignment heuristic for vehicle routing. *Networks*, 11(2):109–124, Jan. 1981.

[25] M. Gary and D. Johnson. Computers and Intractability: A Guide to the Theory of NP-completeness. *The Journal of Symbolic Logic*, 48(2):498, June 1979.

[26] B. Gillett and L. Miller. A heuristic algorithm for the vehicle-dispatch problem. *Operations research*, (February 2014), 1974.

[27] B. Golden, S. Raghavan, and E. Wasil. *The vehicle routing problem : latest advances and new challenges*, volume 43 of *Operations Research/Computer Science Interfaces*. Springer US, Boston, MA, 2008.

[28] M. Grötschel, L. Lovász, and A. Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1981.

[29] M. Grötschel and M. Padberg. *The sharpest cut: the impact of Manfred Padberg and his work*. 2004.

[30] Gurobi Optimization. The Gurobi Optimizer. `http://www.gurobi.com`.

[31] G. Gutin. On an approach to solving the traveling salesman problem. In *Proc. The USSR Conference on System Research*, pages 184–185. 1984.

[32] G. Gutin, A. Punnen, A. Barvinok, E. K. Gimadi, and A. I. Serdyukov. The traveling salesman problem and its variations, 2002.

[33] G. Gutin, A. Yeo, and A. Zverovitch. Exponential neighborhoods and domination analysis for the TSP. *The Traveling Salesman Problem and Its Variations*, 2004.

[34] P. Hansen, N. Mladenović, and J. a. Moreno Pérez. Variable neighbourhood search: methods and applications. *Annals of Operations Research*, 175(1):367–407, Oct. 2009.

[35] P. Hansen, N. Mladenović, and D. Urošević. Variable neighborhood search and local branching. *Computers & Operations Research*, 33(10):3034 – 3045, 2006. Part Special Issue: Constraint Programming.

[36] IBM. ILOG CPLEX: High-performance mathematical programming engine. `http://www.ibm.com/software/integration/optimization/cplex/`.

[37] E. Johnson. Progress in linear programming-based algorithms for integer programming: An exposition. *INFORMS Journal on Computing*, 12(1):2–23, 2000.

[38] M. Jünger, T. Liebling, and D. Naddef. *50 Years of Integer Programming 1958–2008*. 2010.

[39] N. Karmarkar. A new polynomial-time algorithm for linear programming. *Combinatorica*, 4:373–395, December 1984.

[40] A. H. Land and A. G. Doig. An automatic method of solving discrete programming problems. *Econometrica*, 28(3):497–520, 1960.

[41] J. K. Lenstra and A. H. G. R. Kan. Complexity of vehicle routing and scheduling problems. *Networks*, 11(2):221–227, 1981.

[42] A. Lodi. Mixed integer programming computation. *50 Years of Integer Programming 1958-2008*, pages 619–645, 2010.

[43] S. Martello and P. Toth. *Knapsack Problems: Algorithms and Computer Implementations*. John Wiley & Sons, Inc., New York, NY, USA, 1990.

[44] I. Muter, I. Birbil, and K. Bülbül. Simultaneous column-and-row generation for large-scale linear programs with column-dependent-rows. *Mathematical Programming*, June 2012.

[45] I. Muter, I. Birbil, K. Bülbül, and Güvenç. A note on "A LP-based heuristic for a time-constrained routing problem". *European Journal of Operational Research*, 221(2):306–307, Sept. 2012.

[46] Z. Naji-Azimi, M. Salari, and P. Toth. An Integer Linear Programming based heuristic for the Capacitated m-Ring-Star Problem. *European Journal of Operational Research*, 217(1):17–25, Feb. 2012.

[47] W. P. Nanry and J. Wesley Barnes. Solving the pickup and delivery problem with time windows using reactive tabu search. *Transportation Research Part B: Methodological*, 34(2):107–121, Feb. 2000.

[48] G. L. Nemhauser and L. A. Wolsey. *Integer and combinatorial optimization*. Wiley interscience series in discrete mathematics and optimization. Wiley, 1988.

[49] C. Papadimitriou and K. Steiglitz. Combinatorial optimization: algorithms and complexity. 32(6):1258–1259, Dec. 1998.

[50] A. P. Punnen. The Traveling Salesman Problem: new polynomial approximation and domination analysis. *J. Information Optim. Sci. 22*, pages 191–206, 2001.

[51] C. Rego and F. Glover. Chapter 8 Local Search and Metaheuristics. 2007.

[52] M. Salari, P. Toth, and A. Tramontani. An ILP improvement procedure for the Open Vehicle Routing Problem. *Computers & Operations Research*, 37(12):2106–2120, Dec. 2010.

[53] V. Sarvanov and N. Doroshko. The approximate solution of the travelling salesman problem by a local algorithm with scanning neighborhoods of factorial cardinality in cubic time. In *Software: Algorithms and Programs*, volume 31, pages 11–13. 1981.

[54] M. Savelsbergh. A branch-and-price algorithm for the generalized assignment problem. *Operations Research*, 1997.

[55] P. Toth and A. Tramontani. An integer linear programming local search for capacitated vehicle routing problems. *The Vehicle Routing Problem: Latest Advances and New Challenges*, 43, 2008.

[56] P. Toth and D. Vigo. *The vehicle routing problem.* 2002.

[57] P. Toth and D. Vigo. The Granular Tabu Search and Its Application to the Vehicle-Routing Problem. *INFORMS Journal on Computing*, 15(4):333–346, Nov. 2003.

[58] J. Vielma. Mixed Integer Linear Programming Formulation Techniques. pages 1–48, 2013.

[59] VRPLIB. http://www.or.deis.unibo.it/research_pages/ORinstances/VRPLIB/DCVRP.zip.

[60] G. Wang and L. Tang. A row-and-column generation method to a batch machine scheduling problem. *Proceedings of The Ninth International Symposium on Operations Research and Its Applications*, pages 301–308, 2010.

[61] E. J. Zak. Row and column generation technique for a multistage cutting stock problem. *Computers & Operations Research*, 29(9):1143–1156, Aug. 2002.

[62] Zuse Institute Berlin. SCIP: Solving Constraint Integer Programs. `http://scip.zib.de/`.