Universidad de Buenos Aires

Facultad de Ciencias Exactas y Naturales

Departamento de Computación

Un Algoritmo de Búsqueda Local Basado en Programación Lineal Entera Aplicado al Problema de Ruteo de Vehículos Multiperíodo

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

Santiago Aboy Solanes

Directores : Dr. Juan José Miranda Bront y Lic. Agustín Ismael Montero

Buenos Aires, 2018

UN ALGORITMO DE BÚSQUEDA LOCAL BASADO EN PROGRAMACIÓN LINEAL ENTERA APLICADO AL PROBLEMA DE RUTEO DE VEHÍCULOS MULTIPERÍODO

Los Problemas de Ruteo de Vehículos (VRP, por su sigla en inglés) aparecen en la organización de las tareas de distribución de mercadería o personal, planificación de recorridos en robótica móvil o prestación de servicios a un conjunto de clientes mediante una flota de vehículos. Los vehículos realizan sus movimientos a través de una red partiendo de puntos fijos, llamados depósitos. Cada tramo entre dos clientes de esta red tiene asociado un costo y/o tiempo de viaje que puede depender de muchos factores, como por ejemplo del tipo de vehículo o del período durante el cual el tramo es recorrido. Este tipo de problemas es de gran relevancia en empresas de tamaño pequeño a grande, tanto en el sector público como privado. Estos problemas suelen ser $\mathcal{NP}$-Hard y desde el punto de vista del modelado y resolución, las características de cada aplicación conllevan un desafío particular. Una excelente presentación sobre diferentes variantes de estos problemas puede verse en Toth & Vigo [16].

Por cuestiones prácticas, el abordaje que se hace de los problemas de ruteo puede considerar las restricciones de forma simplificada. En muchas aplicaciones, la planificación suele limitarse a determinar cómo realizar la distribución diaria. Estos enfoques han producido significativas mejoras tanto en términos de costos como en la calidad del servicio. Sin embargo, debido a los avances obtenidos en términos algorítmicos y su consiguiente impacto en la práctica, actualmente la tendencia tanto en investigación como en desarrollo es a abordar problemas más complejos. Algunas variantes con creciente interés por parte de la comunidad científica consideran agrupar decisiones usualmente tomadas en etapas separadas (e.g., planificación de más de un día, combinación con asignación de tripulaciones) así como también incorporar restricciones cada vez más realistas (e.g., orden de la carga dentro del camión, balance respecto del eje central por cuestiones de seguridad).

En esta dirección, recientemente se propuso en el marco del *VeRoLog Solver Challenge 2017* (VSC2017) una variante proveniente de una aplicación real en la distribución de herramientas para la medición de calidad en la industria lechera. Partiendo de la demanda de clientes por determinado tipo de herramientas, el problema consiste en resolver de manera integrada la planificación de la utilización de las herramientas en un horizonte de tiempo, medido en días, y la distribución de las mismas incorporando la logística necesaria para el ruteo de las mismas en cada día.

El problema planteado presenta características similares al VRP con Ventanas de Tiempo (VRPTW), junto con características del clásico VRP con *Pickup y Delivery* (VRPPD) y VRP con capacidades (CVRP). Se tiene la posibilidad que una herramienta sea trasladada de un cliente que finaliza a otro que inicia, permitiendo reducir el número de herramientas totales necesarias. Una descripción detallada del problema y las restricciones particulares puede consultarse en el sitio de *VeRoLog* [13], donde también se encuentran disponibles

instancias realistas de distintos tamaños, llegando hasta 2500 clientes y un horizonte de tiempo de 75 días en las instancias de mayor tamaño.

En esta tesis se proponen algoritmos de búsqueda local, utilizando un modelo de reubicación basado en técnicas de programación lineal entera (PLE). Un ejemplo de este esquema para el caso de ruteo de vehículos en períodos simples puede verse en Montero et al. [11]. Se utiliza el paradigma de destrucción/reparación en donde un conjunto de nodos es removido de las rutas y reinsertado a través de la resolución del modelo. Los experimentos realizados muestran que el enfoque utilizado es capaz de mejorar las mejores soluciones obtenidas en la competencia. Además, dicho enfoque es capaz de ser extendido para ser utilizado en diversos contextos.

Palabras claves: VRP multi-período con Recolección y Entrega y ventanas de tiempo, Programación Lineal Entera, Ruteo de Vehículos

AN ILP-BASED LOCAL SEARCH ALGORITHM APPLIED TO A THE MULTI-PERIOD VEHICLE ROUTING PROBLEM

Vehicle Routing Problems (VRP) appear in the planning of personnel or goods distribution tasks, mobile robotics routing, or the service delivery to a set of clients using vehicle fleets. Vehicles move through a network from specific points, called depots. Each segment between two clients in the network has a cost and/or a travel time associated to it, which can depend on many factors such as the vehicle type or the period in which the segment is traversed. This type of problems is important to small and big companies, in both the private and public sector. They tend to be $\mathcal{NP}$-Hard and from the modelling and resolution's point of view, its characteristics usually bring a particular challenge. An excelent presentation of different variants of this type of problems can be seen in Toth & Vigo [16].

For practical reasons, the problem's constraints can be considered in a simplified fashion. In many applications, the planning is limited to a single day. These approaches brought many significant improvements in terms of cost and quality of service. However, due to the algorithmic advances and their impact, the current trend in both industry and academia is to face problems with greater complexity. Some increasingly important variants consider grouping tasks that are usually atomic (e.g., the planning of a horizon of days, combining fleet assignment) as well as adding more realistic restrictions (e.g., load ordering inside a vehicle, axle balance due to safety concerns).

In this vein, recently proposed in the frame of the *VeRoLog Solver Challenge 2017* (VSC2017), comes a variant from a real application in the milk industry of the tool's distribution used to measure the quality of the milk. Each farm is a customer which demands special measuring tools, and those have to be delivered to the customers at their request. After the measurement, the tools have to be picked up again. The problem at hand consists in addressing the design of both the use of the necessary tools in a planning horizon, and the tool's distribution with its corresponding routing within each day.

The presented problem has similar characteristics to VRP with Time Windows (VRPTW), as well as characteristics of the VRP with Pickup and Delivery (VRPPD) and Capacitated VRP (CVRP). An important note is the fact that tools can be transported directly from one customer to another, without going to the depot. This allows to reduce the total amount of tools used in exchange. The detailed problem's description and its restrictions can be found in *VeRoLog* [13]'s site. It also provides realistic instances of different sizes, with an upper bound of 2500 customers and a planning horizon of 75 days.

This thesis proposes a solution using a local search algorithm, based on Integer Linear Programming (ILP) techniques. An example of a similar schema to the vehicle routing problem in simple periods can be seen in Montero et al. [11]. The destroy/repair paradigm is used, in which a set of nodes is removed from the routes and reinserted through solving the ILP model. Based on the obtained results, the approach is able to improve the

competition's best solutions. Furthermore, the approach is flexible enough to able to be adapted to new restrictions.

Keyword: Multiperiod VRP with Pickups and Deliveries and Time Windows, Integer Linear Programming, Vehicle Routing

A mi familia y amigos.

A Hara.

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, la investigación operativa y teoría algorítmica, en donde se buscan 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), se encuentran las áreas de logística, el 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, o clientes.

En el año 1959, Dantzig & Ramser [3] proponen en *The truck dispatching problem* "encontrar el ruteo óptimo 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 Sección 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".

El trabajo de Dantzig & Ramser [3] se desarrolla enormemente con muchas aplicaciones prácticas reales. Se tienen una gran cantidad de estudios sobre VRPs, desde trabajos como Christofides et al. [7] en la década del 70, hasta otros más recientes como Toth & Vigo [15], y Golden et al. [14].

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.

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 siendo el de minimizar el tiempo total de viaje.

Las principales características de los 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. Por ejemplo, se puede tener que un único vehículo debe visitar a los clientes, la entrega debe completarse en una sola visita, o *split delivery* donde 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. A continuación, se introducen las restricciones particulares que aplican al problema de esta tesis.

1.2. Restricciones Particulares

En el marco de esta tesis, el problema abordado consiste en un problema de planificación multi-período sobre un horizonte de días finito. En estos días, se tienen una cierta cantidad de pedidos a satisfacer. Cada pedido tiene que realizar la *entrega* de herramientas, así como también la *recolección* de las mismas (*delivery* y *pickup* en inglés). Cada uno de estos pedidos tiene asociada una cierta cantidad de días en que usa dichas herramientas.

La entrega tiene una ventana de tiempo en la cual debe ser satisfecha, y las herramientas deben ser recolectadas el día siguiente en que el cliente deja de hacer uso de las mismas.

Hay varios tipos de herramientas, y un cliente puede necesitar más de un tipo de ellas. En el caso que suceda esto, el cliente realiza un pedido por cada tipo. Las herramientas asociadas a un pedido son entregadas por un mismo camión en un solo momento. Asimismo, son recolectadas por un solo camión en un solo momento.

Adicionalmente, cada camión puede visitar al mismo cliente para entregar o recolectar solamente herramientas que fueron pedidas. Es decir, no se pueden utilizar a los clientes como depósitos. Las herramientas que utiliza un camión, no pueden ser intercambiadas con otro camión dentro del mismo día.

En contraposición de los problemas de planificación de un día se tiene el problema adicional de decidir en qué día va a ser satisfecho cada pedido. Este no es un problema menor, y el hecho de asignar pedidos a días no puede ser enfrentado por separado del ruteo de cada día. En caso de hacerlo, puede suceder que se decida una cierta asignación válida de pedidos a días pero que luego el ruteo de algún día sea imposible dado a restricciones por distancia, stock, etc. Por esto, los días están interconectados y deben ser considerados como un todo.

El problema se formaliza en la Sección 3.

1.3. Objetivo y contribuciones de la tesis

El VRP y sus variantes son problemas $\mathcal{NP}$-Hard, ampliamente estudiados en la literatura. Para resolverlos, se utilizan enfoques exactos donde el objetivo es encontrar la solución óptima del problema, así como también heurísticos donde se pretende encontrar soluciones de buena calidad sin asegurar de haber llegado al óptimo. Por lo general, se utilizan estos últimos en instancias de gran escala, o en contexto donde el tiempo de resolución es acotado.

En esta tesis se hace hincapié en el refinamiento de una solución factible de buena calidad, y no en la construcción de una solución inicial. Con el fin de mostrar resultados concretos, se utilizará un conjunto particular detallado en la Sección 6.1 como punto de partida. Cabe aclarar que el trabajo presentado también puede ser aplicado a soluciones obtenidas por otro método.

Para lograr esta mejora, se plantea utilizar operadores de búsqueda local. En este contexto, se plantea el conjunto de soluciones como un gran grafo donde las soluciones son los nodos y están conectadas entre sí si son parecidas, para algún criterio establecido. De esta forma, permite navegar el vecindario de soluciones teniendo el propósito de ir hacia soluciones mejores, buscando los óptimos locales.

También, se modela el problema con un modelo de reubicación que utiliza técnicas de programación lineal entera. Una solución similar fue planteada tesis de Agustín Montero [10], el cual es tomado como punto de partida para la creación del modelo.

Dicho modelo se basa en el paradigma de destrucción/reparación. Se eligen nodos a remover que se remueven construyendo una solución restringida. Con los nodos removidos se construyen secuencias de uno o más nodos, que se reinsertan en la solución restringida con el fin de obtener una solución mejor.

En la tesis de Agustín Montero la reubicación de los nodos en la solución se hace a través de la resolución de un modelo ILP. Dicho modelo tiene una cantidad exponencial de variables, por lo que se aplica el enfoque estándar de generación de columnas heurísticamente considerando los costos reducidos y resolviendo el modelo con un *solver* de propósito general (CPLEX). El algoritmo propuesto permite explorar el vecindario de soluciones de reinserción considerando secuencias de nodos de tamaño arbitrario, y su desempeño se estudia en detalle variando una gran cantidad de parámetros.

En esta tesis, incluso utilizando secuencias acotadas, en particular de tamaño 1, se obtuvieron resultados positivos logrando mejorar soluciones que ya eran de las mejores obtenidas por los concursantes del *VeRoLog Solver Challenge 2017* (VSC2017).

A su vez, se complejiza el modelo de Montero para adaptarlo al problema actual que presenta un horizonte de tiempo en vez de un día solo, y múltiples tipos de herramienta en vez de uno solo, entre otros cambios.

Asimismo, se desarrollan distintas variaciones que se le aplican al modelo de reubicación con el objetivo de lograr mejores resultados. Estas variaciones pueden aplicarse en forma unitaria, así como también combinándose entre sí. El modelo es explicado en detalle en la Sección 5, y sus variaciones en la Sección 5.5, donde luego se compararán las distintas combinaciones de variaciones entre sí para observar la mejor configuración de ellas.

También se van a comparar las mejoras del modelo contra la utilización de operadores de búsqueda local. Resulta interesante observar las mejoras de estos dos enfoques distintos. Al utilizar operadores locales, se debe explicitar la manera en que se van a utilizar los mismos. Por otro lado, el modelo de reubicación permite establecer las restricciones que deberán ser cumplidas, y abstraerse de exactamente cómo se recorren las posibles combinaciones.

2. PRELIMINARES

2.1. Programación Lineal

La Programación Lineal (PL) permite resolver problemas que pueden ser modelados para encontrar una valuación en $\mathbb{R}_+$ para un conjunto de variables x no negativas, de manera tal que se maximice (o minimice) una función objetivo lineal $z : \mathbb{R}^n \to \mathbb{R}$ y que se cumpla un conjunto de desigualdades lineales $Ax \leq b$ ($A \in \mathbb{R}_+^{m \times n}, b \in \mathbb{R}_+^m$). Todo problema de PL puede escribirse de la siguiente forma:

$$\begin{array}{ll} \textbf{min} & z(x) = cx \\ \textbf{sujeto a} & \\ & Ax \leq b \\ & x_i \geq 0 \quad \forall i = 1 \dots n \end{array}$$

A su vez, esto tiene una interpretación geométrica que se basa en entender al grupo de restricciones $Ax \leq b$ como rectas que definen a un poliedro en $\mathbb{R}_+^n$. Por ejemplo, en la Figura 2.1 se grafica el siguiente Modelo PL:

$$\begin{array}{ll} \textbf{min} & z(x,y) = x + y \\ \textbf{sujeto a} & \\ & \frac{1}{3}x - y \leq -2 \\ & 2x - y \leq 5{,}5 \\ & \frac{1}{2}x + y \geq 2 \\ & 2x - y \geq 0{,}5 \\ & x, y \in \mathbb{R}_+ \end{array} \tag{2.1}$$

Existen algoritmos que encuentran una solución óptima para cualquier Modelo PL en tiempo polinomial (e.g. ver Karmarkar [9]). Además, existen otros algoritmos que si bien no son polinomiales funcionan bien en la práctica como es el caso de Simplex, algoritmo propuesto por Dantzig [2]. Hoy en día existe una variedad de productos que implementan estos algoritmos de manera eficiente tales como CPLEX, AMPL, Gurobi, etc.

2.2. Programación Lineal Entera Mixta

Hay una gran variedad de problemas, que restringen el dominio de las variables a los enteros ($\mathbb{Z}$). Los llamados Modelos de Programación Lineal Entera Mixma (PLEM) tienen la siguiente estructura:

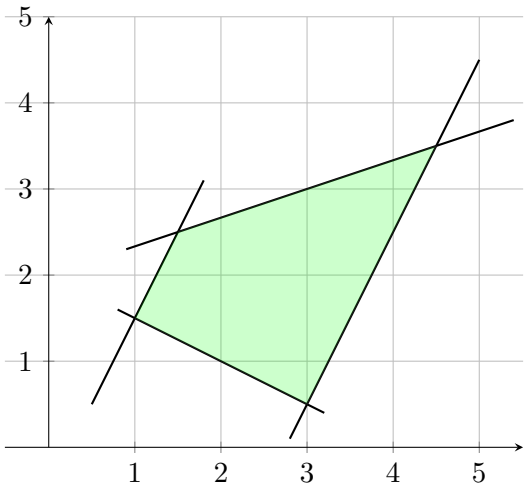


Fig. 2.1: Ejemplo de poliedro descripto por sistema de ecuaciones PL 2.1.

$$\begin{array}{ll} \textbf{min} & z(x) = cx \\ \textbf{sujeto a} & \\ & Ax \leq b \\ & x_i \geq 0 \quad \forall i \in C \cup I \\ & x_i \in \mathbb{N} \quad \forall i \in I \end{array} \tag{2.2}$$

Donde $x = I \cup C$, siendo I las variables que solo pueden tomar valores enteros, y C aquellas que pueden tomar cualquier valor continuo. Si llamamos $P = \{x \in \mathbb{R}^n \mid Ax \leq b, x \geq 0\}$ al poliedro que describe el modelo, el conjunto de soluciones factibles es $S = \left\{x \in P \mid I \in \mathbb{N}^{|I|}\right\}$. Además, decimos que $\bar{x} = \text{mín}\{z(x) \mid x \in P\}$ es la relajación lineal de P. Y como $S \subseteq P$, entonces se puede afirmar que la solución óptima x^* de S cumple que $z(x^*) \geq z(\bar{x})$, es decir, la relajación del PLEM es una cota inferior de la solución óptima dado que estamos considerando un problema de minimización.

Desde el punto de vista geométrico, se llama $conv(S)$ a la cápsula convexa de S, o sea, el poliedro más chico que contiene a todos los puntos de S. Si $conv(S)$ se puede representar con un conjunto polinomial de restricciones, entonces se puede hallar la solución óptima utilizando cualquier algoritmo que resuelva PL. En la Figura 2.2 se puede ver la representación gráfica del PLEM de ejemplo que se presenta a continuación, donde P está de color verde y $conv(S)$ de color azul:

$$\begin{array}{ll} \textbf{min} & z(x,y) = x + y \\ \textbf{sujeto a} & \\ & \frac{1}{3}x - y \leq -2 \\ & 2x - y \leq 5{,}5 \\ & \frac{1}{2}x + y \geq 2 \\ & 2x - y \geq 0{,}5 \\ & x, y \in \mathbb{N} \end{array}$$

El problema de resolver un PLEM general pertenece a la clase NP-Hard, ya que varios

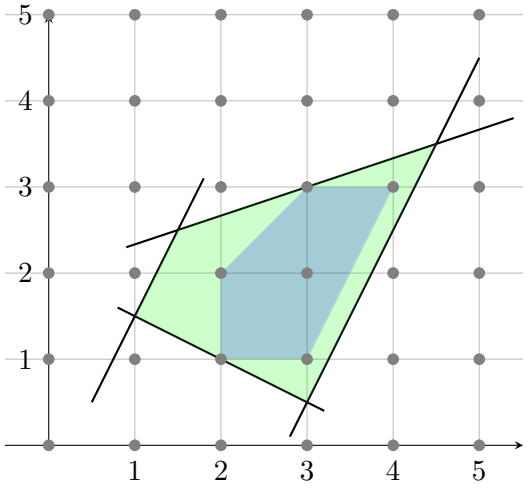


Fig. 2.2: Ejemplo de poliedro descripto por sistema de ecuaciones PLEM 2.2.

problemas que también están en NP-Hard se pueden modelar como PLEM, por ejemplo, el VRP mencionado anteriormente. Como mencionamos anteriormente, se puede probar que si $P = conv(S)$ y se tiene una cantidad polinomial de restricciones para representar a P, entonces se puede utilizar cualquier algoritmo de PL para encontrar la solución óptima de S. Sin embargo, no se sabe si encontrar la cápsula convexa de un conjunto cualquiera S es polinomial. A continuación, se presenta las técnicas más frecuentes para resolver PLEM.

2.3. Algoritmos para PLEMs

2.3.1. Branch&Bound (B&B)

La primera estrategia para resolver PLEM se basa en una utilización de los algoritmos de PL. Cuando se aplica uno de estos algoritmos sobre un PLEM, la estrategia más común es relajar las restricciones que restringen el dominio de las variables a los enteros. Es por esto que la solución óptima que arroja el algoritmo puede contener alguna variable entera tomando un valor fraccionario, lo cual, no es factible en el problema original. En estos casos lo que se hace es aplicar la táctica Divide & Conquer y se ramifica o divide el problema en dos subproblemas, de manera tal que la solución de la relajación no sea solución de ninguno de ellos pero a su vez no se pierda ninguna solución entera.

Comúnmente, se utiliza el denominado “Branching” por Variable, que consiste en tomar una variable binaria que tenga un valor fraccionario en la solución de la relajación y crear dos subproblemas: uno asignando la variable al valor 1, y la otra al 0. El criterio para elegir cual variable x tomar puede variar según cómo se va generando el árbol de ejecución, entre los más comunes están: x más cercana a 0.5, x más cercana a 0, y x más cercana a 1.

Por otro lado, para evitar explorar ramas del árbol que no lleven a una solución óptima, se aplica el denominado “Bounding”, que consiste en aplicar podas sobre el árbol basadas

en cotas del funcional. Generalmente, durante la ejecución del algoritmo se mantiene una solución x_{UB} (x_{LB}) que es la mejor encontrada hasta el momento, y la cuál sirve de cota superior (inferior), si el problema es de minimización (maximización), la cual permite estando en un nodo dejar de explorarlo si la relajación vale $z(x^*) > z(x_{UB})$ ($z(x^*) < z(x_{LB})$). Esta cota también suele llamarse "Cota Primal".

Otro de los parámetros que se puede ajustar es cómo se recorre el árbol de nodos que se va generando. Este puede ser recorrido de las maneras más convencionales como Depth First Search (DFS), lo cuál es particularmente conveniente para conseguir Cotas Primales de manera veloz y para no utilizar tanta memoria, o Best Bound First (BBF) que utiliza más memoria pero pretende minimizar el número de nodos explorados al elegir siempre el nodo con mejor "Cota Dual", o sea, valor de la relajación.

2.3.2. Cutting-Planes

Otra forma de abordar problemas con variables enteras o binarias, es mediante el agregado de *planos de corte*. Un plano de corte es una restricción que se cumple para toda solución en S, pero no se cumple para la solución óptima fraccionaria de la relajación del problema. Un esquema general que representa a los algoritmos de planos de corte se define en los siguientes pasos:

1. Resolver la relajación. Sea x^* el óptimo.
2. Si $x^* \in S$, terminar.
3. Encontrar $a'x \leq b'$ una restricción válida para todo $x \in S$ pero violada por x^* y agregarla a la formulación.
4. Volver al Paso 1.

Mediante esta técnica se busca ir acercándose a representar con restricciones lineales a $conv(S)$ ya que como se mencionó anteriormente, si se obtiene un PL que describa $conv(S)$ entonces va a tener una solución óptima que pertenece a S.

Una de las desventajas de esta técnica es la dificultad de encontrar planos de corte que se ajusten a la cápsula convexa de S, ya sea porque es computacionalmente ineficiente o por la dificultad teórica que esto implica. Es por esto que esta técnica no suele utilizarse por sí misma sino en conjunción con Branch&Bound.

2.3.3. Branch&Cut (B&C)

La técnica Branch&Cut es una combinación de las mencionadas previamente B&B y Cutting-Planes. Este algoritmo surge a partir de la necesidad de reducir la enumeración de nodos al utilizar B&B. Para lograr esto, la idea es aplicar una cierta cantidad de iteraciones de planos de corte en los nodos del árbol para así lograr ajustar su cota dual y por lo tanto poder aplicar Bounding de manera más frecuente.

Esta técnica es particularmente interesante en la etapa de experimentación porque su eficiencia depende de un compromiso entre la cantidad de iteraciones de planos de corte y la cantidad de nodos que se enumeran. Esto debe ser ajustado ya que las iteraciones de plano de corte pueden ser muy costosas y a partir de cierto número la enumeración resulta más eficiente.

Uno de los parámetros a ajustar es la cantidad de iteraciones de planos de corte que se llevan a cabo en cada uno de ellos y otro es la cantidad total de desigualdades se agregan al modelo, por ejemplo. Una estrategia que a veces se utiliza es realizar más iteraciones de plano de corte cuanto más cerca del nodo raíz se está. Esto en general ofrece un buen compromiso entre tiempo invertido en la separación de desigualdades y enumeración de nodos. En particular, cuando solamente se utilizan iteraciones de planos de corte en el nodo raíz, el algoritmo se conoce como Cut&Branch.

2.4. Heurística

En ciertos problemas donde la dificultad de resolverlos de manera exacta es muy elevada, a veces se puede hacer un compromiso y buscar encontrar una solución "buena", aunque no sea la mejor, a cambio de tener un tiempo de cómputo razonable. Este es exactamente el propósito de un algoritmo heurístico. Una observación válida es que los algoritmos heurísticos pueden no encontrar ninguna solución, o incluso estar en el óptimo y no saberlo.

2.4.1. Goloso

Hay distintos tipos de estrategias para hallar soluciones. Una de las más utilizadas por su simpleza y velocidad es la técnica llamada Goloso (o Greedy). Básicamente, se comienza con una solución vacía o trivial y se busca ir extendiéndola de a un paso a la vez buscando maximizar (o minimizar) algún criterio con decisiones locales que finalmente pueden no ser óptimas. Por ejemplo, en VRP se puede ir construyendo un camino partiendo desde el depósito e ir agregando nodos (un paso) al final de este si minimizan la duración parcial (criterio).

2.4.2. Búsqueda Local

Otra técnica un poco más compleja es la Búsqueda Local. La idea es plantear todo el conjunto de soluciones como un gran grafo donde las soluciones son los nodos y están conectadas entre sí si son parecidas, con algún criterio establecido. Una vez definido este grafo, se parte de una solución inicial ya conocida y se lo navega, con algún criterio, yendo a través de sus ejes hasta llegar a una solución satisfactoria.

Lo primero que se debe hacer es definir un criterio de vecindad que indica si dos soluciones son adyacentes o no en el grafo. Un ejemplo de este criterio: si las soluciones

fuesen cadenas de caracteres, es crear un eje entre ellas si difieren en un solo carácter. De este modo, por ejemplo, la solución “casa” y “cama” son adyacentes, pero “casa” y “pala” no.

Una de las cosas interesantes a la hora de definir la vecindad es tener en cuenta su tamaño. Si es muy grande, el grafo va a ser más denso y puede que el camino hacia una solución óptima sea más corto, ya que se necesitan pocos movimientos para hallar una buena solución. Sin embargo, el costo de computar tal vecindad puede ser muy grande, incluso no polinomial. Es por esto, que se busca en general equilibrar estas dos variables.

Otro criterio que se debe definir es cómo recorrer el grafo. Dos maneras muy populares son las llamadas First-accept y Best-accept.

- *First-accept:* Al recorrer el vecindario, si se encuentra una solución mejor que la actual se mueve inmediatamente hacia ella.
- *Best-accept:* Se mueve hacia la mejor solución vecina de la actual, si ésta es mejor.

A su vez, se debe definir los movimientos en sí mediante lo que se conoce como operadores. Los operadores son pasos a seguir para encontrar soluciones vecinas. Un operador, por ejemplo, en el caso de las cadenas de caracteres nombradas anteriormente es sustituir una letra por otra, de esta manera sustituyendo la letra ’s’ por la ’m’ de la palabra “casa” se llega a su solución vecina “cama”.

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.

El esquema general de los algoritmos de Búsqueda Local se muestra en el Algoritmo 1.

Algorithm 1 Esquema general de búsqueda local *first-accept* y *best-accept*

```
1: function BusquedaLocal-FirstAccept(S_0)
2:     S* ← S_0
3:     while ¬CriterioParada(S*) do
4:         for S ∈ V(S*) do
5:             if c(S) < c(S*) then
6:                 S* ← S
7:                 break
8:     return S*
1: function BusquedaLocal-BestAccept(S_0)
2:     S* ← S_0
3:     while ¬CriterioParada(S*) do
4:         S' ← S*
5:         for S ∈ V(S*) do
6:             if c(S) < c(S') then
7:                 S' ← S
8:         S* ← S'
9:     return S*
```

3. DEFINICIÓN DEL PROBLEMA

En el marco del *VeRoLog Solver Challenge 2017* (VSC2017), se presenta como problema a abordar una variante proveniente de una aplicación real en la distribución de herramientas para la medición de calidad en la industria lechera. Dicho problema abarca un planeamiento en un horizonte discreto de días donde existe un conjunto de clientes que tienen pedidos que deben ser satisfechos cumpliendo restricciones particulares.

Luego, el problema consiste en resolver de manera integrada la planificación de la utilización de las herramientas en dicho horizonte de tiempo, medido en días numerados $1, 2, \ldots, D$, y la distribución de las mismas incorporando la logística necesaria para el ruteo de las mismas en cada día.

Sea $G = (V, E)$ un digrafo completo con $V = \{0, 1, \ldots, n\}$ el conjunto de nodos que representan a los pedidos, y E el conjunto de ejes que representan el camino entre los clientes de dichos pedidos. Se tiene un nodo especial, el nodo 0, que representa al depósito. Cada nodo $v \in V$, incluyendo el depósito, tiene una posición en el plano. Cada eje $(i, j) \in E$ tiene una distancia $\delta_{i,j} \geq 0$ asociada, calculada como la parte entera de la distancia euclidiana (i.e. $\lfloor \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \rfloor$).

Cada pedido $v = (v_{\text{ent}}, v_{\text{rec}}) \in V$ se define por su entrega v_{ent} de las herramientas al cliente, y su recolección v_{rec} de las mismas t_v días después. La entrega de un pedido tiene que realizarse dentro de una ventana de tiempo predefinida $[a_v, b_v]$, $1 \leq a_v \leq b_v \leq D$. El pedido v entregado en el día $d \in [a_v, b_v]$ debe ser recogido en el día $d + t_v$. Por ejemplo, si la entrega de un pedido sucede en el día 2 y éste usa las herramientas por 3 días, las mismas deben ser recogidas en el día 5. Nótese que esta ventana de tiempo no es la denotación estándar sino que puede ser pensado como una "ventana de entrega".

Hay $\mathcal{T}$ diferentes tipos de herramientas, y cada tipo $\tau \in \mathcal{T}$ tiene una cierta cantidad de stock máximo disponible $\tau_{\max}$ y un peso q^τ. Cada pedido $v \in V$ requiere una cantidad de herramientas q_v, de un tipo particular notado $\tau(v)$. Para $v = (v_{\text{ent}}, v_{\text{rec}})$ se asume $q_{v_{\text{ent}}} \leq 0$ y $q_{v_{\text{rec}}} \geq 0$, siendo $q_v = -q_{v_{\text{ent}}} = q_{v_{\text{rec}}}$.

Cada uno de los pedidos tiene asociado un cliente que solamente es usado para determinar las coordenadas del pedido. Nótese que si bien cada cliente puede realizar varios pedidos, cada pedido requiere exactamente un tipo de herramienta. Por esto, si un cliente quiere varios tipos de herramientas, debe realizar más de un pedido.

Para entregar estos pedidos se dispone de una flota ilimitada y homogénea de camiones que recorren los arcos $(i, j) \in E$. Tienen una capacidad Q finita y una distancia máxima L para cada día. Las rutas deben comenzar y terminar en el depósito. Asimismo, se permiten los *multi-trips*: visitas intermedias al depósito para cargar o descargar herramientas.

Un punto a aclarar es que solamente el camión que lleva una herramienta va a poder entregarla ese día. Es decir, las herramientas que lleva un camión no pueden ser inter-

cambiadas con otro camión dentro del mismo día. En cualquier momento de la ruta, cada camión puede volver a depósito para dejar herramientas y reusarlas en otra parte de su misma ruta, así como también no utilizarlas. Al finalizar el día, las herramientas vuelven al depósito común de herramientas y pueden ser utilizadas por cualquier camión a partir del día siguiente.

Cada camión puede visitar a un cliente solamente para entregar o recolectar herramientas que fueron pedidas. Las herramientas asociadas a un pedido deben ser entregadas por un mismo camión en un solo momento. Asimismo, son recolectadas por un solo camión en un solo momento, aunque no tiene por qué ser el mismo camión que las entregó.

Se tiene un costo fijo c^f por contratar a cada camión, un costo fijo c^f_{dia} por cada día de uso de cada camión, y un costo por distancia recorrida c^f_{dist}. Además, se tiene un costo c^τ por el uso de cada herramienta de tipo $\tau \in \mathcal{T}$ que es pagado una única vez si se utiliza esa herramienta en algún momento del horizonte.

Una solución válida se compone de planear las entregas y recolecciones de herramientas en el horizonte de días, de forma tal que satisfagan todos los pedidos con un mínimo costo total.

Una primera aproximación es la de separar la asignación de pedidos a días del ruteo en sí. De esta manera, se puede pensar que se tiene un VRP con capacidades para cada día del horizonte. Debido a las restricciones del problema, incluso sabiendo cuál es la asignación óptima de pedidos a días, encontrar el ruteo óptimo es difícil si se considera cada día aislado del resto. Aún más, al aislar cada día del resto, también es un problema no trivial encontrar un ruteo factible debido al stock limitado de herramientas que se tiene.

Debido a la dificultad de lograr obtener una solución desde cero, en esta tesis se hace énfasis en el refinamiento de solución válida. En los siguientes capítulos, se presentan dos maneras distintas de resolver el problema. La primera utiliza operadores de búsqueda local, mientras que la segunda plantea la creación de un modelo de programación lineal entera.

Se presenta la Figura 3.1 para mostrar un ejemplo junto con una posible solución. Por simplificación, se utiliza un horizonte de tres días, donde cada cliente realiza un único pedido y cada pedido utiliza las herramientas únicamente por un día. Asimismo, se omite el tipo y cantidad de herramientas usadas.

En esta figura, se representa la ruta de cada camión con un color distinto. La arista contiene una E o una R, representando que se realiza una entrega o recolección.

En la Figura 3.1a se puede ver la disposición de los clientes en el plano, en adición al depósito representado con el 0. Esta disposición es la misma que se usa el resto de los días, dado que no se suman, agregan o modifican estas posiciones.

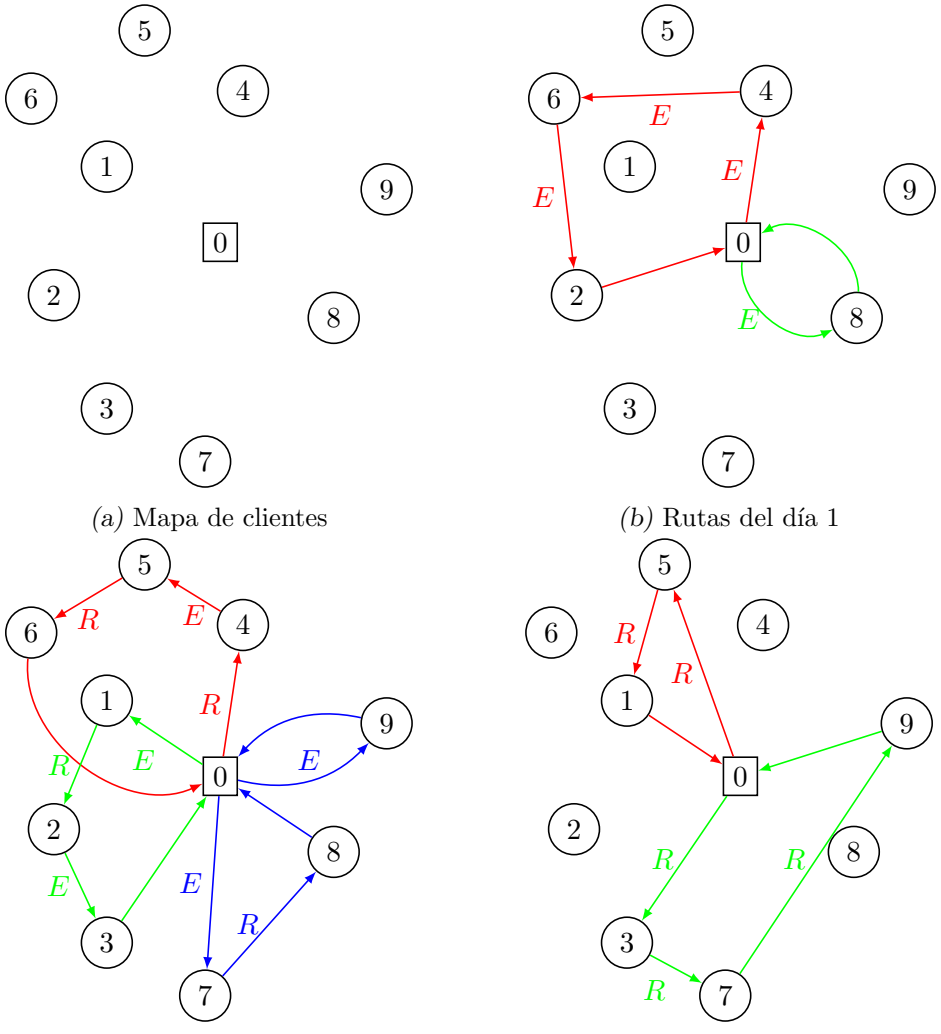
En las Figuras 3.1b - 3.1d se tienen arcos con curva, pero es meramente para que se entiendan los gráficos. Los camiones toman siempre la distancia euclidiana tal como se indicó previamente.

Se nota que cada pedido tiene exactamente una entrega y una recolección. Asimismo,

dicha entrega y recolección están separadas por un día debido a que cada cliente utiliza las herramientas un solo día.

En el primer día de cada horizonte siempre se tienen sólo entregas, así como también en el último sólo recolecciones. En este caso, las Figuras 3.1b y 3.1d respectivamente. Es imposible tener una recolección el primer día porque significa que se tendría que tener una entrega fuera del horizonte de días. Análogamente, no se puede entregar nada el último día porque significaría que no hay recolección dentro del horizonte de días.

Se nota que estas figuras son a modo ilustrativo. Por ejemplo, en la Figura 3.1c la ruta azul se podría pensar como $[0, 7, 8, 0, 9, 0]$ o $[0, 9, 0, 7, 8, 0]$. Se utiliza la primera de estas opciones pero a priori ambas podrían ser válidas. Esta ruta vuelve al depósito luego de visitar al cliente 8, y, por ejemplo, en esta visita al depósito puede dejar las herramientas que recolectó de 8 así como también juntar las herramientas para entregar a 9.



(a) Mapa de clientes

(b) Rutas del día 1

(c) Rutas del día 2

(d) Rutas del día 3

Fig. 3.1: Ejemplo de una posible instancia y su solución

4. BÚSQUEDA LOCAL Y OPERADORES BÁSICOS

Una primera aproximación se centra en utilizar operadores de búsqueda local clásicos. En ellos, se plantea el conjunto de soluciones como un gran grafo donde las soluciones son los nodos y están conectadas entre sí si son parecidas, para algún criterio establecido. De esta forma, permite navegar el vecindario de soluciones pudiendo llegar a óptimos locales.

Para la elección de los operadores, se consultaron las ideas presentadas en P. Toth & D. Vigo [15], W. P. Nanry & J. W. Barnes [12], y Ascheuer et al. [1]. Se utilizaron operadores variados lo cual permite recorrer soluciones más diversas, y no estancarse tan fácilmente en una solución particular. Luego se detallarán operadores que actúan sobre los nodos, así como también operadores que actúan sobre rutas. A su vez, algunos operadores tienen en cuenta de un vecindario restringido a un único día, mientras que otros operadores actúan sobre varios días.

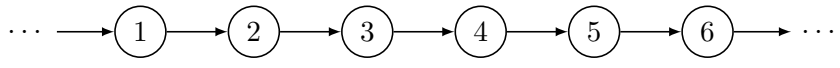
A priori estos operadores pueden utilizarse de distintas maneras, por ejemplo realizando movimientos de la forma *First-accept* o *Best-accept*. Asimismo, también pueden utilizarse para navegar vecindarios con soluciones infactibles. En esta sección vamos a hablar de los operadores locales en sí, para luego en la Sección 4.5 hablar de los algoritmos y criterios utilizados.

A continuación se presentan los distintos operadores usados en la experimentación, en conjunto con figuras para facilitar la lectura. En las siguientes figuras se muestran rutas donde se utiliza el 0 para denotar al depósito, y los números distintos a 0 representan nodos de entregas o recolecciones.

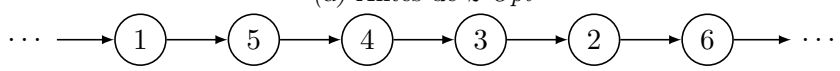
4.1. Operadores Locales Intra-Ruta

4.1.1. 2-Opt

2-Opt es un operador local que consiste en intercambiar dos ejes dentro de la misma ruta. Luego de intercambiar los ejes, se dan vuelta los ejes dentro de la subruta que se rompió para mantener consistencia.



(a) Antes de *2-Opt*



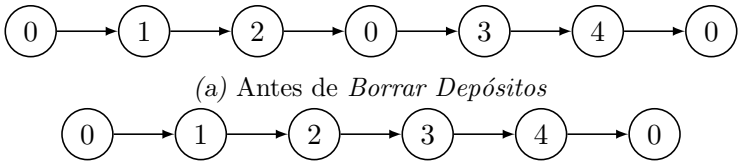
(b) Después de *2-Opt*

Fig. 4.1: Ejemplo donde se da vuelta la subruta del nodos 2 al nodo 5 dentro de la misma ruta

4.1.2. Borrar Depósitos

Borrar Depósitos es un operador local que remueve depósitos intermedios. Nótese que remueve depósitos *intermedios* que son nodos opcionales, a diferencia de los nodos de inicio y fin que son necesarios para que la ruta esté bien definida.

El objetivo de este operador es reducir la distancia de la ruta. Puede suceder que un depósito intermedio fuera necesario para alguna solución intermedia, pero que luego de aplicar otros operadores ya no lo sea. En caso de poder eliminar un depósito intermedio, directamente afecta al funcional ya que la distancia de la ruta es reducida. Nótese que no modifica el funcional en terminos de stock o camiones utilizados.



(a) Antes de *Borrar Depósitos*

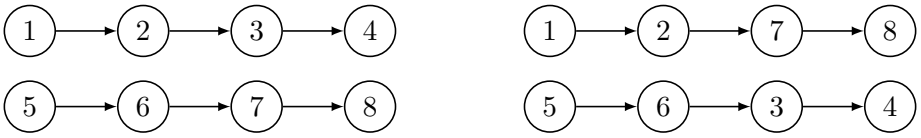
(b) Después de *Borrar Depósitos*

Fig. 4.2: Ejemplo donde se borra un depósito intermedio

4.2. Operadores Locales Inter-Ruta

4.2.1. Doble Intercambio Ruta

Doble Intercambio Ruta es un operador local inter-ruta que actúa sobre distintas rutas del mismo día, combinando el inicio de una ruta con el fin de la otra, y viceversa. En la Figura 4.3 se omiten los depósitos de inicio y fin de las rutas para mayor claridad.



(a) Antes de Doble *Intercambio Rutas* *(b)* Después de Doble *Intercambio Rutas*

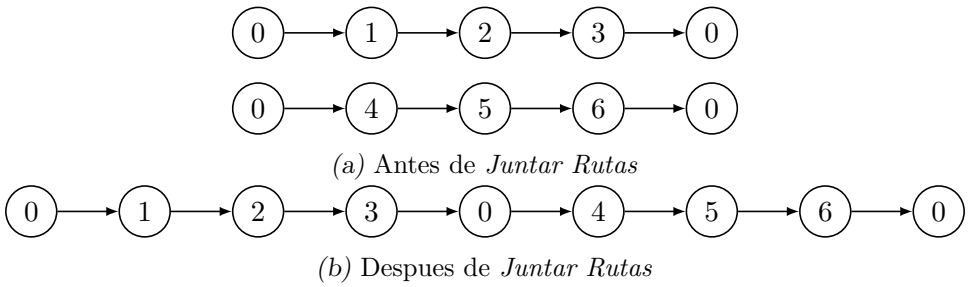
Fig. 4.3: Ejemplo donde se intercambian los finales de dos rutas del mismo día

4.2.2. Juntar Rutas

Juntar Rutas es otro operador local inter-ruta que actúa sobre distintas rutas del mismo día. Combina dos rutas en una sola, "pegando" una ruta al finalizar la otra. El depósito final de una ruta y el inicio de la otra se convierten en un depósito intermedio.

El objetivo de este operador es reducir la cantidad de rutas. En caso de poder juntar dos rutas, directamente afecta al funcional ya que la cantidad de camiones utilizados es

reducida. Además, en caso de ser el día que utilizó más camiones puede reducir los camiones máximos utilizados. Nótese que no modifica el funcional en terminos de stock o distancia.



(a) Antes de *Juntar Rutas*

(b) Despues de *Juntar Rutas*

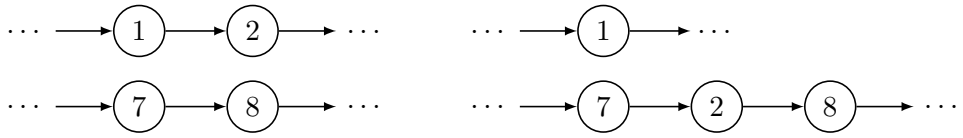
Fig. 4.4: Ejemplo donde se juntan dos rutas del mismo día

4.3. Operadores Locales Intra-ruta e Inter-Ruta

Los siguientes operadores pueden ser usados tanto intra-ruta como inter-ruta. Es decir, pueden actuar únicamente sobre una ruta, o sobre dos. En el caso en que se aplique sobre dos rutas, los chequeos de factibilidad necesarios aumentan pero se mantienen dentro del mismo orden de complejidad. Para un mayor detalle, ver la Sección 4.6.

4.3.1. Mover Nodo Mismo Día

Mover Nodo Mismo Día es un operador que remueve el nodo de su posición actual para colocarlo en otra posición dentro del mismo día. Esta nueva posición puede quedar dentro de la misma ruta, como en otra ruta del mismo día.

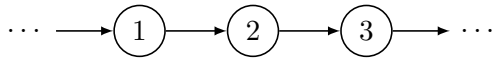


(a) Antes de *Mover Nodo Mismo Día* *(b)* Después de *Mover Nodo Mismo Día*

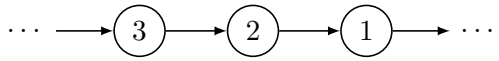
Fig. 4.5: Ejemplo donde se mueve al nodo 2 a otra ruta dentro del mismo día

4.3.2. Intercambio Nodos Mismo Día

Intercambio Nodos Mismo Día es un operador local que intercambia la posición de dos nodos dentro del mismo día. Los nodos pueden pertenecer a la misma ruta, o a distintas.



(a) Antes de *Intercambio Nodos Mismo Día*



(b) Después de *Intercambio Nodos Mismo Día*

Fig. 4.6: Ejemplo donde se intercambian los nodos 1 y 3 dentro de la misma ruta

4.4. Operador Local Distinto Día

4.4.1. Mover Entrega Recolección Distinto Día

Mover Entrega Recolección Distinto Día es el operador similar a *Mover Nodo Mismo Día* pero actúa moviendo el par de nodos de día. Por ser análogo a su versión del mismo día, se omite figura para este operador.

Nótese que en el problema abordado los nodos están asociados en pares entrega y recolección, los cuales están separados por una cantidad fija de días. De esta forma, si se realiza un cambio de días para un elemento de este par, el otro elemento debe moverse la misma cantidad de días para mantener la distancia de días requerida entre el par. Al mover dos nodos de día, en particular afecta a más de una ruta por lo que es un operador inter-ruta.

A modo de ejemplo, en la Figura 4.7 se muestra como si se mueve un par entrega recolección que tiene tres días de uso, se debe mantener esa distancia entre la entrega y la recolección al mover los nodos de días. Se utiliza las barras roja y verde para mostrar, respectivamente, cuáles eran los días en que el pedido es satisfecho dentro del horizonte de días.

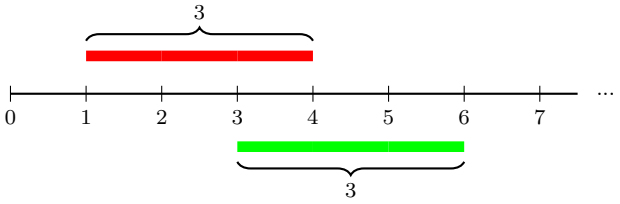


Fig. 4.7: Ejemplo donde se mueve un par entrega-recolección de los días 1 y 4 a los días 3 y 6

Se recuerda que esta distancia es una constante que depende del pedido en cuestión y no puede modificarse. Es decir, no se pueden retirar las herramientas antes de tiempo, así como tampoco se puede decidir no recogerlas en el día que corresponde.

4.5. Algoritmos de Operadores Locales

Cada operador se comporta de una manera similar, recorriendo el horizonte de días en orden y dentro del día en el orden de las rutas. En el caso que el operador actúe sobre nodos, a su vez recorre los nodos en el orden de la ruta.

Al aplicarse el operador correspondiente, se utiliza la forma *Best-Accept* explicada la Sección 2.4.2: Se mueve hacia la mejor solución vecina de la actual, si es una mejora. Además, en los operadores sobre nodos, en caso de encontrar una mejor solución se reinicia al comienzo de la ruta.

Cada operador busca el mejor movimiento factible que puede hacer, teniendo en cuenta todos los movimientos posibles para él. Por ejemplo, en el caso de *Mover Nodo Mismo Día* una vez que se tiene el nodo a mover, se lo intenta insertar en todas las rutas del día y entre cada par de nodos.

Los Algoritmos 2 y 3 presentan un pseudo-código que ilustra el comportamiento de los operadores sobre nodos y sobre rutas, respectivamente.

Algorithm 2 Comportamiento de Operadores Sobre Nodos

```
1: for dia ∈ 𝒟 do
2:     for ruta ∈ ℛ_dia do
3:         for nodo ∈ ruta do
4:             S* ← OPERADOR(S, nodo)
5:             if c(S*) < c(S) then
6:                 S ← S*
7:                 REINICIA_AL_COMIENZO_DE_LA_RUTA()
8: return S
```

Algorithm 3 Comportamiento de Operadores Sobre Rutas

```
1: for dia ∈ 𝒟 do
2:     for ruta ∈ ℛ_dia do
3:         S* ← OPERADOR(S, ruta)
4:         if c(S*) < c(S) then
5:             S ← S*
6: return S
```

Por otro lado, al tener distintos operadores se tiene que decidir un orden en la corrida de dichos operadores. A continuación se presentan distintas versiones: *Ad-hoc*, *Nodos, Rutas*, y *Mismo Día, Distinto Día*.

Ad-hoc es la configuración de nodos que se eligió al momento de decidir una configuración. Luego, fue refinada en *Nodos, Rutas* para realizar primero los operadores que actuaban sobre nodos para luego los que actúan sobre rutas.

Otra opción para refinamiento fue *Distinto Día, Mismo Día*. Mover los nodos de días

resulta difícil debido a las restricciones del problema ya que implica reacomodar un par entrega-recolección en una posición distinta en días distintos, que sea factible y mejore el funcional.

Solamente la factibilidad es un problema no menor. El operador *Mover Entrega Recolección Distinto Día* resulta el más difícil de utilizar debido a que requiere que se puedan remover tanto el nodo de entrega como de recolección de sus rutas, así como también poder insertarlos en rutas de otros días que mantengan la distancia necesaria entre ellos.

En términos de distancia de la ruta, las rutas de donde se remueven los nodos siempre van a estar dentro del límite pero en las que se agregan nodos pueden ahora superar dicha distancia máxima.

Por otro lado, la cantidad de herramientas necesaria para satisfacer los pedidos debe ser recomputada, así como también el peso de los camiones a lo largo de las rutas afectadas. Por si esto fuera poco, también se tiene que tener en cuenta el stock a lo largo de los días en el horizonte ya que la utilización de las herramientas es distinta.

Por esto, se creó una configuración que intente primero realizar esto para luego utilizar los operadores que actúan sobre el mismo día.

Estas versiones utilizan a los operadores ejecutándolos uno atrás de otro en un orden particular. Generalizando, se tiene el Algoritmo 4 que recibe una lista de operadores, y ejecuta cada operador sobre S según lo ilustrado en los Algoritmos 2 y 3.

Algorithm 4 Algoritmos Generales

```
1: function ALGORITMO GENERAL(Lista<Operador> operadores, Solucion S)
2:     for operador in operadores do
3:         S ← OPERADOR(S)
4:     return S
```

Numerando los operadores cómo:

1. *Mover Nodo Mismo Día*
2. *Intercambio Nodos Mismo Día*
3. *2-Opt*
4. *Mover Entrega Recolección Distinto Día*
5. *Doble Intercambio Ruta*
6. *Juntar Rutas*
7. *Borrar Depósitos*

El orden de las configuraciones es:

- *Ad-Hoc*: $[1, 2, 3, 4, 5, 6, 7]$
- *Nodos, Rutas*: $[1, 2, 4, 3, 5, 6, 7]$
- *Mismo Día, Distinto Día*: $[4, 1, 2, 3, 5, 6, 7]$

Nótese que siempre se llama último a *Borrar Depósitos* ya que se usa como limpieza luego de utilizar el resto de los operadores.

En la Sección 6.5 se presentan los experimentos realizados con estas configuraciones.

4.6. Factibilidad de Operadores y Recálculo del Valor del Funcional

Debido a que los operadores actúan de forma local, se crean estructuras de datos redundantes que guardan resultados ya calculados. Éstas, permiten testear rápidamente la factibilidad de utilizar los operadores así como también acelerar el cálculo del nuevo valor del funcional. Para testear la factibilidad, se necesita recalcular la distancia y el stock únicamente de las rutas modificadas, así como también el stock global. En esta Sección, se llama stock global al conteo de cantidad de herramientas usadas cada día.

La distancia y el stock deben ser actualizados ya que se remueven o agregan nodos a las rutas afectadas por el operador. El stock global debe ser actualizado y revisado ya que las modificaciones de stock de un operador pueden llegar a repercutir a todos los días, en el peor caso. Por ejemplo, si se mueve un par de nodos entrega-recolección no solo afectan al stock de los días en que se removieron y agregaron dichos nodos, sino que también afectan a los días entre la entrega y recolección tanto original como la nueva.

En términos de distancia, se debe revisar que no se pase de la distancia máxima. Análogamente, con el peso del camión para cada ruta. Como caso especial, no hace falta revisar que la distancia sea menor a la máxima de una ruta donde solo se remueven nodos y no se agrega ningún otro, ya que la distancia total va a ser menor o igual al remover un nodo de la ruta. Se recuerda que las distancias utilizadas son euclidianas como se explicó en la Sección 3.

Por otro lado, también se debe tener la cantidad y el tipo necesario de herramientas para cada entrega de la ruta. Asimismo, se debe tener el cuenta el peso del camión en cada paso de la ruta. En cuanto al stock global, hay que asegurarse que la cantidad de herramientas utilizadas no excedan a la cantidad de herramientas posibles en ningún día para ningún tipo.

La cantidad de rutas modificadas está acotada según el operador y a lo sumo son cuatro rutas. Los operadores intra-ruta (Sección 4.1) afectan exactamente una ruta, los operadores inter-ruta (Sección 4.2) exactamente dos rutas, mientras que los operadores tanto intra como inter-ruta (Sección 4.3) afectan a lo sumo dos rutas en el caso de usarse en dos rutas distintas del mismo día. En el caso de operadores de distinto día (Sección 4.4) afectan a cuatro rutas en el caso general, y a tres rutas en el caso donde uno de los

nodos del par entrega-recolección removido se mueve a la ruta de la cual su par asociado fue removido.

Al actualizar estas estructuras de datos, también tenemos la ventaja de poder obtener el nuevo funcional sin tener que recurrir a computar todo de nuevo. Como dichos chequeos se hacen a menudo, actualizar solamente lo necesario permite testear factibilidad y obtener el nuevo valor del funcional con una mayor rapidez. Esta optimización tiene un impacto considerable en el tiempo de ejecución de los operadores.

4.7. Complejidad de operadores

Las estructuras de datos subyacentes se crean con el objetivo de realizar el agregado y remoción de nodos de forma rápida. En términos de complejidades, el costo del operador recae en la cantidad de posibilidades que tiene. Esto se debe a que luego de todo cambio debemos actualizar las estructuras de datos, según lo hablado en la Sección 4.6, y estos chequeos tienen un costo mayor que la utilización de los operadores.

El costo del chequeo es: $O(\mathcal{L}) + O(D)$, donde $\mathcal{L}$ la longitud máxima de las rutas (longitud en términos de cantidad de nodos en la ruta), y D la cantidad de días en el horizonte. Esto se debe a que se deben revisar las rutas modificadas, así como también el stock global.

La longitud de las rutas está acotado superiormente por la cantidad de nodos máximos posibles para esos días, que en el peor caso son todos los nodos. Además, puede tener depósitos intermedios. Sin embargo, estos depósitos nunca están uno al lado del otro por lo que a lo sumo la longitud de la ruta se duplica. Luego, la complejidad de dicho chequeo es $O(n) + O(D)$, donde n es la cantidad de nodos.

Por otro lado, la cantidad de movimientos posibles para cada operador es $O(n)$ para cada operador, ya que a lo sumo es probar de utilizarlo en cada lugar posible. En este contexto el "lugar" son o bien los nodos o las aristas de las rutas que como se vio esta acotado por $O(n)$. A lo sumo se buscan dos lugares para insertar pero su complejidad es la misma, ya que se lo multiplicaría por la constante dos.

Finalmente, la complejidad de cada operador es $O(n) \times (O(n) + O(D))$. En general n es mayor a D, ya que en caso de no serlo se tiene un caso donde la cantidad de pedidos es menor a la cantidad de días. Luego, la complejidad de cada operador es $O(n^2)$.

5. MODELO DE REUBICACIÓN

Un operador que se consideró es el de mover varios nodos a la vez de la posición actual a otra, con el objetivo de mejorar el valor de la solución. Si se quiere implementar este algoritmo, se tiene que explicitar la manera en que se recombinan estos nodos: intentar todas las opciones posibles, utilizar una política *first-accept*, etc.

Generar un algoritmo que logre probar las distintas combinaciones es una tarea dificultosa dada la cantidad de restricciones de este problema. Por suerte, este es el dominio por excelencia de la programación lineal ya que permite expresar las restricciones requeridas y obtener una solución, sin entrar en detalles de cómo generar las posibles combinaciones. De esta forma, se va a abordar la creación de un modelo de programación lineal para poder obtener dichas combinaciones.

5.1. Esquema General

En este trabajo, se considera el esquema propuesto por De Franceschi et al. [8] para el VRP con Capacidad y Restricciones de Distancia, que fue aplicado con éxito a otras variantes del VRP. 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 Modelo de Reubicación.

En De Franceschi et al. se propone el algoritmo SERR (Selección, Extracción, Recombinación y Reubicación), el cual a partir de una solución inicial considera un vecindario de tamaño exponencial. Para un detalle de este procedimiento ver el Algoritmo 5.

Siguiendo la notación propuesta por De Franceschi et al., sea $\mathcal{Z}$ el conjunto de todas las soluciones factibles, y sea z_0 una solución factible inicial. Siguiendo el paradigma de destrucción/reparación, se seleccionan heurísticamente un conjunto de vértices llamado $\mathcal{F}$, y una solución restringida $z_0(\mathcal{F})$ concatenando vértices consecutivos en z_0 luego de la extracción.

Cada eje en $z_0(\mathcal{F})$ es llamado *insertion point* y se denota $\mathcal{I} = \mathcal{I}(z_0, \mathcal{F})$ al conjunto de todos los *insertion point* de $z_0(\mathcal{F})$. Sea $\mathcal{S} = \mathcal{S}(\mathcal{F})$ el conjunto de todas las secuencias posibles, sin repeticiones y de cualquier tamaño, obtenida por las recombinaciones de los vértices de $\mathcal{F}$.

Cada secuencia $s \in \mathcal{S}$ puede ser asignada a lo sumo a uno de los *interstion point* de $\mathcal{I}$, y cada $i \in \mathcal{I}$ puede ser asignado como mucho a una secuencia. El vecindario denotado como $\mathcal{N}(z_0, \mathcal{F})$ considera todas las soluciones factibles que provienen de asignar las secuencias de $\mathcal{S}$ a *insertion points* de $\mathcal{I}$. Luego, se tiene el objetivo de obtener una solución mejor a la inicial, explorando dicho vecindario resolviendo un ILP.

Algorithm 5 SERR (Selección, Extracción, Recombinación y Reubicación)

1: **while** ¬Criterio de Parada **do**
2: (Selección) Aplicar algún criterio para seleccionar los vértices a extraer. Estos vértices extraídos componen al conjunto $\mathcal{F}$.
3: (Extracción) Extraer los vértices del paso anterior y construir la solución restringida concatenando los nodos no extraídos consecutivos. El conjunto $\mathcal{I}$ contiene a los ejes en la solución restringida llamados *insertion points*.
4: (Recombinación) Las secuencias de nodos consecutivos extraídos por el punto anterior generan las llamadas "secuencias básicas", las cuales son guardadas en un conjunto de secuencias. Luego, se generan heurísticamente las secuencias utilizando los nodos pertenecientes a $\mathcal{F}$ y son agregadas al conjunto de secuencias. Adicionalmente, cada secuencia s es heurísticamente asociada con un subconjunto $\mathcal{I}_s \subseteq \mathcal{I}$ indicando que la secuencia s puede ser insertada en el *insertion point* $i \in \mathcal{I}_s$. En particular, las secuencias básicas contienen a los insertion points originales, de forma tal de poder regenerar la solución inicial.
5: (Reubicación) Se decide una asignación factible de secuencias a insertion points, formulando y resolviendo el ILP utilizando un ILP solver de propósito general. Se definen las variables binarias x_{si} que valen 1 si y solo si la secuencia s es asignada al *insertion point* i. Se impone un tiempo límite a la ejecución del solver y se obtiene la mejor solución hasta el momento.

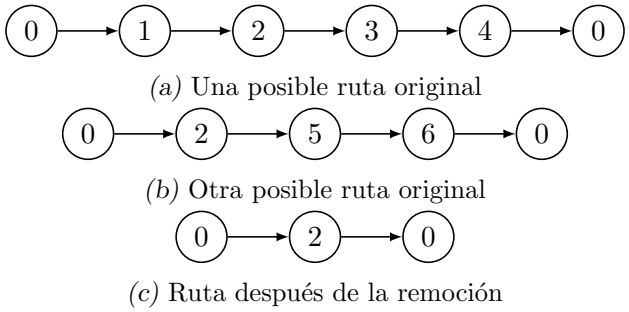
5.2. Ejemplos

A continuación se procede a profundizar estas definiciones, mostrando ejemplos para facilitar la lectura. Una secuencia se define como una cadena no vacía, finita de nodos con un orden. Estos nodos pueden ser una entrega o una recolección pero no pueden ser depósitos. En la Figura 5.1 se muestran ejemplos de secuencias posibles, donde los números representan nodos que pueden ser de entrega o recolección.

Fig. 5.1: Ejemplos de secuencias

Un *insertion point* son todos aquellos puntos en donde estas secuencias pueden ser ubicadas. En el modelo, equivalen a las aristas de las rutas una vez que los nodos fueron removidos. Debido a su definición, distintas rutas pueden generar los mismos *insertion point* una vez que los nodos son removidos. A continuación se presenta un ejemplo de esto donde los números distintos de 0 representan nodos de entrega o recolección, mientras que el 0 representa depósitos.

En la Figura 5.2, se tienen dos posibles rutas a las cuales se les remueven nodos (los conjuntos $\{1, 3, 4\}$ y $\{5, 6\}$ respectivamente) y terminan generando la misma ruta después de la remoción. Incluso, puede suceder que de alguna ruta no se remueva ningún nodo. De esta forma, la ruta $0 \longrightarrow 2 \longrightarrow 0$ también es un ejemplo válido como "ruta original" en la Figura 5.2.



(a) Una posible ruta original

(b) Otra posible ruta original

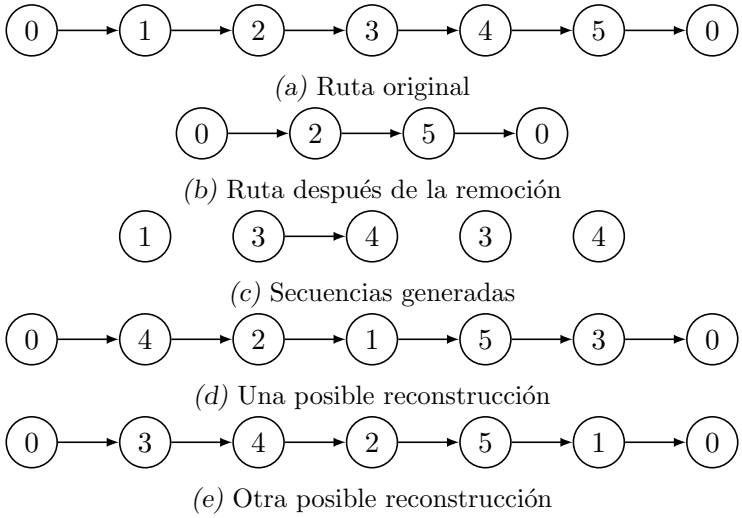
(c) Ruta después de la remoción

Fig. 5.2: Ejemplo de distintas rutas que generan los mismos *insertion points*

Nótese que de la ruta de la Figura 5.2a se remueve el nodo 1, generando un *insertion point*. Por otro lado, el par (3, 4) genera un solo *insertion point* debido a que están uno al lado del otro. Es decir, por más que se hayan removido dos nodos, se genera un único *insertion point*. De esta forma, se removieron 3 nodos y se generaron 2 *insertion point*.

Al momento de la reconstrucción, se tiene que los nodos tienen que ser reubicados para poder generar una solución válida, ya que todos los clientes deben ser satisfechos. Sin embargo, no necesariamente todas las secuencias son reubicadas porque puede suceder que un nodo esté en varias secuencias.

Se recuerda que cada *insertion point* admite una sola secuencia, por lo que no puede suceder que más de una secuencia esté asociada a un mismo *insertion point*. Asimismo, cada secuencia puede ser asignada a lo sumo a un *insertion point*, ya que en caso contrario algún pedido tendría más de una entrega o recolección.



(a) Ruta original

(b) Ruta después de la remoción

(c) Secuencias generadas

(d) Una posible reconstrucción

(e) Otra posible reconstrucción

Fig. 5.3: Ejemplo donde se remueven los nodos 1, 3 y 4 y se reacomodan en la misma ruta

En la Figura 5.3 se tiene una ruta a la cual se le remueven tres nodos, los cuales van a ser reacomodados. Este reacomodo puede ser realizado en distintas rutas e incluso en distintos días, pero se va a mantener dentro en la misma ruta por simplicidad.

Nótese que en la Figura 5.3b se tienen 3 *insertion point* posibles: En las aristas (0,2),

(2,5), y (5,0). Sean estos *insertion point* A, B, y C respectivamente.

En la Figura 5.3c se puede observar como las secuencias generadas son tanto los nodos unitarios, como la secuencia $3 \longrightarrow 4$. A priori, el modelo permite secuencias de tamaño arbitrario.

Además de estas secuencias, se generan las secuencias necesarias para poder reconstruir la solución inicial, sin importar el tamaño de las mismas. Esto permite que el modelo siempre pueda llegar a una solución mejor o igual a la que ya tenía, así como también poder devolver siempre una solución.

Debido a que los nodos 3 y 4 están colocados uno al lado del otro, se genera la secuencia que contiene a ambos con el fin de poder reconstruir la ruta original. Luego, se muestran dos posibles reacomodos.

En 5.3d se asignan los nodos unitarios 4, 1 y 3 en los insertion point A, B, y C, respectivamente. Por otro lado, en 5.3e se puede ver que la secuencia $3 \longrightarrow 4$ fue asignada en el *insertion point* A, mientras que el nodo 1 fue asignado en el *insertion point* C. Cabe aclarar que B quedó "vacío", es decir, ninguna secuencia fue asignada a él. Por esta razón, la arista se mantiene igual que previa la asignación de secuencias a los *insertion point*.

A continuación, se utiliza el día 2 del ejemplo que se presentó en la definición del problema (Sección 3) para mostrar una posible recombinación de dicho día. Es interesante revisar este día en particular ya que contiene a todos los clientes y tiene tanto entregas como recolecciones.

Se utiliza la recombinación sólo de dicho día por simplificación, pero se nota que este reacomodo puede pasar para varios días a la vez. Incluso puede suceder que se cambien entregas o recolecciones de días.

En la Figura 5.4a se toma como la solución factible inicial a la mostrada en la Sección 3. Luego, se seleccionan los nodos $\{1, 7, 9\}$ para remover (Figura 5.4b), que generan el conjunto $\mathcal{F}$.

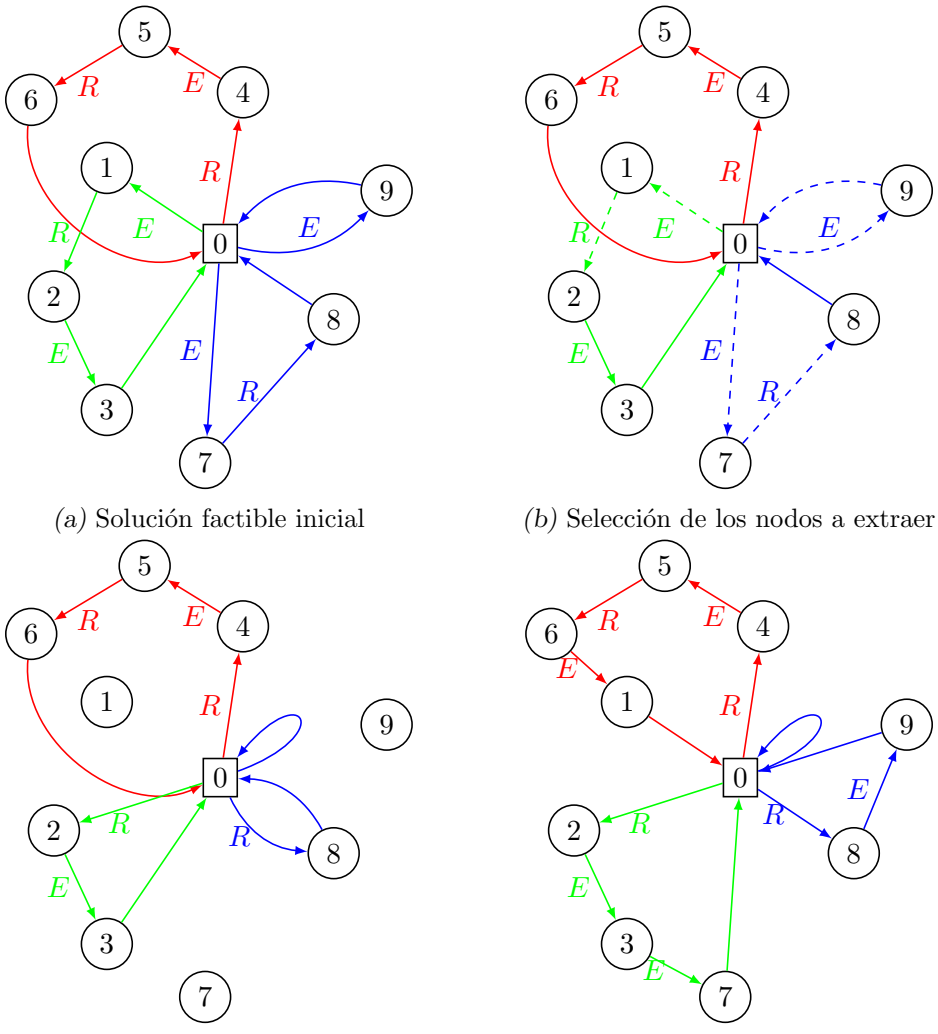
Una vez que se extraen los nodos, se genera la solución restringida $z_0(\mathcal{F})$ (Figura 5.4c). Se nota que esta solución es infactible ya que no todos los pedidos tienen entrega. En particular, sucede que los nodos removidos son todas entregas pero se nota que también se pueden remover recolecciones.

Los *insertion points* son $0 \longrightarrow 4$, $4 \longrightarrow 5$, $5 \longrightarrow 6$, y $6 \longrightarrow 0$ de la ruta roja; $0 \longrightarrow 2$, $2 \longrightarrow 3$, y $3 \longrightarrow 0$ de la ruta verde; y $0 \longrightarrow 8$, $8 \longrightarrow 0$, y $0 \longrightarrow 0$ de la ruta azul.

Un *insertion point* interesante es el de $0 \longrightarrow 0$ en la ruta azul. Como se tenía la subruta $0 \longrightarrow 9 \longrightarrow 0$ y se sacó al 9, se tiene un *loop* volviendo al depósito. Se nota que en caso de que ese *loop* se mantenga, se tendrían dos depósitos seguidos. Si bien esto no cambia el valor del funcional (ya que no se suma distancia, stock, etc.) estas ocurrencias de dos depósitos seguidos son fácilmente detectables y removibles. Se nota que siempre es posible remover cuando hay dos depósitos seguidos y nunca se llega a una infactibilidad.

De esta forma, se tienen diez *insertion point* y tres nodos removidos. En particular, las secuencias básicas necesarias para restaurar la solución factible inicial son todas unitarias, pero podría darse que no. Por ejemplo, si se hubieran removido los nodos 4 y 5, se tendría la secuencia básica $4 \longrightarrow 5$ para poder volver a la solución inicial.

Finalmente, se genera una solución del modelo de reubicación (Figura 5.4d) al colocar los nodos removidos: 1 es colocado en $6 \longrightarrow 0$, 7 en $3 \longrightarrow 0$, y 9 en $8 \longrightarrow 0$. En este ejemplo, se colocan los nodos justo antes de llegar a un depósito pero se nota que pueden colocarse en cualquier parte de la ruta.



(a) Solución factible inicial

(b) Selección de los nodos a extraer

(c) Solución restringida (infactible)

(d) Solución de reubicación

Fig. 5.4: Ejemplo de extracción y reubicación de nodos

5.3. Formulación del Modelo

Debido al tamaño del modelo, éste va a ser explicado incrementalmente para facilitar la lectura. Para la conveniencia del lector, se presenta la Tabla A.1 en el Apéndice A

enunciando la notación utilizada.

Sea $\mathcal{D} = \{1, 2, \ldots, D\}$ el conjunto de días consecutivos que componen al horizonte de tiempo. $\mathcal{T}$ es el conjunto de los tipos de herramientas. Q indica la capacidad máxima de cada vehículo, mientras que q^τ es el peso de cada herramienta de tipo $\tau \in \mathcal{T}$.

En cuanto a las secuencias, $\mathcal{S}$ es el conjunto de ellas que son divididas según los días. $\mathcal{S}_d$ denota al conjunto de secuencias que pueden ser asignadas a un *insertion point* en el día $d \in \mathcal{D}$. Notar que $\mathcal{S} = \cup_{d \in \mathcal{D}} \mathcal{S}_d$. A su vez, se define a $\mathcal{D}_s$ como los días posibles para insertar a la secuencia $s \in \mathcal{S}$.

Se define como $\mathcal{R}$ al conjunto de rutas, $\mathcal{D}_r$ al día de la ruta $r \in \mathcal{R}$, y $\mathcal{R}_d$ como las rutas del día $d \in \mathcal{D}$. Se nota que los $r \in \mathcal{R}$ son las rutas pertenecientes a la solución restringida $z_0(\mathcal{F})$.

Cada ruta puede tener depósitos intermedios, y son partidas en subrutas que comienzan y terminan en depósitos. Se nota que solo se considera una subruta como tal, si no contiene depósitos intermedios dentro de ella. Se define K_r como el conjunto de subrutas de la ruta $r \in \mathcal{R}$, numeradas desde el 0. r^k se utiliza para nombrar a la subruta de índice $k \in \{0, \ldots, |K_r| - 1\}$ de la ruta $r \in \mathcal{R}$.

A modo de ejemplo siguiendo la Figura 5.5, sean $\{k_0, \ldots, k_3\}$ los depósitos de la ruta r. k_0 y k_3 son los depósitos de comienzo y fin de la ruta, mientras que k_1 y k_2 son depósitos intermedios. La ruta r es partida en r_0, r_1 y r_2 teniendo en cuenta a los depósitos de la ruta. Se recuerda que las subrutas no pueden contener depósitos intermedios dentro de ellas, por lo que no existe la subruta que va desde k_0 hasta k_2 dado que el depósito k_1 está en el medio.

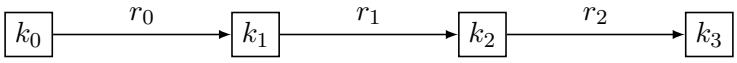


Fig. 5.5: Explicación de la concordancia entre una ruta y sus subrutas

$\mathcal{I}$ representa al conjunto de *insertion points*. Éstos se dividen por rutas, con $\mathcal{I}_r$ indicando conjunto de *insertion points* de $r \in \mathcal{R}$. También, se tiene a $\mathcal{I}_{r^k}$, restringido a los que pertenecen a la subruta r^k. Luego, r_i denota a la ruta correspondiente al *insertion point* $i \in I$.

5.3.1. Explicación Depósitos y Subrutas

Para explicar la notación utilizada acerca de la concordancia de subrutas y depósitos se utiliza como soporte la Figura 5.6. En ella, se toma una simplificación del modelo donde se considera un solo tipo de herramienta, y una sola ruta por motivos de una explicación más sencilla. Se notan a los depósitos como k_i para indicar que es la i-ésima visita al depósito, comenzando desde el número 0. Se recuerda que el depósito es único para toda ruta y siempre el mismo.

La subruta r_i comprende a los nodos entre la visita i e $i + 1$ al depósito. Se nota como

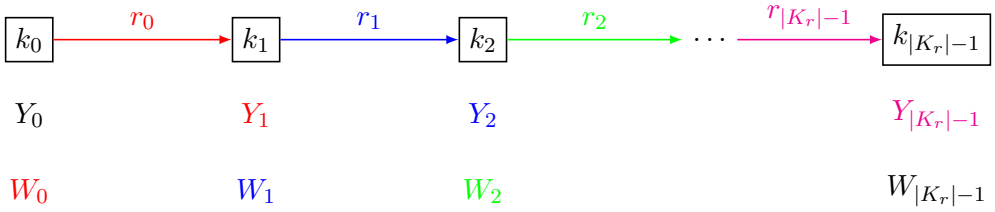


Fig. 5.6: Explicación de la concordancia entre depósitos y subrutas

W_i e Y_i, respectivamente, a la cantidad de herramientas que saca y que devuelve el camión, en la visita i-ésima. Luego, cada subruta r_i puede pensarse que toma herramientas de la visita i por el depósito y las devuelve en la $i+1$. De esta forma, la subruta r_i está asociada a W_i y a Y_{i+1}

Nótese que hay dos variables que no están asociadas a ninguna subruta: Y_0 y $W_{|K_r|-1}$. Se definen igual a 0 por completitud ya que no hay ruta que devuelva herramientas antes de comenzar, o que tome herramientas para entregar después de la última visita por el depósito.

La consistencia entre variables W_i e Y_i así como también su generalización a varios tipos de herramienta, se formaliza en la Sección 5.3.4.

5.3.2. Constantes Adicionales

A continuación se definen constantes que ayudan a contabilizar la disponibilidad de herramientas, así como también el peso de los camiones. Para definirlas, se extienden algunas definiciones de Montero et al. [10] al caso de *multi-commodity*.

Se nota como $\tau(v)$ el tipo de herramienta usada en el nodo v, y q_v como la demanda de herramientas del nodo v en la secuencia, siendo $q_v \geq 0$ para las recolecciones y $q_v \leq 0$ para las entregas. Se recuerda que cada nodo está asociado a un único tipo de herramienta por lo que no se tienen demandas de tipos distintos.

Sea la secuencia $s \in \mathcal{S}$, se define a $\overline{q}(s)$ como el *peso máximo* de la secuencia s; $\underline{q_\tau}(s)$ como el *stock requerido* por la secuencia s para cada tipo τ; y $q_\tau(s)$ como la *diferencia del stock* de tipo τ entre el comienzo y el fin de la secuencia s. Sus definiciones son las siguientes:

$$\overline{q}(s) = \text{máx}\left\{0, \text{máx}_{l=1,\ldots,|s|} \sum_{\tau \in \mathcal{T}} \left(q^\tau \times \sum_{\substack{j=1 \\ \tau(v_j)=\tau}}^{l} q_{v_j}\right)\right\}$$

$$\underline{q_\tau}(s) = \text{mín}\left\{0, \text{mín}_{l=1,\ldots,|s|} \sum_{\substack{j=1 \\ \tau(v_j)=\tau}}^{l} q_{v_j}\right\}$$

$$q_\tau(s) = \sum_{\substack{j=1 \\ \tau(v_j)=\tau}}^{|s|} q_{v_j}$$

Sea el *insertion point* $i \in I$, se define $q_a^\tau(i)$ como la *diferencia del stock restringida* de tipo τ hasta el *insertion point* i (inclusive). Se nota que solamente se tiene en cuenta a la subruta de la solución restringida, a la que dicho *insertion point* pertenece.

$$q_a^\tau(i) = \sum_{\substack{j=(x,y)\in I(r_i)\ :\\ j\leq i\ \wedge\ \tau(x)=\tau}} q_x$$

5.3.3. Restricciones de Capacidad

Se define la variable binaria x_{si} que toma valor 1 si y solo si la secuencia s es asignada al *insertion point* i. Por otro lado, $w_r^{\tau,k}$ indica el número de herramientas de tipo τ para la ruta r al salir del depósito k-ésimo. Se recuerda que la subruta l está asociada con los $w_r^{\tau,l}$, es decir la del mismo índice, como se mostró en la Sección 5.3.1.

$$\sum_{s\in\mathcal{S}_{(\mathcal{D}_r)}} \overline{q}(s)x_{si} + \sum_{\tau\in\mathcal{T}} q^\tau \times \left(w_r^{\tau,k} + q_a^\tau(i) + \sum_{\substack{j\in\mathcal{I}_{r^k}\\ j<i}} \sum_{s\in\mathcal{S}_{(\mathcal{D}_r)}} q_\tau(s)x_{sj} \right) \leq Q$$
$$r \in \mathcal{R}, k \in \{0,\ldots,|K_r|-1\}, i \in \mathcal{I}_{r^k} \qquad (5.1)$$

Esta restricción exige que al agregar las secuencias los camiones no se pasen de su capacidad máxima Q, en cualquier momento de la ruta. Para asegurar que no sucede en cada momento de la ruta, basta revisar que no suceda en cada *insertion point*.

El primer sumando suma el peso máximo de la secuencia a agregar en el *insertion point* i. Se recuerda que a lo sumo una secuencia puede ser asignada a cada *insertion point*. El segundo sumando suma el peso total de las herramientas hasta ese punto. Para hacer esto, se multiplica el peso de cada herramienta por la cantidad de herramientas que se tienen hasta el *insertion point* exclusive.

Dicha cantidad de herramientas es representada por los tres sumandos dentro del paréntesis, respectivamente: la cantidad de herramientas que salen del depósito, la diferencia de herramientas teniendo en cuenta solamente a la solución restringida, y la diferencia de herramientas teniendo en cuenta a la asignación de los *insertion points* anteriores de la misma subruta. Se nota que estos dos últimos sumandos pueden ser negativos (e.g si se tienen solo entregas), pero la suma total dentro del paréntesis nunca va a ser menor a 0. Esto último es asegurado por las restricciones de la Sección 5.3.4.

5.3.4. Restricciones de Stock por Ruta

La variable $y_r^{\tau,k}$ indica el número de herramientas de tipo τ de la ruta r al llegar al depósito k-ésimo. Se recuerda que la subruta l esta asociada con los $y_r^{\tau,l+1}$, es decir la del siguiente índice, como se mostró en la Sección 5.3.1.

Para una mayor claridad en la restricción (5.3), se usa $q^{\tau}_{r^k}$ para indicar la diferencia del stock restringida de tipo τ al finalizar la subruta r^k. Se recuerda que solamente se tiene en cuenta a la subruta de la solución restringida.

$$w_r^{\tau,k} + q_a^{\tau}(i) + \sum_{\substack{j \in \mathcal{I}_{r^k} \\ j<i}} \sum_{s \in \mathcal{S}_{(\mathcal{D}_r)}} q_\tau(s) x_{sj} + \sum_{s \in \mathcal{S}_{(\mathcal{D}_r)}} \underline{q_\tau}(s) x_{si} \geq 0$$

$$r \in \mathcal{R}, k \in \{0, \ldots, |K_r| - 1\}, i \in \mathcal{I}_{r^k}, \tau \in \mathcal{T} \tag{5.2}$$

$$y_r^{\tau,k+1} = w_r^{\tau,k} + q_{r^k}^{\tau} + \sum_{i \in \mathcal{I}_{r^k}} \sum_{s \in S(\mathcal{D}_r)} q_\tau(s) x_{si}$$

$$r \in \mathcal{R}, k \in \{0, \ldots, |K_r| - 1\}, \tau \in \mathcal{T} \tag{5.3}$$

Estas restricciones exigen que el stock sea el correcto dentro de cada ruta. Esto es, que siempre haya stock para entregar a los pedidos de la ruta (5.2), y que todo el stock del camión al final de la ruta vuelva al depósito (5.3). Los casos de $y_r^{\tau,0}$ y $w_r^{\tau,|K_r|}$ son contemplados en la Sección 5.3.5.

La restricción (5.2) asegura que en cada *insertion point* y para cada tipo se tenga un stock no negativo. Se descompone en, respectivamente: las herramientas que salieron de depósito, la diferencia de cantidad de herramientas teniendo en cuenta solamente a la solución restringida, la diferencia de stock de los *insertion point* anteriores, y la cantidad de herramientas que demanda la secuencia que puede ser insertada en el *insertion point* que se está considerando.

La restricción (5.3) establece la concordancia entre variables w e y. Las herramientas que vuelven al depósito al finalizar la subruta son la suma de: las que salieron de depósito, la diferencia de cantidad de herramientas teniendo en cuenta solamente a la solución restringida, y la la diferencia de cantidad de herramientas de los *insertion point* de la subruta.

5.3.5. Concordancia entre Subrutas y Rutas

El número de herramientas nuevas de tipo τ sacadas en la visita k-ésima al depósito para la ruta r es denotado como $\eta_r^{\tau,k}$. "Nuevas" se refiere a las herramientas sacadas en ese momento, y que no fueron sacadas previamente ni recogidas en las recolecciones de esta misma ruta.

Por otro lado, $\psi_r^{\tau,k}$ indica el número de herramientas del tipo τ que tiene a favor la ruta r en la visita k por el depósito, una vez que sale el camión. Se recuerda que si un camión deja herramientas en depósito, solamente él puede utilizarlas ese día.

Debido a que cuenta las herramientas a favor, ψ es una variable acumulada para facilitar el cálculo. En caso de no hacerlo, habría que recalcular utilizando todas las w e y anteriores. Se nota que un camión puede dejar las herramientas a favor en el depósito en

una pasada para luego utilizarlas en alguna otra subruta siguiente, así como también no utilizarlas.

$$\eta_r^{\tau,0} = w_r^{\tau,0} \qquad \tau \in T, r \in \mathcal{R} \tag{5.4}$$
$$\eta_r^{\tau,k} = \text{máx}(0, w_r^{\tau,k} - \psi_r^{\tau,k-1} - y_r^{\tau,k}) \qquad \tau \in T, r \in \mathcal{R}, k \in \{1, \ldots, |K_r|\} \tag{5.5}$$
$$\psi_r^{\tau,0} = 0 \qquad \tau \in T, r \in \mathcal{R} \tag{5.6}$$
$$\psi_r^{\tau,k} = \text{máx}(0, \psi_r^{\tau,k-1} + y_r^{\tau,k} - w_r^{\tau,k}) \qquad \tau \in T, r \in \mathcal{R}, k \in \{1, \ldots, |K_r|\} \tag{5.7}$$
$$y_r^{\tau,0} = 0 \qquad \tau \in T, r \in \mathcal{R} \tag{5.8}$$
$$w_r^{\tau,|K_r|} = 0 \qquad \tau \in T, r \in \mathcal{R} \tag{5.9}$$

La concordancia entre rutas y sus subrutas se mantienen debido a las restricciones definidas en esta sección: (5.4) y (5.5) controlan el stock que sale del depósito, (5.6) y (5.7) controlan el stock a favor que tiene el camión en un determinado momento, y (5.8) y (5.9) son los casos agregados por completitud para las variables y y w, respectivamente.

Se pueden linealizar las restricciones 5.5 y 5.7 utilizando una formulación *big M*. Para linealizar una restricción del estilo $a = \text{máx}(0, b)$ se tienen las restricciones:

- $a - b \geq 0$ (En nuestros casos, $a \geq 0$ siempre vale por la definición de la variable)
- $a - b - binaria_a \times BigM \leq 0$, con $binaria_a$ una variable binaria que vale 0 si a toma el valor de b, y 0 si no. $BigM$ es un valor constante suficientemente grande.
- $a + binaria_a \times BigM \leq BigM$

Se presenta la Figura 5.7 para ilustrar el comportamiento de las variables η y ψ. Al igual que la Figura 5.6 de la Sección 5.3.1 se utiliza un modelo simple en donde se tiene un solo tipo de herramienta para simplificar la explicación. Dado que los distintos tipos de herramientas no interfieren entre sí, se lo puede generalizar fácilmente al caso *multi-commodity*.

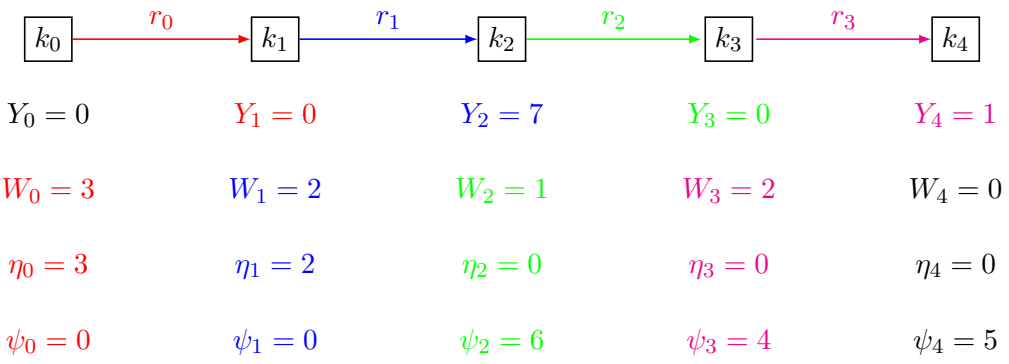


Fig. 5.7: Ejemplo de una ruta para la explicación de las variables η y ψ

En r_0 se sacan tres herramientas del depósito, que son todas nuevas por lo que η_0 también es igual a tres. No se devuelven herramientas en esta subruta.

En r_1 se sacan dos herramientas y se devuelven siete. η_1 es igual a dos ya que se sacan dos herramientas nuevas. Se nota que ψ_1 es igual a cero por más que parezca que se tienen herramientas a favor debido a que se devuelven más que las que se sacan. Esto es porque el ψ_1 se calcula en relación a ψ_0, Y_1, y W_1.

En r_2 sale una herramienta pero no se considera como nueva ($\eta_2 = 0$). Esto es ya que se saca una pero previemente se devolvieron siete. De esta manera, también quedan seis a favor (i.e $\psi_2 = 6$). Al finalizar esta ruta, no se devuelven herramientas.

Finalmente, en r_3 se sacan dos herramientas, pero no se consideran nuevas ($\eta_3 = 0$). Esto es debido a que se tenían herramientas a favor y se usaron esas. Es decir, $\psi_3 = 4$ cuando $\psi_2 = 6$. Acá se puede observar la naturaleza acumulativa de la ψ.

Debido a que no hay una r_4, W_4 y η_4 son iguales a 0. Nótese que ψ_4 es igual a cinco, lo cual indica cuántas herramientas se tienen a favor al finalizar la ruta. Esto se puede pensar como la cantidad de herramientas que no estaban al comienzo de la ruta y sí al final. Esta noción de herramientas al finalizar la ruta se profundiza en la Sección 5.3.6.

5.3.6. Restricciones de Stock entre Rutas

La variable μ_τ^d denota el número de herramientas en el depósito de tipo τ en el día d, después que hayan salido todos los camiones y antes que ninguno haya vuelto. Es decir, controla el stock que queda en el depósito.

La variable α_τ^d indica cuántas herramientas del tipo τ salieron del depósito en el día d, considerando como "salida" si la herramienta estaba al inicio del día pero no al final. Se define ζ_τ^d como el opuesto: las herramientas del tipo τ que entraron al depósito en el día d, considerando como "entrada" si la herramienta no estaba al inicio del día pero sí al final.

$$\mu_\tau^d = \mu_\tau^{d-1} + \zeta_\tau^{d-1} - \alpha_\tau^d \qquad d \in \{2, \dots, D\}, \tau \in \mathcal{T} \tag{5.10}$$

$$\alpha_\tau^d = \sum_{r \in \mathcal{R}_d} \sum_{k \in K_r} \eta_r^{\tau,k} \qquad d \in \mathcal{D}, \tau \in \mathcal{T} \tag{5.11}$$

$$\zeta_\tau^d = \sum_{r \in \mathcal{R}_d} \psi_r^{\tau,|K_r|} \qquad d \in \mathcal{D}, \tau \in \mathcal{T} \tag{5.12}$$

Las restricciones mostradas garantizan que el stock inter-rutas sea el correcto: (5.10) controla el stock que queda en el depósito teniendo en cuenta las herramientas que había al finalizar el día anterior, y las que salen el día actual. Las igualdades (5.11) y (5.12) garantizan que sean correctas las herramientas que salen y vuelven del depósito, respectivamente.

Nótese que la restricción (5.10) comienza en el día 2 y no en el 1. μ_τ^1 se encuentra en

el cálculo del valor del funcional, como se va a ver más adelante en la Sección 5.3.12.

Se nota que en la restricción (5.12) basta tomar la última ψ ya que es acumulativa. En contraposición en la restricción (5.11), hay que sumar todas las η ya que éstas no lo son.

5.3.7. Restricciones de Comienzo y Fin

La variable binaria z_{vd}^{inicio} se define como igual a 1 si y solo si el pedido v comienza en el día d, y z_{vd}^{fin} siendo igual 1 si y solo si el pedido v finaliza en el día d (es decir, que las herramientas deben ser recolectadas ese día).

V define al conjunto de pedidos, tanto los extraídos como los que no lo fueron, mientras que $\mathcal{F}_{\text{ent}}$ y $\mathcal{F}_{\text{rec}}$ denotan al conjunto de nodos extraídos que son entregas y recolecciones, respectivamente. El conjunto de rutas del día $d \in \mathcal{D}$ es denotado como $\mathcal{R}_d$.

$$z_{vd}^{\text{inicio}} = \sum_{\substack{s \in \mathcal{S}_d \\ v \in s}} \sum_{r \in \mathcal{R}_d} \sum_{i \in \mathcal{I}_r} x_{si} \qquad v \in \mathcal{F}_{\text{ent}}, d \in \mathcal{D} \tag{5.13}$$

$$\sum_{d \in \mathcal{D}} z_{vd}^{\text{inicio}} = 1 \qquad v \in \mathcal{F}_{\text{ent}} \tag{5.14}$$

$$z_{vd}^{\text{fin}} = \sum_{\substack{s \in \mathcal{S}_d \\ v \in s}} \sum_{r \in \mathcal{R}_d} \sum_{i \in \mathcal{I}_r} x_{si} \qquad v \in \mathcal{F}_{\text{rec}}, d \in \mathcal{D} \tag{5.15}$$

$$\sum_{d \in \mathcal{D}} z_{vd}^{\text{fin}} = 1 \qquad v \in \mathcal{F}_{\text{rec}} \tag{5.16}$$

$$z_{vd}^{\text{inicio}} = z_{p\ d+t_v}^{\text{fin}} \qquad v \in \mathcal{F}_{\text{rec}} \cap \mathcal{F}_{\text{ent}}, d \in \mathcal{D} \tag{5.17}$$

Las restricciones (5.14) y (5.16) aseguran que cada pedido tenga exactamente un único comienzo y fin, mientras que (5.13) y (5.15) se aseguran que estos comienzos y fines estén entre los días permitidos.

Finalmente, la igualdad (5.17) asegura que para cada pedido su comienzo y su fin estén separados de la cantidad de días correcta. Cabe aclarar que solo importan los nodos que fueron sacados tanto para entrega como para recolección. Si para un pedido v solamente se sacó su entrega o recolección, el removido tiene un único día posible para ser asignado y las secuencias que contienen a dicho pedido no van a poder ser asignadas en otro día que no sea el correcto.

5.3.8. Restricciones de Distancia

La variable δ'_r denota la longitud de la ruta r luego que se hayan asignado las secuencias.

Nótese que si bien δ'_r es variable, las siguientes distancias son constantes. δ_r es la distancia de la ruta $r \in \mathcal{R}$ sin contener los nodos removidos ni las secuencias asignadas. δ_s

indica la distancia de la secuencia $s \in \mathcal{S}$, y $\delta v_1, v_2$ denota a la distancia entre los clientes de los pedidos $v_1, v_2 \in V$.

Se nota como s_0 al primer nodo de la secuencia $s \in \mathcal{S}$, y como s_n al último. En el caso de tener una secuencia de un solo nodo, se tiene que $s_0 = s_n$. Asimismo, se nota que si la secuencia s tiene un solo nodo, δ_s es igual a 0.

$$\delta'_r = \delta_r + \sum_{s=(s_0,\ldots,s_n)\in\mathcal{S}_d} \sum_{i=(x,y)\in\mathcal{I}_r} x_{si} \times \Big(\delta_s + \delta_{x,s_0} + \delta_{s_n,y} - \delta_{x,y}\Big) \qquad d \in \mathcal{D}, r \in \mathcal{R}_d \tag{5.18}$$

Esta restricción controla la longitud de cada ruta en la nueva solución, luego que se hayan asignado las secuencias. La nueva longitud es la que se tenía antes más la consideración de la introducción de las nuevas secuencias dentro de la ruta. La diferencia de longitud está representada dentro de los paréntesis: la longitud de la secuencia, sumado a los nuevos dos ejes, y restando el eje de la solución restringida.

Para asegurar de tener una solución válida, δ'_r es acotada superiormente por la distancia máxima permitida en la Sección 5.3.12.

5.3.9. Restricciones de Unicidad de Secuencias e *Insertion Point*

$$\sum_{s\in\mathcal{S}_{(D_{(r_i)})}} x_{si} \leq 1 \qquad i \in \mathcal{I} \tag{5.19}$$

$$\sum_{d\in\mathcal{D}_s} \sum_{r\in\mathcal{R}_d} \sum_{i\in\mathcal{I}_r} x_{si} \leq 1 \qquad s \in \mathcal{S} \tag{5.20}$$

La restricción (5.19) asegura que haya una única secuencia por *insertion point*, mientras que (5.19) asegura que cada secuencia esté en a lo sumo un *insertion point*.

5.3.10. Restricciones de Rutas Usadas

El conjunto Ω denota a las rutas que pueden quedar vacías, es decir, aquellas rutas donde todos sus nodos estén asignados a alguna secuencia. Vale que $\Omega \subseteq \mathcal{R}$.

La variable ω_r es definida como igual a 0 si y solo si la ruta $r \in \Omega$ queda vacía luego que se hayan asignado las secuencias. Se puede afirmar que una ruta $r \in \Omega$ queda vacía si y solo si todos sus *insertion points* no son asignados a secuencias.

El objetivo de saber qué rutas quedan vacías recae en que pueden ser removidas. En caso de removerlas, se tiene una solución estrictamente mejor a no hacerlo ya que el costo

de utilizar el camión desaparece. Además, puede tener la ventaja de reducir la cantidad máxima de camiones utilizados lo cual reduciría aún más el valor del funcional.

$$\omega_r = 0 \Leftrightarrow \sum_{i \in \mathcal{I}_r} \sum_{s \in \mathcal{S}_{(D_{(r_i)})}} x_{si} = 0 \qquad r \in \Omega \tag{5.21}$$

La restricción (5.21) controla si una ruta termina siendo vacía. En caso que quede vacía, el camión asociado a esa ruta se va a poder dejar de usar reduciendo el costo de la solución. Esta restricción es importante ya que permite decidir si vale la pena una reubicación de nodos teniendo en cuenta las remociones de rutas.

Es posible linealizar (5.21) utilizando una formulación *big M*. Para linealizar una restricción del estilo $a = 0 \Leftrightarrow b = 0$ primero se las desagrega en $a = 0 \implies b = 0$ y $a = 0 \impliedby b = 0$. Luego, para $a = 0 \implies b = 0$ se tiene:

- $a - b \times BigM \geq 0$, donde $BigM$ es un número lo suficientemente grande.

Se nota que en nuestros casos vale que $a \geq 0$ y $b \geq 0$ por la definición de las variables. Análogamente se tienen las restricciones para $a = 0 \impliedby b = 0$.

5.3.11. Restricciones de Camiones

Λ_d denota la cantidad de camiones usados en el día d, y Λ la cantidad de camiones máxima usados en el horizonte de días.

$$\Lambda_d = |\mathcal{R}_d| - \sum_{\substack{r \in \Omega \\ \mathcal{D}_r = d}} (1 - \omega_r) \qquad d \in \mathcal{D} \tag{5.22}$$

$$\Lambda \geq \Lambda_d \qquad d \in \mathcal{D} \tag{5.23}$$

La restricción (5.22) controla que en cada día la cantidad de camiones sea la justa (teniendo en cuenta la posibilidad de remover rutas), y (5.23) controla la cantidad de camiones necesarios en general. Como se verá luego, el funcional se va a encargar que Λ sea el mínimo posible.

5.3.12. Función Objetivo y Composición del Modelo

Se recuerda que se tiene un costo fijo c^f por contratar a un camión, un costo fijo $c^f_{\text{día}}$ por cada día de uso de cada camión, y un costo por distancia recorrida c^f_{dist}. Además, se tiene un costo c^τ por el uso de cada herramienta de tipo τ

$$\sum_{r \in \mathcal{R}} \delta'_r * c^f_{\text{dist}} + \tag{5.24}$$

$$\sum_{\tau \in T} (\mu^1_\tau + \alpha^1_\tau) * c^\tau + \tag{5.25}$$

$$\sum_{d \in \mathcal{D}} \Lambda_d * c^f_{\text{día}} + \tag{5.26}$$

$$\Lambda * c^f \tag{5.27}$$

La función objetivo presentada calcula el costo total de ruteo, el cual va a ser minimizado. Nótese que el funcional calcula exactamente el valor de la solución: (5.24) se encarga del costo de la distancia, (5.25) del costo de las herramientas, (5.26) del costo de los vehículos por día, y (5.27) del costo de los camiones máximos utilizados.

La cantidad de herramientas usadas, calculada en (5.25), se piensa como las herramientas que había en depósito el primer día una vez que salieron los camiones más las que salieron ese día.

Finalmente, el modelo se compone de:

mín (5.24) + (5.25) + (5.26) + (5.27)
sujeto a:
(5.1) - (5.23)

$x_{si}, z^{\text{inicio}}_{vd}, z^{\text{fin}}_{vd}, \omega_r \in \{0, 1\}$
$\eta^{\tau,k}_r, w^{\tau,k}_r, y^{\tau,k}_r, \mu^d_\tau, \psi^{\tau,k}_r \in \{0, \ldots, \tau_{\max}\}$
$\delta'_r \in \{0, \ldots, L\}$
$\Lambda_d, \Lambda \in \{0, \ldots, |V|\}$

5.4. Utilización del Modelo

En esta sección se va a explicar como se utiliza el modelo de reubicación de la Sección 5.3. Este algoritmo tiene distintas variaciones que se detallan en la Sección 5.5. Teniendo una solución inicial como input, su implementación es la siguiente:

Algorithm 6 Ciclo General ILP

```
1: while ¬CriterioParada do
2:     Elegir nodos
3:     Armar secuencias
4:     Reducir vecindad
5:     Llamar ILP
6:     Reconstruir solucion en base al resultado del ILP
```

Primero se eligen los nodos a remover, para luego armar secuencias con ellos y una vez

que están armadas las secuencias se procede a reducir la vecindad de nodos.

Dicha reducción es opcional, pero a lo largo de este trabajo se va a mostrar la utilidad de reducirla. Tiene cómo objetivo achicar heurísticamente el tamaño del modelo, cortando soluciones factibles en pos de lograr iteraciones más rápidas.

Cuando esto termina, se procede a ejecutar el modelo para finalmente construir una nueva solución con el resultado del mismo. Como criterio de parada se puede elegir realizar una cierta cantidad de iteraciones o correr una cierta cantidad de tiempo.

5.5. Variaciones del Modelo

En esta sección se van a proponer distintas variaciones que se pueden aplicar al modelo de reubicación, con el objetivo de obtener un valor objetivo final menor. Se basan en reducir astutamente el tamaño del modelo descartando heurísticamente combinaciones, lo cual permite realizar más iteraciones.

Una pregunta importante es si el hecho de considerar un subconjunto de combinaciones compensa su miopía logrando un mejor valor final de la solución. Así, la motivación es lograr hacer iteraciones más rápido con el objetivo de lograr más de ellas, sin sacrificar mucho la calidad de cada iteración.

Una técnica utilizada es la de reducción del *pool* de nodos. Ésta basa su efectividad en utilizar un criterio para determinar los nodos posibles a escoger para ser removidos.

Por otro lado, se utilizan reducciones del vecindario con el objetivo de reducir el tamaño del modelo. Se nota como el *vecindario* de una secuencia, a los *insertion points* posibles para la misma. Tener un vecindario reducido corta soluciones que a priori pueden ser factibles en pos de lograr iteraciones más rápidas.

Las reducciones de vecindario se basan en filtrar *insertion points* de secuencias, pero se podría haber hecho de la manera inversa (i.e filtrar las secuencias posibles para cada *insertion point*). Filtrar teniendo en cuenta a las secuencias pone el foco en ellas y permite que cada secuencia tenga por lo menos un *insertion point* posible.

En la reducción de vecindario se van a hablar de cercanía de secuencias a *insertion point*. Insertar la secuencia s en el insertion point i tiene una distancia asociada en base a la distancia entre ellos. La Figura 5.8 ilustra los ejes que se agregan y el que se saca.

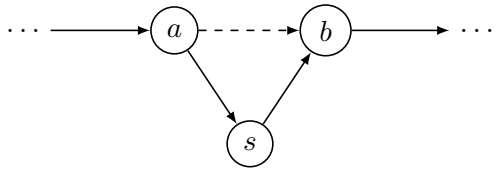


Fig. 5.8: Ejemplo donde se inserta la secuencia s en el insertion point $i = (a, b)$

Se nota que las distancias son las euclidianas, por lo que sucede que $\delta_{a,b} \leq \delta_{a,s} + \delta_{s,b}$.

Asimismo, como se utilizan secuencias de tamaño uno (i.e una sola entrega o recolección) la longitud de la secuencia es cero. De esta forma, si se tienen secuencias de tamaño más grande, se podría llegar a evaluar la utilidad de considerar la longitud de la secuencia al costo de insertar a s en i.

Un punto a aclarar es que las secuencias básicas que se generan con el propósito de poder reconstruir la solución inicial, solo pueden ser insertadas en su único *insertion point* correspondiente. De esta manera, no son consideradas para filtrar su vecindario. Asimismo, para que siempre sean posibles de elegir no van a ser filtradas con las variaciones que reducen el *pool* de nodos.

En las Secciones 5.5.1 y 5.5.2 se consideran técnicas de reducción del *pool* de nodos, mientras que las Secciones 5.5.3 y 5.5.4 tienen como objetivo reducir el tamaño del vecindario.

5.5.1. Ventanas de Tiempo

Una variación posible es la de considerar una cantidad contigua y acotada de días dentro del horizonte de tiempo, y se restringen los nodos a elegir para reacomodar a aquellos que pertenecen a rutas dentro de estos días. Se basa en que los pedidos que están en los días lejanos tienden a tener poca relación entre ellos. De esta manera, al restringir la elección de días se espera que las combinaciones de nodos se afecten entre sí para lograr mejores resultados en vez de ser combinaciones independientes.

Se define como *ventana de tiempo* a los días considerados. Luego, para poder considerar todos los días en el horizonte, esta variación se combina con el mecanismo de ventana deslizante (*sliding window* en inglés). Esto permite que el modelo corra para todos los días, y le da una posibilidad de elección a todos los nodos de la instancia.

De esta forma, se tienen dos parámetros a contemplar: el ancho de la ventana de tiempo, y el corrimiento de la misma. Se nota *corrimiento* a los días de diferencia entre una ventana de tiempo y la siguiente.

En el caso de tener corrimientos muy grandes, las ventanas de tiempo no tendrían solapamiento entre sí. Esto produce que haya días contiguos que entrarían dentro de una ventana de tiempo pero que no lo hacen debido a que el corrimiento las saltea. Por ello, pueden existir pares entrega-recolección que no son escogidos a la vez para ser removidos.

Elegir los nodos de a pares permite que éstos cambien de días por lo que el conjunto de combinaciones crece y por ende crece la posibilidad de una solución mejor. Por esto, se decidió tener solapamiento de las mismas.

En la Figura 5.9 se tiene un ejemplo donde cada barra negra horizontal representa a una ventana de tiempo. Se tiene un horizonte de cuatro días, y una ventana de tiempo de dos con un corrimiento de un día. De esta forma, se van a correr tres modelos considerando cada uno considerando una opción: $\{1, 2\}$, $\{2, 3\}$, y $\{3, 4\}$.

En la Figura 5.10 se puede ver el mismo caso, pero con un corrimiento de dos días.

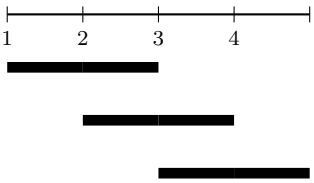


Fig. 5.9: Ejemplo de ventanas de tiempo donde se considera una ventana de tiempo de tamaño dos con un corrimiento de un día, para un horizonte de cuatro días

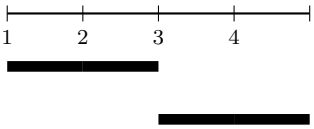


Fig. 5.10: Ídem con un corrimiento de dos días

Esto produce que las opciones sean $\{1, 2\}$, y $\{3, 4\}$. Es decir, no se considera el $\{2, 3\}$. De esta forma, pueden existir nodos en los días 2 y 3 que nunca van a poder ser escogidos a la vez.

5.5.2. Foco Entrega Recolección

Foco Entrega Recolección es una variación que se concentra en elegir nodos de a pares entrega-recolección. De esta forma, o bien ambos pares de entrega y recolección están en secuencias, o ambos no lo están.

La motivación recae en que los pedidos siempre puedan cambiar de días. Cada pedido debe mantener una distancia fija entre su entrega y su recolección, por lo que mover uno de ellos de día sin mover el otro llevaría a una infactibilidad. Se recuerda que los nodos que no son elegidos para remover quedan fijos, y en particular no pueden cambiar de día.

Para realizar esto, los nodos escogidos a remover son únicamente los de entrega. Luego, se escogen a los nodos de recolección asociados a las entregas escogidas.

5.5.3. Corte por Valor

En el trabajo de Toth & Tramontani [14], se comenta una reducción posible que se centra en recortar el vecindario filtrando *insertion points* suficientemente lejanos. Se basa en que los *insertion point* más lejanos tienen un costo más elevado que los cercanos simplemente por un hecho de distancias.

Además, como las rutas tienen una distancia máxima permitida, las rutas que recorren nodos lejanos terminan recorriendo menos nodos. En caso de que suceda, se necesitarían más rutas para recorrer todos los nodos que conlleva cantidad mayor de camiones y un costo mayor.

Una vez elegido el valor de corte, se filtran los *insertion points* que sean más grandes

que dicho valor. Se utiliza un valor de corte por secuencia, ya que la definición de distancia es dependiente de la misma. Para las secuencias básicas, se consideran siempre los *insertion points* iniciales para poder reconstruir la solución inicial.

En nuestra experimentación utilizamos dos cortes distintos: la media de las distancias de los *insertion points* posibles, y uno similar que utiliza la mediana en vez de la media. La media tiende a ser afectada por *outliers* por lo que resulta interesante comparar ambas medidas.

5.5.4. Reducción Vecindad por Días Posibles

Una vez que se realizó la variación de *Foco Entrega Recoleccion* (ver Sección 5.5.2), se pudo observar que los modelos tardaban considerablemente más tiempo en correr. Esto sucedía ya que al aplicar esta reducción todos los pedidos removidos pueden cambiar de días, y por ende su vecindades crecen debido a estos nuevos *insertion points* posibles.

Por esta razón, se implementó la reducción de la vecindad por días posibles para cada secuencia, quedándose con la fracción $1/|\mathcal{D}_s|$ más cercana para cada secuencia $s \in \mathcal{S}$. De esta forma, si un pedido tenía 300 posibles *insertion point* y sus días posibles son 5, sus posibles *insertion point* se reducen a los 60 más cercanos.

Se nota que los *insertion point* que son finalmente escogidos, no son elegidos en base a qué día pertenecen. A priori, pueden pertenecer todos al mismo día.

6. EXPERIMENTACIÓN Y RESULTADOS

Los modelos se implementaron en el lenguaje de programación C++11. Los experimentos se realizaron en una computadora con CPU AMD FX-8350 @ 4.20GHz con 8 GB de memoria. Se utiliza CPLEX 12.7.1 como paquete de resolución PL y PLEM para resolver la relajación del nodo raíz y para desarrollar algoritmos de B&C.

6.1. Instancias

El dataset utilizado para los experimentos se realizó en base al conjunto de instancias presentado por el *VeRoLog solver challenge 2017* [13]. En este *challenge* se presenta una competición en la cual hay distintas instancias a resolver con el objetivo de obtener la solución con costo mínimo.

Se tienen dos datasets: *Hidden* [4] y *Late* [5]. El dataset *Late* es usado por los participantes puedan poner a prueba sus algoritmos previo a la competencia en sí. Para competir, cada competidor entrega el código fuente que es corrido por jueces de la competencia sobre el dataset *Hidden*. Ambos datasets de instancias tienen un tiempo máximo de corrida de $2|P| + 10$ segundos en las máquinas de los jueces, donde P es la cantidad de pedidos de la instancia.

Estos datasets se pueden agrupar en distintos sub-datasets, indicados por el número final del nombre del archivo. Cada uno de estos sub-datasets tienen características definidas, y en esta tesis se van a explorar los datasets que tienen como costo predominante el de la distancia.

Recordamos que el objetivo de la tesis se centra en mostrar la utilidad de un modelo de reubicación. Para utilizar nuestros algoritmos sobre estas instancias, se requieren soluciones factibles como punto de partida. Para las instancias *Hidden* se utilizaron como punto de partida las mejores soluciones logradas por los participantes [6].

Para las instancias *Late* no se obtuvieron resultados públicos. Para poder testear estas instancias, se utilizaron las mejores soluciones que pudo encontrar el ganador del primer premio del concurso[1].

A partir de este momento vamos a hablar de las instancias con el siguiente formato $x\_ry dz$, donde x indica el dataset, y el número de pedidos de la instancia y z la cantidad de días, siguiendo la nomenclatura utilizada en el *VeRoLog solver challenge 2017*. Ejemplos de esta nomenclatura son: late_r1000d20, hidden_r500d10, hidden_r2500d70.

[1] Martin J. Geiger, a quien se le agradece por su colaboración

6.1.1. Detalle de Instancias

A continuación se detallan las instancias con el objetivo de poner en contexto al lector, en particular el tamaño de las mismas así como también sus costos.

El tamaño de la instancia se caracteriza por dos valores: la cantidad de pedidos y la cantidad de días. En los datasets *Late* y *Hidden* los tamaños son los siguientes:

Pedidos	Días
1000	20
1000	25
1500	30
2000	50
2000	65

Fig. 6.1: Tamaños instancias *Late*

Pedidos	Días
500	10
500	15
1000	20
1000	25
1500	30
1500	40
2000	50
2000	65
2500	70
2500	75

Fig. 6.2: Tamaños instancias *Hidden*

Se observa que las instancias *Late* tienen cantidad de pedidos y días iguales a otras instancias *Hidden*. Asimismo, el conjunto de instancias *Hidden* contiene instancias más chicas y más grandes que las *Late*.

En cuanto a costos, se presenta un ejemplo para demostrar la predominancia del costo de la distancia recorrida.

Máximo de Camiones Usados	Camiones Por Día	Stock Tipo 1	Stock Tipo 2	Distancia
1	4.000	30.000.000	50.000.000	300.000

Fig. 6.3: Costos para la instancia late_r1500d30

Los costos de camiones máximos suelen estar en las pocas decenas, camiones por día en las centenas, stock por las centenas para cada tipo de herramienta, y distancia en los millones. Por esta razón, si bien el costo del stock es mayor al de la distancia, al estar en contexto del problema se observa que el impacto del costo de la distancia es el predominante en el valor del funcional.

Esta relación de costos se mantienen con valores e impacto similares, para los distintos tamaños de instancias, ya sean los más chicos o los más grandes.

6.2. Detalle de Soluciones

En cuanto a las soluciones también se pueden obtener datos interesantes, como por ejemplo el de la distancia recorrida en las rutas. Aquellas soluciones que van a ser analizadas son las nombradas anteriormente en este capítulo.

6.2.1. Uso de Herramientas

En el problema, se tienen una cierta cantidad de herramientas con un costo por utilizar cada una por lo menos una vez. Una métrica a evaluar es el porcentaje de uso de herramientas ya que se puede llegar a observar si un tipo de herramientas es más utilizado, o si las herramientas en general lo están. El porcentaje para cada tipo se calcula como $usadas(\tau)$ / $\tau_{\max}$, siendo $usadas(\tau)$ la cantidad total de herramientas usadas de tipo $\tau \in \mathcal{T}$.

Observando estos porcentajes en general, el promedio de uso de las herramientas suele ser parejo para las distintas instancias y los tipos. Si se promedian todas las herramientas para todas las instancias, su uso es de un 73.52 %. A modo de ejemplo, se presenta el detalle del uso de las herramientas para la solución inicial de la instancia hidden_r2000d50.

Tipo 1	Tipo 2	Tipo 3	Tipo 4	Promedio
80.26 %	68.29 %	69.57 %	77.63 %	73.93 %

Fig. 6.4: Ejemplos de porcentaje usado de herramientas para la solución inicial de la instancia hidden_r2000d50

Como se puede observar en la Figura 6.4, se tiene un gran espacio de maniobra en cuanto a las herramientas. El porcentaje promedio de uso es de 73.93 % lo cual es incluso superior al promedio general. Sin embargo, incluso con el tipo 1 se tiene aproximadamente un 20 % de las herramientas sin ser utilizadas.

Las herramientas no son un factor determinante en cuanto a la dificultad de obtener distintas soluciones, ya que es posible obtener soluciones que usen más herramientas. Si se estuvieran usando todas las herramientas, o un número cercano al 100 %, el vecindario de soluciones posibles sería más acotado.

6.2.2. Distancia de Rutas Utilizada

La distancia recorrida por las rutas es un factor importante a la hora de observar vecindarios. Sea la ruta $r \in \mathcal{R}$, se calcula el porcentaje de distancia como $\delta(r)$ / L. Se recuerda que $\delta(r)$ es la distancia de la ruta r, y L la distancia máxima permitida.

Su promedio general es de 87.32 % que no parece ser tan grande como para influir significativamente. Sin embargo, se suelen tener rutas que llegan cerca del límite permitido

con un remanente que no se puede colocar en esas rutas. Dicho remanente es el que baja el promedio de la distancia. A continuación se detalla un día de la solución inicial de la instancia hidden_r500d10 que muestra esto:

Ruta 1	Ruta 2	Ruta 3	Ruta 4	Ruta 5	Promedio
99.12 %	98.12 %	97.00 %	99.83 %	37.09 %	86.23 %

Fig. 6.5: Ejemplos de porcentaje usado de rutas para la solución inicial de instancia hidden_r500d10

Como se puede ver en la Figura 6.5, las rutas están cerca del límite de distancia permitida. Este es un claro ejemplo donde una ruta (la ruta 5 en este caso) baja el promedio de distancia mientras que el resto está bastante por encima.

Curiosamente, si se quiere equiparar la distancia utilizada en las rutas puede ser que se lleguen a casos bastante peores. Por ejemplo, si se mueve un nodo de la ruta 1 a la ruta 5, la distancia de la primera va a mantenerse menor o igual pero la distancia en la ruta 5 puede crecer en mayor medida que la mejora de la ruta 1. Como ejemplo, si un camión recorre nodos por el norte de una ciudad, otro camión por el sur y uno de los nodos se pasa de un camión a otro, el promedio de distancia recorrida aumenta considerablemente.

6.2.3. Uso de Camiones

Un último caso a contemplar es el de los camiones en sí. Estos son infinitos a priori pero cada uno de ellos tiene un costo tanto de obtenerlos, como el de uso de cada día. En las instancias utilizadas, el costo de obtener un camión es bajo mientras que el uso de los camiones por día es alto en comparación a los camiones totales. Sin embargo, el costo de ambos es bastante más bajo que el de las herramientas y la distancia.

Por más que sus costos no impacten tanto, es interesante observar cuántos de los camiones son usados respecto a la cantidad de camiones obtenidos. Esto es, para cada instancia, para cada día $d \in D$ el porcentaje es Λ_d / Λ.

Debido a la forma en que se calcula el porcentaje, siempre va a existir por lo menos un día con el 100 % de los camiones utilizados. El porcentaje promedio general es de 70.42 %, lo cual muestra que las instancias no se preocupan en utilizar el 100 % de los camiones constantemente. Esto es esperado ya que el costo por día es más grande que el costo por adquirir un camión, y ambos son bajos respecto a los otros costos. A continuación se presenta un ejemplo detallado:

Día	Porcentaje de Camiones
1	100.00 %
2	71.43 %
3	35.71 %
4	50.00 %
5	50.00 %
6	50.00 %
7	57.14 %
8	35.71 %
9	71.43 %
10	100.00 %

Fig. 6.6: Ejemplos de porcentaje de camiones para la solución inicial de la instancia hidden_r500d10

En este ejemplo, se utilizan más camiones en el primer y último día. Estos días son especiales ya que en el primer día solo puedo realizar entregas y en el último solo recolecciones. Por esto, en esos días es imposible realizar enganches y se tiene que volver a depósito cuando el camión está vacío o lleno, respectivamente. En los días intermedios, se aprovecha a realizar enganches por lo que termina utilizando menos camiones en estos días.

6.3. Nomenclatura de las Variaciones Utilizadas

A las variaciones, detalladas en el Capítulo 5.5, se le asigna la siguiente nomenclatura:

- *Sin Variación*: No se utilizó ninguna variación
- *Ventana de Tiempo*: Utilización de ventanas de tiempo, en conjunto a *sliding windows*
- *Corte Media*: Utilización de la media de las distancias para filtrar vecinos
- *Corte Mediana*: Utilización de la mediana de las distancias para filtrar vecinos
- *Foco Par*: Al momento de elegir vecinos, se eligen de a pares entrega-recolección
- *Reducción por Días*: Reducción de la vecindad de cada secuencia por su cantidad de días posibles

6.4. Elección de Parámetros del algoritmo del Modelo de reubicación

En el Algoritmo 6 de la Sección 5.4 se detalló el ciclo general del uso del algoritmo del modelo de reubicación. Las distintas configuraciones obtenidas utilizando las variaciones comentadas en el Capítulo 5.5 tienen algunos parámetros en común.

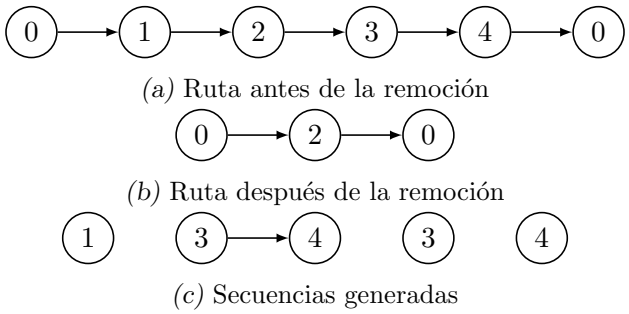
6.4.1. Generación de Secuencias

El primer parámetro que se va a detallar es la generación de secuencias. Una vez que se eligen los nodos a sacar, se procede a generar secuencias con ellos. Se decidió generar solamente secuencias de tamaño uno, es decir, las secuencias se componen de un solo nodo. Generar secuencias de tamaño uno ya produce que el modelo sea lo suficientemente grande como para que sea difícil resolverlo.

A modo de ejemplo, si se remueven n nodos, y se generan secuencias unitarias se tienen $O(n)$ secuencias. Si además se generan secuencias de tamaño dos, se tienen $O(n^2)$ secuencias. De esta forma, al crecer en uno el tamaño máximo de las secuencias, se aumenta considerablemente la cantidad de las mismas. Se nota que la cantidad total de secuencias tiene un impacto directo en el tamaño del modelo.

Además de estas secuencias, se recuerda que se generan las secuencias necesarias para poder reconstruir la solución inicial, sin importar el tamaño de las mismas. Esto permite que el modelo siempre pueda llegar a una solución mejor o igual a la que ya tenía, así como también poder devolver siempre una solución factible.

A continuación se presenta un ejemplo para mostrar la generación se secuencias:



(a) Ruta antes de la remoción

(b) Ruta después de la remoción

(c) Secuencias generadas

Fig. 6.7: Ejemplo donde se remueven los nodos 1, 3 y 4

En el ejemplo de arriba, se generan las secuencias de tamaño uno y una secuencia de tamaño dos que se corresponde a $3 \longrightarrow 4$. Se observa que se puede recuperar la ruta anterior colocando la secuencia 1 y la secuencia $3 \longrightarrow 4$ en sus lugares correspondientes.

Asimismo, se puede ver que todas las secuencias unitarias no entran en la ruta: hay dos *insertion point* y tres secuencias unitarias. Esto lleva al siguiente parámetro en común de las configuraciones: el porcentaje de nodos removidos.

6.4.2. Porcentaje de Nodos Removidos

Cuántos más nodos de la solución se remueven, menos *insertion point* disponibles hay. Esto se debe a que los *insertion points* son las aristas de la solución resultante luego de la remoción de los nodos elegidos.

Si se tienen ru rutas, nt nodos totales, se remueven nr ($nr \leq nt$) de los nodos, y

con *dep* indicando la cantidad de depósitos intermedios entre todas las rutas, se tienen $nt - nr + ru + dep$ *insertion point*. De esta forma, remover una cantidad grande de nodos perjudica al modelo ya que va a poder realizar una menor cantidad de recombinaciones dado que van a existir menos *insertion point*.

Se realizó una etapa de búsqueda de parámetros para poder concretar el porcentaje de nodos removidos. Como porcentajes posibles se utilizaron 10 %, 25 % y 40 %. Se llegó a la conclusión que utilizar 10 % provocaba que las recombinaciones sean pocas ya que los nodos removidos eran pocos, mientras que utilizar 40 % generaba lo mencionado anteriormente: obtiene poca variabilidad debido a la relación entre secuencias e *insertion points*.

Finalmente, 25 % resultó ser un buen punto medio ya que tenía los suficientes nodos removidos como para permitir muchas recombinaciones, así como también tenía una buena cantidad de *insertion points* como para permitir variabilidad.

6.4.3. Tiempo de Ejecución

Los algoritmos de los experimentos que se detallan en la Sección 6.5 fueron corridos cinco veces para cada instancia y para cada configuración, con un tiempo máximo de 300 segundos por corrida. Como el modelo de reubicación utilizado no es determinístico, cada una de estas corridas tiene asociado un numero escogido al azar que actúa como semilla.

Luego, se promedia el resultado de estas cinco corridas para obtener el resultado final. Esto se realiza debido a que un resultado puede ser particularmente bueno o malo debido a la semilla que le tocó. Por esto, realizar cinco corridas permite paliar el azar.

A continuación se explica brevemente la razón de la elección de los 300 segundos por corrida. Se corrieron las instancias por 30 minutos y se observó cómo se comporta el algoritmo a lo largo del tiempo. Para las configuraciones utilizadas se observan *diminishing returns*, es decir, cada vez menos mejora a lo largo que el tiempo de incrementa.

Aproximadamente entre un 40 % y un 50 % de la mejora se hace en los primeros 120 segundos, y un 60 %-70 % en los primeros 300 segundos (tomando la mejora a los 30 minutos como el 100 %). A continuación se presenta un ejemplo de dichos *diminishing returns*.

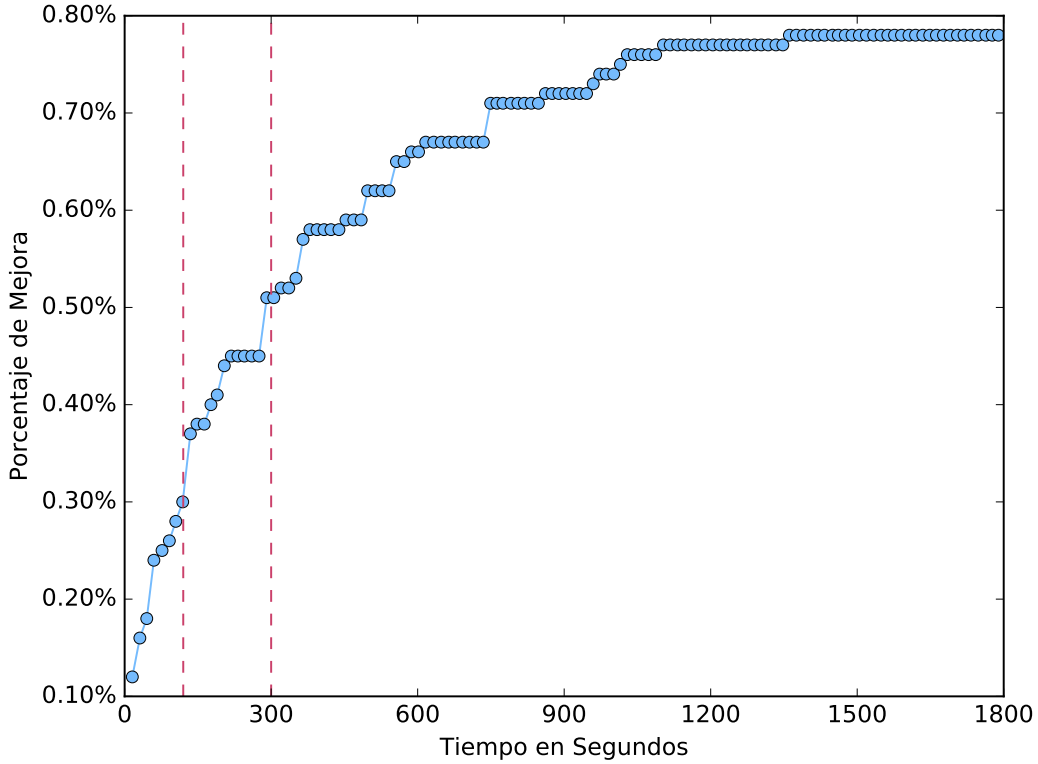


Fig. 6.8: Mejora del algoritmo del modelo de reubicación sobre la instancia late_r1000d20

En la Figura 6.8 se observa la mejora del algoritmo del modelo de reubicación en la instancia *late_r1000d20*, utilizando las variaciones de ventana de tiempo y corte por la mediana. Los puntos circulares en el gráfico indican una iteración del algoritmo del modelo, mientras que las líneas verticales punteadas marcan los 120 y 300 segundos respectivamente.

Como se puede observar, las mejoras son más importantes al inicio de la corrida. Para el final de la ejecución, las mejoras terminan siendo cada vez mas chicas, con saltos de la mejora cada vez menos frecuentes.

Tomar 300 segundos como tope brinda un número lo suficientemente grande como para probar la utilidad los algoritmos, y lo suficientemente chico como para ser rápido de correr. Se recuerda que este modelo de reubicación tiene el objetivo de extraer un porcentaje de mejora extra de una solución, por lo que si se corre por una cantidad de tiempo muy grande pierde parte de su atractivo.

6.4.4. Elección de la Ventana de Tiempo

Al utilizar la variación *Ventana de Tiempo*, explicada en la Sección 5.5, se tienen dos parámetros a ajustar: el ancho de la ventana de tiempo, y el corrimiento de la misma.

En cuanto al ancho de la ventana de tiempo, se probaron distintos tamaños. Utilizar una ventana de tiempo muy ancha va en contra del objetivo de utilizar esta variación. Por otro lado, usar una ventana de tiempo muy chica no permite cambiar los nodos de días y restringe las posibilidades de combinación.

Con el objetivo de realizar muchas ventanas de tiempo relativamente chicas, se decidió utilizar una ventana de tiempo de tamaño 5, comenzando en cada día posible. Una razón es que la diferencia de días entre entrega y recolección es menor a cinco.

Además, se realiza una corrida del algoritmo del modelo para cada día por lo que permite asegurar que siempre se pueda remover cada par entrega-recolección. Se recuerda que si se tiene un corrimiento lo suficientemente grande puede ocurrir que los pares entrega-recolección nunca se elijan a la vez. Finalmente, este tamaño estos parámetros son lo suficientemente chicos como para poder ejecutar varias corridas del algoritmo del modelo.

6.5. Resultados

A continuación se presentan los resultados de la experimentación realizada utilizando los operadores y el algoritmo que implementa el modelo de reubicación presentado. Un punto de comparación va a ser la diferencia del funcional antes y después de aplicar los algoritmos. Dicho porcentaje de mejora se calcula como $(c_{\text{original}} - c_{\text{nuevo}})$ / c_{original}, donde c_{original} y c_{nuevo} son los valores del funcional antes y después de aplicar los algoritmos, respectivamente.

6.5.1. Operadores

A continuación se compararan las tres distintas configuraciones de operadores que se detallaron en la Sección 4.5. En la siguiente figura, se tienen tiempos de ejecución que se miden en segundos.

Instancia	Ad-hoc		Nodos, Rutas		Distinto Día, Mismo Día	
	Tiempo	Mejora	Tiempo	Mejora	Tiempo	Mejora
late_r1000d20	4.15	0.17 %	7.23	**0.23 %**	4.30	0.21 %
late_r1000d25	3.55	0.07 %	7.97	**0.23 %**	4.85	0.21 %
late_r1500d30	6.13	0.08 %	14.41	0.22 %	16.26	**0.23 %**
late_r2000d50	7.29	0.12 %	11.31	**0.22 %**	11.37	**0.22 %**
late_r2000d65	4.03	0.11 %	4.91	0.12 %	4.74	**0.13 %**
hidden_r500d10	2.27	0.10 %	2.44	0.22 %	4.00	**0.31 %**
hidden_r500d15	1.23	**0.37 %**	1.37	0.35 %	1.35	0.28 %
hidden_r1000d20	4.02	0.28 %	5.79	**0.37 %**	5.75	0.35 %
hidden_r1000d25	3.01	0.10 %	7.96	0.34 %	14.01	**0.52 %**
hidden_r1500d30	7.21	0.16 %	16.78	**0.41 %**	16.68	**0.41 %**
hidden_r1500d40	3.54	0.04 %	3.75	**0.06 %**	3.75	**0.06 %**
hidden_r2000d50	7.60	0.05 %	16.77	**0.18 %**	13.43	**0.18 %**
hidden_r2000d65	5.47	0.13 %	11.69	**0.21 %**	9.41	**0.21 %**
hidden_r2500d70	7.19	0.13 %	7.96	**0.15 %**	8.01	**0.15 %**
hidden_r2500d75	4.06	0.11 %	11.03	**0.17 %**	8.79	**0.17 %**
Promedio	4.72	0.13 %	8.76	0.23 %	8.45	**0.24 %**

Fig. 6.9: Tiempo de ejecución y porcentaje de mejora de las distintas configuraciones de operadores

Un detalle a notar es que los operadores convergen a una solución en cuestión de segundos. Dada la naturaleza de los operadores que se manejan, al llegar a un óptimo local no pueden salir de éste.

La configuración *Ad-hoc* es considerablemente peor a las demás en porcentaje de mejora, ya que siempre tiene una solución peor que las otras dos configuraciones. Asimismo, no tiene un tiempo de ejecución considerablemente mejor como para justificar utilizarlo frente a las demás.

Entre *Nodos, Rutas* y *Distinto Día, Mismo Día* no hay una diferencia clara: logran una cantidad de mejoras similar, y el tiempo de ejecución también lo es. Hay casos como hidden r500d15 donde *Nodos, Rutas* logra una mayor mejora (0.35 % vs 0.28 %) pero también hay otros donde *Distinto Día, Mismo Día* es mejor como hidden_r1000d25 (0.34 % vs 0.52 %).

6.5.2. Variaciones de Manera Unitaria

En esta sección se detallan las distintas variaciones del algoritmo del modelo de reubicación, y como afectan al funcional utilizándose de manera unitaria. Esto es, una única variación a la vez. Se utiliza la nomenclatura establecida en la Sección 6.3. Los resultados que se van a discutir a continuación se encuentran en la Figura 6.10.

Cada modelo fue corrido 300 segundos, y en las tablas se van a comparar las variaciones en cantidad de iteraciones. Esto es, la cantidad de veces que se resolvió un problema de programación lineal entera en dichos 300 segundos.

Instancia	Sin Variación		Ventana de Tiempo		Corte Media		Corte Mediana		Foco Par		Reducción por Días	
	Iter.	Mejora	Iter.	Mejora	Iter.	Mejora	Iter.	Mejora	Iter.	Mejora	Iter.	Mejora
late_r1000d20	45.4	0.37 %	227.2	0.39 %	62.2	**0.42 %**	63.2	0.40 %	10.4	0.27 %	57.6	0.38 %
late_r1000d25	77.2	0.34 %	445.2	0.31 %	103.8	0.27 %	102.0	0.32 %	20.4	**0.35 %**	95.2	0.33 %
late_r1500d30	46.8	0.30 %	327.6	0.28 %	61.6	**0.40 %**	61.2	0.36 %	4.6	0.14 %	56.6	0.36 %
late_r2000d50	18.8	0.20 %	276.0	0.20 %	26.2	0.20 %	27.2	0.23 %	1.4	0.04 %	25.0	**0.24 %**
late_r2000d65	40.2	0.18 %	671.0	0.23 %	55.0	0.25 %	56.8	**0.27 %**	14.4	0.19 %	53.0	0.24 %
hidden_r500d10	57.0	0.57 %	136.8	0.58 %	70.6	0.55 %	73.2	0.54 %	8.2	0.22 %	72.2	**0.67 %**
hidden_r500d15	210.0	0.69 %	690.8	0.62 %	282.4	**0.72 %**	288.8	0.63 %	52.8	0.68 %	270.4	0.67 %
hidden_r1000d20	37.6	0.51 %	192.0	0.52 %	46.2	0.51 %	47.2	**0.53 %**	2.4	0.08 %	47.0	0.51 %
hidden_r1000d25	56.8	0.38 %	319.2	0.32 %	74.6	0.46 %	75.2	0.45 %	12.8	**0.56 %**	70.2	0.42 %
hidden_r1500d30	17.4	0.32 %	182.0	0.39 %	23.4	0.35 %	24.4	**0.40 %**	1.0	0.03 %	23.2	0.36 %
hidden_r1500d40	99.8	0.17 %	907.2	0.20 %	146.4	0.20 %	146.0	**0.21 %**	37.2	0.16 %	130.4	0.19 %
hidden_r2000d50	40.6	0.17 %	432.4	0.11 %	55.4	0.20 %	55.8	**0.21 %**	13.2	0.20 %	50.8	0.19 %
hidden_r2000d65	51.8	**0.24 %**	549.0	0.21 %	72.2	0.23 %	71.4	**0.24 %**	15.6	0.21 %	67.2	0.23 %
hidden_r2500d70	27.0	0.24 %	396.0	0.21 %	35.6	0.22 %	36.0	**0.25 %**	6.8	0.15 %	35.4	0.20 %
hidden_r2500d75	32.2	**0.23 %**	639.0	0.19 %	41.8	**0.23 %**	42.2	**0.23 %**	11.6	0.14 %	40.4	**0.23 %**
Promedio	57.24	0.33 %	426.09	0.32 %	77.16	**0.35 %**	78.04	**0.35 %**	14.19	0.23 %	72.97	**0.35 %**

Fig. 6.10: Comparación de iteraciones y mejora de las variaciones unitarias del algoritmo del modelo de reubicación

En primera medida, se puede observar que la variante *Foco Par* es la única variante donde se ejecutan menos iteraciones que *Sin Variación*. Esto se debe a que en esta variación se eligen los nodos de a pares entrega-recolección, lo cual permite que cada nodo elegido pueda moverse de día. Como cada par de nodos puede moverse de día, sus vecindarios crecen en base a la cantidad de días posibles, lo cual hace crecer exponencialmente la cantidad de combinaciones que generan que resolver el modelo sea muy costoso.

En el otro lado del espectro, está la variación *Ventana de Tiempo*, que tiene una cantidad mucho más grande de iteraciones respecto a las otras configuraciones. Debido a que la ventana de tiempo es más chica, resolver el modelo también lo es. Tomando como ejemplo a *late_r1000d20*, se usan ventanas de tiempo 4 veces más chicas (5 vs. 20) lo cual permite resolver aproximadamente 5 veces más modelos.

En cuanto al porcentaje de mejora, se puede observar que las variaciones suelen tener mejores resultados que no utilizar variación. La variante *Foco Par* en varios casos es peor que no utilizar variaciones como en *late_r1500d30* donde logra la mitad de la mejora respecto a *Sin Variación*. Por otro lado, cabe destacar que la variante *Foco Par* no es totalmente inútil. Usando como ejemplo a *hidden_r1000d25* en donde dicha variante logró el mejor porcentaje de mejora incluso realizando menos iteraciones que las demás.

Comparando *Ventana de Tiempo* contra *Corte Mediana*, se observa que esta última es mejor en todos los casos salvo en *hidden_r500d10*. En general, esta tendencia se mantiene: las variaciones de la elección de nodos (*Ventana de Tiempo* y *Foco Par*) suelen ser peores que las de reducción de vecindario (*Corte Media*, *Corte Mediana* y *Reducción por Días*) al ser usadas en forma unitaria.

Esto es importante ya que no siempre resolver más modelos implica llegar a una mejor solución. Por ejemplo, en *hidden_r500d15* la variación *Ventana de Tiempo* tiene un peor resultado que *Sin Variación*, realizando tres veces más iteraciones. Esto puede suceder si hay iteraciones que no pueden mejorar, o que las variaciones que tienen menos iteraciones puedan mejorar más por iteración, por ejemplo.

Esta mejora de las variaciones de reducción de vecindario es un punto importante: realizarlas muestra la posibilidad de correr más iteraciones del algoritmo del modelo, restringiendo soluciones posibles. Además, se llega a mejores resultados que sin haber restringido los vecindarios, lo cual indica que estas reducciones son valiosas no solo en cantidad de iteraciones sino también en cuanto a la solución final.

También se pueden comparar estos resultados contra los de los operadores, que se encuentran más arriba en la Figura 6.9. Se puede observar que utilizar el modelo por 300 segundos ya suele ser mejor que operadores, incluso sin usar variaciones.

Además, al utilizar variaciones estas mejoras suelen ser más consistentes. Cabe aclarar que la variación *Foco Par* termina siendo la peor de las variaciones unitarias, incluso teniendo varios casos donde es peor que utilizar operadores.

Se recuerda que la corrida de los modelos son de 300 segundos, mientras que los operadores convergen en menos de 20 segundos. De esta forma, realizar el modelo de reubicación termina llegando a una solución mejor a costa de utilizar más tiempo de ejecución.

6.5.3. Variaciones Combinadas

A continuación, se procede a comparar la combinación de *Foco Par* con la de *Reducción por Días*. El mayor problema de la variación *Foco Par* recae en que al sacar los pares entrega-recolección, los vecindarios crecen en una gran medida debido a los días posibles en los que se pueden insertar la secuencia. Justamente, la variante *Reducción por Días* ataca a este problema en particular por lo que resulta interesante observar que es lo que sucede cuando se las utiliza en conjunto.

En la Figura 6.11, además de mostrar los resultados de dicha combinación (llamada *Ambos* para esta sección), se repite información de *Foco Par* y *Reducción por Días* que se encuentra en la Figura 6.10 para la comodidad del lector.

Instancia	Foco Par		Reducción por Días		Ambos	
	Iter.	Mejora	Iter.	Mejora	Iter.	Mejora
late_r1000d20	10.4	0.27 %	57.6	**0.38 %**	21.8	0.33 %
late_r1000d25	20.4	0.35 %	95.2	0.33 %	36	**0.44 %**
late_r1500d30	4.6	0.14 %	56.6	**0.36 %**	8.4	0.20 %
late_r2000d50	1.4	0.04 %	25	**0.24 %**	3.4	0.06 %
late_r2000d65	14.4	0.19 %	53	**0.24 %**	27.4	0.19 %
hidden_r500d10	8.2	0.22 %	72.2	**0.67 %**	13.8	0.33 %
hidden_r500d15	52.8	0.68 %	270.4	0.67 %	97.4	**0.73 %**
hidden_r1000d20	2.4	0.08 %	47	**0.51 %**	2.2	0.11 %
hidden_r1000d25	12.8	**0.56 %**	70.2	0.42 %	16.4	0.55 %
hidden_r1500d30	1	0.03 %	23.2	**0.36 %**	1.2	0.05 %
hidden_r1500d40	37.2	0.16 %	130.4	0.19 %	62	**0.20 %**
hidden_r2000d50	13.2	0.20 %	50.8	0.19 %	21.4	**0.25 %**
hidden_r2000d65	15.6	0.21 %	67.2	0.23 %	28.4	**0.26 %**
hidden_r2500d70	6.8	0.15 %	35.4	**0.20 %**	14.2	0.19 %
hidden_r2500d75	11.6	0.14 %	40.4	**0.23 %**	19.8	0.22 %
Promedio	14.19	0.23 %	72.97	**0.35 %**	24.92	0.27 %

Fig. 6.11: Comparación de número de iteraciones y porcentaje de mejora de las distintas combinaciones de *Foco Pares* y *Reducción por Días* del algoritmo del modelo de reubicación

En un primer momento, resulta interesante comparar *Foco Par* contra *Ambos* ya que era el objetivo de este experimento. Efectivamente, utilizar la reducción por días ayuda a mejorar las soluciones del algoritmo del modelo. Se mejoran tanto la cantidad de iteraciones como la calidad de las mismas. En cuanto a la mejora de las soluciones, una única instancia es peor (*hidden_r1000d25*) y aún cuando lo es, es por un margen muy chico.

Las instancias que no mejoraron en cantidad de iteraciones fueron *hidden_r1000d20 hidden_r1500d30* que tuvieron tan pocas corridas del algoritmo del modelo que no llegan a poder mejorar lo suficiente en términos de tiempo. Además, cabe aclarar que en ambos casos, se logra una mejor solución. Así, resulta evidente que agregar la *Reducción por Días* a *Foco Par* lo ayuda en una gran manera.

Otra forma de ver los resultados es considerar que la variante *Foco Par* se agregó a *Reducción por Días*. Respecto a la cantidad de iteraciones, bajan en todas las instancias por lo menos a la mitad. Esto se debe a que el "peso" que le agrega utilizar *Foco Par* al tiempo de ejecución no se logra compensar utilizando *Reducción por Días*.

Por más que tenga menos iteraciones, hay casos donde *Ambos* obtiene la mejor solución como por ejemplo *late_r1000d25* y *hidden_r500d15*. Estos dos casos muestran que si bien se tienen menos cantidad de iteraciones, los pocos modelos resueltos impactan de gran manera al funcional.

Del otro lado, *late_r1000d50* y *late_r2000d65* muestran que *Reducción por Días* sigue logrando una mejor solución sin utilizar *Foco Par*. En estos dos casos, la cantidad de iteraciones de *Ambos* es lo suficientemente alta como para compararlos de par en par. Asimismo, se tienen casos como *hidden_r1500d30* donde utilizar *Foco Par* impacta fuertemente en la cantidad de iteraciones y por ende *Reducción por Días* obtiene una solución muy superior a las demás.

En términos de cantidad de iteraciones, al contrastar con no utilizar variaciones (ver la Figura 6.10), *Ambos* sigue teniendo menos. Esto se debe a que si bien los vecindarios se achicaron al mismo tamaño que *Sin Variación*, las combinaciones entre los distintos nodos son más grandes.

En *Sin Variación,* un nodo se retira con su par solo cuando ambos son elegidos, lo cual no sucede a menudo[2]. Cuando se retira un solo nodo del par, el nodo removido (ya sea entrega o recolección) no puede cambiarse de día por lo que no se combina con otros nodos de otros días. Usando *Foco Par*, se sacan siempre de a pares por lo que todos los nodos se pueden mover de días. Esta combinación de nodos de distintos días es la que produce que el modelo tarde mas tiempo en resolverse, lo cual influye directamente en la cantidad de iteraciones.

Por lo hablado en esta sección, se procede a juntar *Ambos* con las variaciones faltantes, con el objetivo de mejorar en cantidad de iteraciones así como también el resultado final.

6.5.4. Juntando Variaciones

A continuación, se lleva la idea de combinar variaciones al extremo juntando todas las variaciones posibles. Nuestras variaciones son todas compatibles con todas salvo *Corte Media* con *Corte Mediana*. Ambas tienen el mismo objetivo: reducir vecindarios por un valor dado por secuencia y no se van a utilizar en conjunto. Además, no utilizarlas a la vez nos da un punto de comparación entre los dos cortes.

El siguiente experimento compara los distintos cortes entre sí, donde cada una de las configuraciones incluye a *Ventana de Tiempo*, *Foco Par*, y *Reducción por Días*.

[2]La probabilidad que se saquen ambos es de $25\,\% \times 25\,\% = 6.25\,\%$, ya que su elección es independiente

Instancia	Todas Variaciones Corte Media		Todas Variaciones Corte Mediana	
	Iter.	Mejora	Iter.	Mejora
late_r1000d20	441.6	0.49 %	454.4	**0.51 %**
late_r1000d25	491.4	**0.59 %**	495.6	**0.59 %**
late_r1500d30	312.0	**0.52 %**	312.0	0.51 %
late_r2000d50	414.0	0.28 %	414.0	**0.29 %**
late_r2000d65	1220.0	0.28 %	1256.6	**0.30 %**
hidden_r500d10	162.0	**0.54 %**	169.2	0.51 %
hidden_r500d15	957.0	0.68 %	959.2	**0.71 %**
hidden_r1000d20	227.2	0.63 %	233.6	**0.72 %**
hidden_r1000d25	361.2	1.12 %	357.0	**1.14 %**
hidden_r1500d30	187.2	**0.56 %**	192.4	**0.56 %**
hidden_r1500d40	1360.8	0.20 %	1339.2	**0.21 %**
hidden_r2000d50	487.6	**0.23 %**	460.0	0.22 %
hidden_r2000d65	671.0	0.22 %	671.0	**0.23 %**
hidden_r2500d70	541.2	0.22 %	594.0	**0.23 %**
hidden_r2500d75	937.2	0.26 %	994.0	**0.27 %**
Promedio	584.76	0.45 %	593.48	**0.47 %**

Fig. 6.12: Comparación de número de iteraciones y porcentaje de mejora, juntando las variaciones del algoritmo del modelo de reubicación, para los distintos cortes

Como se puede ver en la Figura 6.12, al utilizar uno u otro corte se obtienen resultados similares tanto en cantidad de iteraciones como en porcentaje de mejora. Sin embargo, utilizar el corte por la mediana suele brindar mejores resultados aunque esta mejora es relativamente chica.

Por otro lado, la cantidad de iteraciones no parece tener una relevancia alta al momento de decidir entre estos dos cortes. Hay casos donde la mediana tenia más iteraciones y un mejor resultado (*late_r1000d20*) y otros donde tenía menos iteraciones pero una mejora más grande (*hidden_r150040*). Asimismo, hay casos donde la media tuve mejores resultados teniendo más y menos iteraciones (*hidden_r200050* y *hidden_r50010*, respectivamente).

El objetivo principal de estas variaciones era poder hacer más iteraciones, manteniendo el valor del funcional final lo menos degradado posible. No sólo se logró tener muchas más iteraciones en la misma cantidad de tiempo, sino que también comparando contra *Sin Variación* ambas suelen producir mejores resultados.

Un caso a destacar es el *hidden_r1000d25* donde las configuraciones *Todas Variaciones Corte Media* y *Todas Variaciones Corte Mediana* obtuvieron 1.12 % y 1.14 % de mejora, respectivamente. En comparación, *Sin Variación* produce una mejora de 0.38 %. Esto significa que estas configuraciones pudieron obtener el triple de mejora, en la misma cantidad de tiempo.

6.5.5. Sustrayendo Variaciones en el Modelo

Una vez que se tenía esta solución se propuso observar si alguna variación influye negativamente en el valor final. Se recuerda que esto sucedía en el ejemplo de la Sección 6.5.3 donde utilizar *Foco Par* con *Reducción por Días* producía un resultado peor contra usar solo *Reducción por Días*.

Al remover una de estas variaciones podemos observar que sucedería en caso de no tenerlas. Así, también se puede pensar que se está observando el caso donde se agregan dichas variaciones en vez de quitarlas. En este contexto, podemos observar cuáles variaciones influyen positivamente al ser agregadas en cantidad de iteraciones y mejora del valor final, observando aquellas que afectan negativamente al sacarse.

A continuación se presenta dicho experimento, utilizando como base a *Todas Variaciones Corte Mediana*. En la Figura 6.13 se utilizan los nombres de las variaciones con el prefijo "sin" para indicar la componente faltante.

En términos de cantidad de iteraciones, la única variación que las aumenta es *Foco Par*. Esto es coherente con lo que se vino discutiendo ya que usar *Foco Par* aumenta la cantidad de combinaciones. El resto de las variaciones las disminuye consistentemente.

La que más afecta es *Ventana de Tiempo*. Al observar todo el horizonte de tiempo en un modelo provoca que los nodos que se saquen sean muchos. Para mostrar un ejemplo, en *late_r2000d65* se sacan 500 nodos de entrega y sus pares asociado llevándolo a tener que reacomodar 1000 nodos. Además como se sacan de a pares estos nodos pueden ser reubicados en distintos días, lo cual también afecta a la cantidad de combinaciones y por ende la cantidad de iteraciones. Usando el mismo ejemplo, al sacarse las ventanas de tiempo las iteraciones bajan *late_r2000d65* de 1256.6 a 64.4: aproximadamente 19 veces menos.

Una comparación curiosa es la de *Sin Corte Mediana* y *Sin Reducción por Dias* ya que ambas reducen el vecindario posible. Al no tener corte por la mediana, se reduce el vecindario únicamente por la cantidad de días en las cuales la secuencia puede colocarse. Asimismo, al no reducir por días se reducen los vecindarios por la mediana, es decir, se deja el 50 % más cercano.

Si tenemos en cuenta que en nuestras instancias los pedidos siempre tienen una distancia de por lo menos dos días entre la entrega y la recolección, se deja un porcentaje menor o igual al 50 % más cercano. Luego, el vecindario que deja *Sin Corte Mediana* es contenido en el de *Sin Reducción por Dias*. Esto se evidencia en la tabla, donde al sacar la reducción por días se tiene una mayor caída de las iteraciones que sacando el corte por la mediana.

En cuanto a la mejora, comparando estas dos últimas variaciones entre sí se puede ver que *Sin Corte Mediana* suele tener mejores resultados. Esto se debe a que si bien el vecindario de *Sin Reducción por Dias* es más grande, las posibilidades que se agregan al vecindario no ayudan tanto como para compensar lo que entorpecen al ser agregadas.

Instancia	Todas Corte Mediana		Sin Ventana de Tiempo		Sin Corte Mediana		Sin Foco Par		Sin Reducción por Días	
	Iter.	Mejora	Iter.	Mejora	Iter.	Mejora	Iter.	Mejora	Iter.	Mejora
late_r1000d20	454.4	**0.51 %**	63.4	0.43 %	192.0	0.47 %	412.8	**0.51 %**	134.4	0.42 %
late_r1000d25	495.6	**0.59 %**	68.0	0.55 %	285.6	0.56 %	722.4	0.30 %	231.0	**0.59 %**
late_r1500d30	312.0	**0.51 %**	18.2	0.31 %	208.0	0.49 %	520.0	0.35 %	156.0	**0.51 %**
late_r2000d50	414.0	**0.29 %**	16.4	0.20 %	230.0	0.25 %	460.0	0.23 %	147.2	0.21 %
late_r2000d65	1256.6	0.30 %	64.4	0.28 %	610.0	**0.31 %**	1171.2	0.26 %	427.0	0.27 %
hidden_r500d10	169.2	0.51 %	48.0	0.52 %	79.2	**0.59 %**	235.2	0.57 %	54.0	0.57 %
hidden_r500d15	959.2	0.71 %	237	0.74 %	497.2	**0.91 %**	1117.6	0.59 %	366.7	0.83 %
hidden_r1000d20	233.6	**0.72 %**	18.8	0.41 %	115.2	0.56 %	300.8	0.58 %	90.7	0.48 %
hidden_r1000d25	357.0	**1.14 %**	39.4	0.85 %	210.0	1.11 %	537.6	0.44 %	168.0	0.83 %
hidden_r1500d30	192.4	**0.56 %**	1.4	0.04 %	104.0	0.53 %	312.0	0.46 %	78.0	0.48 %
hidden_r1500d40	1339.2	0.21 %	142.6	0.20 %	770.4	0.20 %	1476.0	**0.22 %**	576.0	0.21 %
hidden_r2000d50	460.0	0.22 %	37.8	0.23 %	322.0	**0.26 %**	644.0	0.09 %	230.0	0.25 %
hidden_r2000d65	671.0	0.23 %	61.8	0.23 %	451.4	**0.26 %**	793.0	0.18 %	366.0	**0.26 %**
hidden_r2500d70	594.0	0.23 %	43.0	**0.28 %**	330.0	0.22 %	594.0	0.21 %	264.0	0.23 %
hidden_r2500d75	994.0	**0.27 %**	40.2	0.26 %	497.0	0.24 %	994.0	0.20 %	355.0	0.23 %
Promedio	593.48	**0.47 %**	60.03	0.37 %	326.80	0.46 %	686.04	0.35 %	242.93	0.42 %

Fig. 6.13: Comparación de iteraciones y mejora de las variaciones del algoritmo del modelo de reubicación, removiendo las variaciones de a una, y utilizando el *Corte Mediana*

Comparando las distintas combinaciones contra *Todas Corte Mediana*, esta última suele tener el mejor resultado, lo cual indica que todas las variaciones suelen influir positivamente. Un contraejemplo a contemplar es *hidden_r500d15*. En él, tanto *Sin Corte Mediana* como *Sin Reducción por Dias* logran tener mejores resultados por un margen considerable. En este caso, al usar el vecindario tan restringido en *Todas Corte Mediana* resultó ser contraproducente ya que se cortaron soluciones que hubieran servido.

En cuanto a *Sin Ventana de Tiempo* y *Sin Foco Par* están en situaciones opuestas en cuanto a la cantidad de iteraciones, pero similares en cuanto a porcentaje de mejora. Ambas suelen ser peor que utilizar todas las combinaciones, pero tienen algunos casos donde son mejores (*hiddenr2500d70* para la primera y *hidden_r500r10* para la segunda). También tienen casos donde producen una solución bastante peor en casos como *late_r1500d30* para *Sin Ventana de Tiempo*, *late_r1000d25* para *Sin Foco Par*, y *hidden_r1000d25* para ambos.

Un caso a destacar es que en la Sección 6.5.3 se observó como utilizar solamente *Reducción por Días* era mejor que utilizar esta variación combinada con *Foco Par*. Sin embargo, se puede ver que ahora que se tienen todas las combinaciones, no utilizar *Foco Par* suele producir peores resultados.

Esto se debe a que utilizar únicamente esas dos variaciones no permitía tener muchas iteraciones por lo que *Foco Par* terminaba entorpeciendo más de lo que ayudaba. Al tener todas las variaciones juntas *Foco Par* brilla al permitir lograr combinaciones de mayor calidad, sin reducir la cantidad de iteraciones a un número significativamente menor.

Los experimentos planteados en esta sección también se realizaron utilizando la media en vez de la mediana como corte. Sus resultados pueden ser observados en la Figura B.1 del Apéndice B. Dichos resultados son análogos a utilizar la mediana y se exponen por completitud.

6.5.6. Comparaciones Entre Categorías

A continuación se comparan las distintas categorías de configuraciones, llamadas *Categoría Operadores*, *Categoría Sin Variación*, *Categoría Todas Variaciones*. Para *Categoría Sin Variación* tenemos una única opción que es utilizar a la variación *Sin Variación*, pero para las otras dos hay más de una.

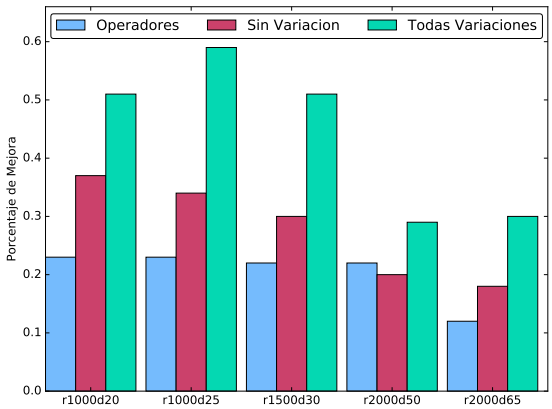
Para no mostrar resultados similares que no sumen a los gráficos, se decidió elegir un único representante para las categorías *Categoría Operadores*, y *Categoría Todas Variaciones*. Nótese que estas configuraciones están encerradas por comillas para diferenciarlas de las que utilizan itálica, y mostrar que representan conjuntos de configuraciones en vez de una configuración en particular.

Se recuerda que en la Sección 6.5.1 se vio que los resultados de operadores fueron similares para *Nodos, Rutas* y *Distinto Día, Mismo Día*, siendo ambas superior a *Ad-hoc*. Para elegir un representante de esta categoría, se decidió por *Nodos, Rutas*.

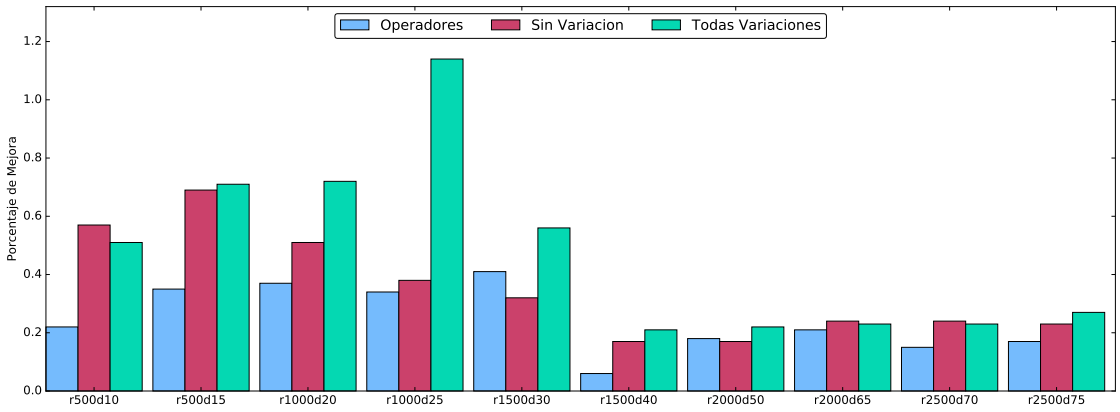
En la Sección 6.5.4 se observó que las configuraciones *Todas Variaciones Corte Media* y *Todas Variaciones Corte Mediana* eran similares. Para elegir un representante de esta categoría, se decidió por *Todas Variaciones Corte Mediana*.

En los siguientes gráficos se van a separar en las instancias *Late* y *Hidden*. Al tener esta separación, los gráficos van a obviar el prefijo "late_" y "hidden_" al nombrar las instancias.

También, se recuerda que los operadores corren hasta converger (y lo hacen en menos de 30 segundos) mientras que los algoritmo del modelo corren por 300 segundos.



(a) Instancias Late



(b) Instancias Hidden

Fig. 6.14: Comparación general de categorías en las instancias Late y Hidden

En estas comparaciones, se va a hablar de mejora relativa. Esto es, si una solución logro el 0.5 % de mejora y otra el 1.0 % se va a decir que la segunda tuvo una mejora del 100 % respecto a la primera.

En la Figura 6.14a se pueden observar los resultados pertinentes a las instancias *Late*. Comparando *Categoría Todas Variaciones* contra "Sin Variación", se puede ver como el primero tiene una mejora de aproximadamente 50 % en las soluciones. La diferencia es

menos notable en algunas soluciones, pero la mejora de utilizar las variaciones es clara.

Comparando *Categoría Todas Variaciones* contra *Categoría Operadores* la diferencia es aún más clara, llegando hasta a obtener más del 100 % de mejora en *late_r1000d25*. La diferencia es más chica en *late_r2000d50*, pero aún es una diferencia importante ya que es aproximadamente un 50 % de mejora respecto a *Categoría Operadores*.

"Sin Variación" versus *Categoría Operadores* es una comparación más reñida. Se separa un 50 % en las soluciones salvo en *late_r2000d50* donde los operadores logran obtener más mejoras. Esto se debe a que en esta instancia, "Sin Variación" logró un promedio de 18.8 iteraciones lo cual es un número bastante bajo. A modo de referencia, para esta instancia *Categoría Todas Variaciones* logró hacer 414 iteraciones.

En cuanto a las instancias *Hidden*, observadas en la Figura 6.14b, se van a separar en tres conjuntos que observan características similares:

- Chicas: hasta 15 días inclusive
- Medianas: hasta 30 días inclusive
- Grandes: hasta 75 días inclusive

En las instancias chicas, se ven que ambas configuraciones del algoritmo del modelo son superiores a utilizar operadores. Sin embargo, utilizar las variaciones no suele ser fructuoso y cuando lo es, tampoco es por mucha diferencia.

En las instancias medianas, se pueden ver como brillan las variaciones. Tenemos diferencias de por lo menos 50 % respecto a "Sin Variación" y 100 % respecto a *Categoría Operadores*. En este conjunto está *hidden_r1000d25* donde se obtuvo la mejora más amplia: un 200 %.

En las instancias grandes vemos cuando el modelo sufre más. Al ser instancias más grandes le cuesta más despegarse de "Sin Variación" y cuando lo hace no es por mucho. Incluso hay dos instancias donde no utilizar variaciones es mejor. Respecto a *Categoría Operadores Categoría Todas Variaciones* siempre es mejor, y "Sin Variación" también lo es salvo en *hidden_r2000d50*.

6.5.7. Aumentando el Tiempo de Ejecución

Como ya se vio en la Sección 6.4.3, se observan *diminishing returns* al momento de correr el modelo. A continuación se pone a prueba la variación *Todas Variaciones Corte Mediana*, comparando los valores a 120, 300 y 1800 segundos.

Para lograr estos valores, se corrió para cada instancia cinco veces y se promedió el mejor valor conseguido hasta ese momento de tiempo. En la Tabla 6.15 se pueden observar los resultados.

Instancia	Todas Variaciones Corte Mediana		
	120s	300s	1800s
late_r1000d20	0.34 %	0.48 %	0.75 %
late_r1000d25	0.42 %	0.53 %	0.83 %
late_r1500d30	0.34 %	0.52 %	0.81 %
late_r2000d50	0.20 %	0.29 %	0.43 %
late_r2000d65	0.20 %	0.29 %	0.44 %
hidden_r500d10	0.45 %	0.57 %	0.69 %
hidden_r500d15	0.52 %	0.66 %	0.81 %
hidden_r1000d20	0.46 %	0.69 %	1.23 %
hidden_r1000d25	0.75 %	1.16 %	2.23 %
hidden_r1500d30	0.37 %	0.64 %	1.08 %
hidden_r1500d40	0.15 %	0.22 %	0.29 %
hidden_r2000d50	0.15 %	0.26 %	0.44 %
hidden_r2000d65	0.17 %	0.21 %	0.33 %
hidden_r2500d70	0.14 %	0.20 %	0.40 %
hidden_r2500d75	0.20 %	0.28 %	0.44 %
Promedio	0.31 %	0.45 %	0.72 %

Fig. 6.15: Comparación de porcentaje de mejora de a tramos, utilizando todas las variaciones y el corte por mediana

Si bien los valores cambian instancia a instancia, su evolución dentro de las mismas suele ser similar.

Comparando los valores a los 120 segundos contra los de 300, se logra aproximadamente un 45 % de mejora. Por ejemplo, *hidden_r2500d75* pasa de 0.20 % a 0.28 %. Esa ganancia se produjo al correr 180 segundos más el modelo, que se corresponden a un 150 % (1,5 veces) más de tiempo de corrida.

Comparando los valores a los 300 segundos contra los de 1800, se logra aproximadamente un 60 % de mejora. Por ejemplo, *late_r1000d25* pasa de 0.53 % a 0.83 %. Esa ganancia se produjo al correr 1500 segundos más el modelo, que se corresponden a un 500 % (5 veces) más de tiempo de corrida.

Comparando las puntas, 120 contra 1800, se logra una mejora de 130 % extra con un tiempo adicional de 15 veces más.

De esta forma, si se está a 120 segundos y se quiere 45 % más, se tiene que hacer un 1,5 veces más el tiempo que ya se hizo. Si ahora se está en 300s y se quiere mejorar un 60 % tengo que hacer 5 veces más el tiempo que ya se hizo. Esto muestra que es cada vez mas difícil mejorar.

7. CONCLUSIONES Y TRABAJO A FUTURO

En esta tesis se aborda el problema planteado en el *VeRoLog solver challenge 2017* mediante un algoritmo de recombinación que utiliza técnicas de Programación Lineal Entera. Para ello, se toma como punto de partida la formulación propuesta en la tesis de Agustín Montero [10], que es modificada con el objetivo de aplicarla a un problema más complejo.

A su vez, se proponen distintas variaciones que refuerzan el modelo utilizado, mejorando la calidad de las soluciones obtenidas. Dichas variaciones producen una mejora considerable sin producir penalizaciones, por lo que es altamente recomendado usarlas.

Uno de los desafíos de este problema es el tamaño de las instancias, que llegan hasta los 5000 nodos repartidos a lo largo de 75 días. Se mostró que incluso en instancias grandes es fructífero utilizar el modelo, aunque no estén dentro del rango de mayor mejora.

Finalmente, esta tesis deja varias líneas de investigación a futuro abiertas, entre ellas:

- *Generar secuencias más largas*: Se utilizaron secuencias de tamaño uno, además de las secuencias iniciales. Una mejora posible yace en la generación de secuencias más grandes donde la posibilidad de recombinación aumenta.
- *Distinta elección de nodos*: Una variación utilizada fue remover los nodos de pares entrega-recolección. Se puede aplicar la misma idea utilizándola para remover los nodos que pertenezcan a una cierta ruta, con el objetivo de reacomodar los nodos y por ende utilizar un camión menos. Para las instancias utilizadas se aplicó esta elección de nodos, pero no fue fructuosa debido a que las rutas estaban muy restringidas por la distancia. Al agrandar artificialmente la distancia, se pudo observar mejoras considerables incluso agrandando la distancia máxima permitida tan poco como un 10 %.
- *Variaciones específicas para familias de instancias*: El modelo presentado permite utilizar numerosas variaciones. Una línea de investigación posible es concentrase en una familia de instancias particular, para las cuales se generan variaciones específicas que puedan explotar explícitamente la estructura de dichas familias.

Apéndice

A. NOTACIÓN UTILIZADA

Notación	Descripción	Introducida en
$\mathcal{D}$	Conjunto de días del horizonte de tiempo	3
G	Digrafo de pedidos, y los caminos entre los clientes	3
V	Conjunto de pedidos	3
E	Conjunto de caminos entre los clientes	3
t_v	Días de uso del pedido	3
$\mathcal{T}$	Tipos de herramientas	3
$\tau_{\max}$	Cantidad de stock máximo disponible para $\tau \in \mathcal{T}$	3
q^τ	Peso de cada herramienta de tipo $\tau \in \mathcal{T}$	3
q_v	Cantidad de herramientas requerida por el pedido $v \in V$	3
$\tau(v)$	Tipo de herramienta requerida por el pedido $v \in V$	3
Q	Capacidad máxima de cada vehículo	3
L	Distancia máxima diaria de cada vehículo	3
c^f	Costo fijo por contratar a cada camión	3
c^f_{dia}	Costo fijo por día de uso de cada camión	3
c^f_{dist}	Costo fijo por distancia recorrida	3
c^τ	Costo fijo de uso de cada herramienta de tipo $\tau \in \mathcal{T}$	3
$\mathcal{S}$	Conjunto de secuencias	5.1
$\mathcal{S}_d$	Conjunto de secuencias posibles para el día $d \in \mathcal{D}$	5.3
$\mathcal{D}_s$	Conjunto de días posibles a insertar la secuencia $s \in \mathcal{S}$	5.3
$\mathcal{R}$	Conjunto de rutas	5.3
$\mathcal{D}_r$	Día de la ruta $r \in \mathcal{R}$	5.3
$\mathcal{R}_d$	Rutas del día $d \in \mathcal{D}$	5.3
K_r	Conjunto de subrutas de la ruta $r \in \mathcal{R}$	5.3
r^k	Subruta de índice $k \in \{0, \dots, \lvert K_r\rvert - 1\}$ de la ruta $r \in \mathcal{R}$	5.3
$\mathcal{I}$	Conjunto de *insertion points*	5.1
$\mathcal{I}_r$	Conjunto de *insertion points* de $r \in \mathcal{R}$	5.3
$\mathcal{I}_{r^k}$	Conjunto de *insertion points* de la subruta r^k	5.3
r_i	Ruta del *insertion point* $i \in I$	5.3
$\overline{q}(s)$	Peso máximo del camión en la secuencia $s \in \mathcal{S}$	5.3.2
$\underline{q_\tau}(s)$	Stock requerido por la secuencia $s \in \mathcal{S}$ para cada tipo $\tau \in \mathcal{T}$	5.3.2
$\overline{q_\tau}(s)$	Diferencia del stock de $\tau \in \mathcal{T}$ entre el comienzo y el fin de $s \in \mathcal{S}$	5.3.2
$q^\tau_a(i)$	Diferencia del stock de $\tau \in \mathcal{T}$ hasta $i \in \mathcal{I}$, restringida a r_i	5.3.2
x_{si}	Igual a 1 si y solo si $s \in \mathcal{S}$ es asignada a $i \in I$	5.3.3
$w^{\tau,k}_r$	Herramientas de $\tau \in \mathcal{T}$ que salen del depósito $\#k$ de $r \in \mathcal{R}$	5.3.3
$y^{\tau,k}_r$	Herramientas de $\tau \in \mathcal{T}$ que llegan al depósito $\#k$ de $r \in \mathcal{R}$	5.3.4
$\eta^{\tau,k}_r$	Herramientas de $\tau \in \mathcal{T}$ sacadas del depósito $\#k$ de $r \in \mathcal{R}$	5.3.5
$\psi^{\tau,k}_r$	Herramientas a favor de $\tau \in \mathcal{T}$ del depósito $\#k$ de $r \in \mathcal{R}$	5.3.5
μ^d_τ	Stock que queda en el depósito de tipo $\tau \in \mathcal{T}$ en el día $d \in \mathcal{D}$	5.3.6
α^d_τ	Herramientas de $\tau \in \mathcal{T}$ que salen del depósito en $d \in \mathcal{D}$	5.3.6
ζ^d_τ	Herramientas de $\tau \in \mathcal{T}$ que llegan al depósito en $d \in \mathcal{D}$	5.3.6
z^{inicio}_{vd}	Igual a 1 si y solo si el pedido $v \in V$ comienza en el día $d \in \mathcal{D}$	5.3.7
z^{fin}_{vd}	Igual a 1 si y solo si el pedido $v \in V$ finaliza en el día $d \in \mathcal{D}$	5.3.7
$\mathcal{F}$	Conjunto de nodos extraídos, tanto entregas como recolecciones	5.1
$\mathcal{F}_{\text{ent}}$	Conjunto de nodos extraídos que son entregas	5.3.7
$\mathcal{F}_{\text{rec}}$	Conjunto de nodos extraídos que son recolecciones	5.3.7
$\mathcal{R}_d$	Conjunto de rutas del día $d \in \mathcal{D}$	5.3.7
δ'_r	Longitud de $r \in \mathcal{R}$ luego que se hayan asignado las secuencias	5.3.8
δ_r	Longitud de la ruta $r \in \mathcal{R}$	5.3.8
δ_s	Distancia de la secuencia $s \in \mathcal{S}$	5.3.8
$\delta(v_1, v_2)$	Distancia entre los clientes de los pedidos $v_1, v_2 \in V$	5.3.8
Ω	Conjunto de las rutas que pueden quedar vacías	5.3.10
ω_r	Igual a 0 si y solo si $r \in \Omega$ queda vacía	5.3.10
Λ	Cantidad de camiones máxima usados en el horizonte de días	5.3.11
Λ_d	Cantidad de camiones usados en el día $d \in \mathcal{D}$	5.3.11

Tab. A.1: Notación utilizada

B. RESULTADOS DE LAS VARIACIONES DEL MODELO DE REUBICACIÓN

Instancia	Todas Corte Media		Sin Ventana de Tiempo		Sin Corte Media		Sin Foco Par		Sin Reducción por Días	
	Iter.	Mejora	Iter.	Mejora	Iter.	Mejora	Iter.	Mejora	Iter.	Mejora
late_r1000d20	441.6	**0.49 %**	58.8	0.41 %	192.0	0.47 %	403.2	0.46 %	131.2	0.39 %
late_r1000d25	491.4	**0.59 %**	67.4	0.51 %	285.6	0.56 %	726.6	0.30 %	231.0	0.51 %
late_r1500d30	312.0	**0.52 %**	18.8	0.30 %	208.0	0.49 %	520.0	0.36 %	156.0	0.45 %
late_r2000d50	414.0	**0.28 %**	17.0	0.20 %	230.0	0.25 %	460.0	0.26 %	138.0	0.21 %
late_r2000d65	1220.0	0.28 %	60.4	0.28 %	610.0	**0.31 %**	1159.0	0.26 %	427.0	0.24 %
hidden_r500d10	162.0	0.54 %	45.0	0.43 %	79.2	**0.59 %**	229.2	0.56 %	52.8	0.52 %
hidden_r500d15	957.0	0.68 %	233.2	0.72 %	497.2	**0.91 %**	1115.4	0.58 %	367.4	0.84 %
hidden_r1000d20	227.2	**0.63 %**	18.8	0.44 %	115.2	0.56 %	300.8	0.56 %	83.2	0.53 %
hidden_r1000d25	361.2	**1.12 %**	38.2	0.78 %	210.0	1.11 %	537.6	0.45 %	168.0	0.99 %
hidden_r1500d30	187.2	**0.56 %**	2.0	0.06 %	104.0	0.53 %	306.8	0.48 %	78.0	0.48 %
hidden_r1500d40	1360.8	0.20 %	146.8	0.19 %	770.4	0.20 %	1483.2	0.20 %	576.0	**0.21 %**
hidden_r2000d50	487.6	0.23 %	37.8	0.22 %	322.0	**0.26 %**	644.0	0.12 %	230.0	0.25 %
hidden_r2000d65	671.0	0.22 %	59.8	0.24 %	451.4	**0.26 %**	768.6	0.19 %	366.0	0.25 %
hidden_r2500d70	541.2	0.22 %	39.6	**0.27 %**	330.0	0.22 %	541.2	0.22 %	250.8	0.20 %
hidden_r2500d75	937.2	0.26 %	39.0	**0.27 %**	497.0	0.24 %	937.2	0.22 %	355.0	0.23 %
Promedio	584.76	0.45 %	58.84	0.35 %	326.80	**0.46 %**	675.52	0.35 %	240.69	0.42 %

Fig. B.1: Comparación de iteraciones y mejora de las variaciones del modelo de reubicación, removiendo las variaciones de a una, y utilizando el *Corte Media*

Bibliografía

[1] Norbert Ascheuer, Matteo Fischetti, and Martin Grötschel. Solving the asymmetric travelling salesman problem with time windows by branch-and-cut. *Mathematical Programming*, 90(3):475–506, 2001.

[2] Vasek Chvátal, editor. *Linear Programming*. W. H. Freeman, 1983.

[3] George Bernard Dantzig and John Ramser. The truck dispatching problem. *Management science*, 1959.

[4] Conjunto de Instancias "Hidden" del VeRoLog solver challenge 2017. `https://verolog.ortec.com/wp-content/uploads/2017/07/ORTECinstancesHidden.zip`.

[5] Conjunto de Instancias "Late" del VeRoLog solver challenge 2017. `https://verolog.ortec.com/wp-content/uploads/2017/04/ORTECLateInstances.zip`.

[6] Mejores Soluciones de las Instancias "Hidden" del VeRoLog solver challenge 2017. `https://verolog.ortec.com/wp-content/uploads/2017/07/BestSolsORTECinstancesHidden.zip`.

[7] Samuel Eilon, Nicos Christofides, and Carl Donald Tyndale. Watson-Gandy. *Distribution management: mathematical modelling and practical analysis*. 1971.

[8] Roberto De Franceschi, Matteo Fischetti, and Paolo Toth. A new ILP-based refinement heuristic for vehicle routing problems. *Mathematical Programming*, 105(2-3):471–499, November 2006.

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

[10] Agustín Montero, Juan José Miranda-Bront, and Isabel Méndez-Díaz. An ILP-based local search procedure for the VRP with pickups and deliveries. *Annals of Operations Research*, 259(1-2):327–350, 2017.

[11] Agustín Montero, Isabel Méndez-Díaz, and Juan José Miranda-Bront. An integer programming approach for the time-dependent traveling salesman problem with time windows. *Computers & Operations Research*, 88:280 – 289, 2017.

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

[13] VeRoLog solver challenge 2017. `https://verolog.ortec.com/`.

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

[15] Paolo Toth and Daniele Vigo, editors. *The vehicle routing problem.* Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2001.

[16] Paolo Toth and Daniele Vigo, editors. *Vehicle Routing: Problem, Methods and Applications.* Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2014.