Universidad de Buenos Aires
Facultad de Ciencias Exactas y Naturales
Departamento de Computación

Algoritmos basados en programación lineal entera para el problema de ruteo de buses escolares

Tesis de Licenciatura en Ciencias de la Computación

Martín Mongi Badía

Director: Dr. Javier Marenco

Buenos Aires, 2019

ALGORITMOS BASADOS EN PROGRAMACIÓN LINEAL ENTERA PARA EL PROBLEMA DE RUTEO DE BUSES ESCOLARES

En esta tesis, estudiamos maneras de resolver el problema de ruteo de buses escolares. Este problema consiste en, dado un sector urbano, un conjunto de potenciales paradas, estudiantes y sus ubicaciones, buscar un conjunto de rutas y asignación de paradas óptimos. Aunque se pueden usar varias definiciones de optimalidad, nosotros buscamos minimizar la distancia recorrida por los buses. Nuestro enfoque es en la resolución exacta de este problema.

Presentamos tres modelos diferentes de programación lineal entera básicos. Generamos variantes sobre los modelos particulares, con el fin de mejorar la performance. Con el objetivo de reducir la simetría de nuestros modelos, presentamos una técnica de agrupación de estudiantes. Creamos una heurística para utilizar una solución inicial y una heurística primal sobre subnodos. Realizamos una extensa experimentación y presentamos los resultados obtenidos y las conclusiones al respecto.

Palabras claves: Problema de ruteo de buses escolares, problema de ruteo de vehículos, programación lineal entera, sistemas de transporte.

ALGORITHMS BASED ON INTEGER LINEAR PROGRAMMING FOR THE SCHOOL BUS ROUTING PROBLEM

In the thesis, our focus lies on the school bus routing problem. Given an urban sector, a set of potential stops, students and their locations, the problem consists in finding an optimal set of routes and stop selection for students. Although multiple optimality definitions can be used for this problem, we focus on minimizing the total traveled distance by the buses. We try to find an exact solution for this problem.

We present three different integer linear programming basic models. We generate variants over the particular models, aiming to improve performance. With the objective of reducing model symmetry, we present a student clustering technique. We develop an heuristic to provide an initial solution and a primal heuristic to run on subnodes. We carry out extensive experimentation and present the results and our conclusions.

Keywords: School bus routing problem, vehicle routing problem, integer linear programming, transportation systems

AGRADECIMIENTOS

En primer lugar, tengo que agradecer a mis padres Marcela Badía y Marcelo Mongi. Desde siempre pusieron muchísimo énfasis en la importancia de la educación y se hicieron cargo de que no me faltara absolutamente nada en ningún aspecto para llegar a esta instancia.

Luego, tengo que agradecer a mi hermana Daniela Mongi Badía. Sus aportes en forma de compañía durante los experimentos, consejos sobre el texto u otras fueron invalorables para esta tesis.

También tengo que agradecer a mi abuela Dora Ventoso. El apoyo que me ha dado en estos años fue importantísimo.

Además, agradezco a mis amigos, los de siempre: Alan Barbagiovanni, Paco Caminoa, Francisco Fels, Juanma Fernández, Facu Ferraro, Agos Gargiulo, Félix Irigoyen, Nico Martín, Nacho Montiliengo, Nico Piferrer, Franco Preci, Fer Rey, Fer Ruiz, Juanma Santoro, Agus Sarquis, Lea Vidal, Agus Zelona y algunos más de los que seguro me olvido en este momento que estuvieron en viajes, reuniones y momentos buenos y malos.

Tengo que agradecer también a Lau Pezzatti, quien fue mi entrenadora de olimpiadas durante casi todo mi secundario y me sugirió seguir esta carrera.

Agradezco también a los chicos de Willterns y GARG, que hicieron de las pasantías que hice con ellos experiencias inolvidables. Además, sus consejos relativos a la carrera y la industria me ayudaron muchísimo.

También debo agradecer a los docentes de FCEN por la dedicación que desinteresadamente le prestaron a sus materias.

Particularmente, debo agradecer al director de este trabajo, Javier Marenco. Sin su ayuda, buena predisposición y constante disponibilidad este trabajo habría estado muy lejos de ser posible.

Finalmente, agradezco a los jurados Paula Zabala y Federico Pousa, que aceptaron ser jurados de esta tesis.

A mi abuelo, Máximo Badía.

Índice general

1. INTRODUCCIÓN

El transporte de estudiantes desde y hacia las escuelas es de vital importancia para las sociedades modernas. Cuando éste es centralizado y ofrecido sin cargo a los estudiantes, puede ser una valiosa herramienta de inclusión educativa. Adicionalmente, es importante que dicho transporte sea provisto de forma eficiente en términos de costos y contaminación, conveniente y segura para los estudiantes. Por ejemplo, en Estados Unidos, 480 mil buses escolares transportan a 26 millones de estudiantes todos los días. A gran escala, la optimización de estos servicios puede acarrear ahorros muy significativos para el erario público.

En este capítulo introductorio, se presenta el problema de ruteo de los buses escolares, se hace un comentario sobre la literatura existente en el tema y se define el problema que trataremos en el resto del trabajo.

1.1. Ruteo de buses escolares

Desde el trabajo seminal de Newton y Thomas (1969) [1], el problema ha sido definido de diferentes formas. La interpretación más general es la siguiente: dadas una escuela, un conjunto de potenciales paradas, y un conjunto de estudiantes, una solución al problema es una asignación de cada estudiante a una parada, y un conjunto de rutas que sirvan a todas las paradas asignadas a uno o más estudiantes. En los diferentes trabajos sobre el tema, se utilizaron diferentes definiciones de optimalidad; algunos ejemplos son: la cantidad de buses utilizados, la distancia total recorrida por los buses, la distancia total recorrida por los estudiantes, la distancia caminada por los estudiantes a las paradas, el balance de carga de buses, la distancia de ruta máxima y la pérdida de tiempo de estudiantes. Al mismo tiempo, también variaron las restricciones particulares utilizadas en cada trabajo. Algunos ejemplos son: la capacidad de los buses (homogénea o heterogénea), el tiempo o la distancia máxima que puede estar un estudiante en el bus, la distancia máxima que puede caminar un estudiante a la parada y la mínima cantidad de estudiantes para crear una ruta adicional.

Para este trabajo, tomamos las siguientes decisiones. Asumiremos que existe una única escuela a la que asisten todos los estudiantes. Asumiremos que la capacidad de los buses es homogénea; es decir, todos los buses de nuestra flota tienen la misma capacidad. Utilizaremos como parámetro a optimizar la distancia total recorrida por nuestra flota de buses. Estableceremos las restricciones de distancia máxima de caminata de estudiante a la parada. Asumiremos que los garages de cada bus están dentro de la ciudad y que todos los puntos de interés de nuestro modelo (paradas, escuela y garages) están en esquinas de la ciudad.

Este problema se puede descomponer varios subproblemas: la selección de paradas a utilizar, la asignación de estudiantes a paradas, y la generación de las rutas que las recorren. En este trabajo, utilizaremos modelos que toman todas estas decisiones conjuntamente en el proceso de optimización.

Luego, queremos que nuestros algoritmos, dados estos parámetros de entrada, devuelva una asignación de parada a cada estudiante y, para cada bus, una ruta que comience en su punto de partida, recorra un subconjunto de paradas, y finalice en la escuela. Todos los

estudiantes deben tener paradas asignadas y todas las paradas con estudiantes asignados deben ser visitadas.

Un ejemplo de una instancia del problema y su correspondiente resolución se pueden ver en las Figuras 1.1 y 1.2.

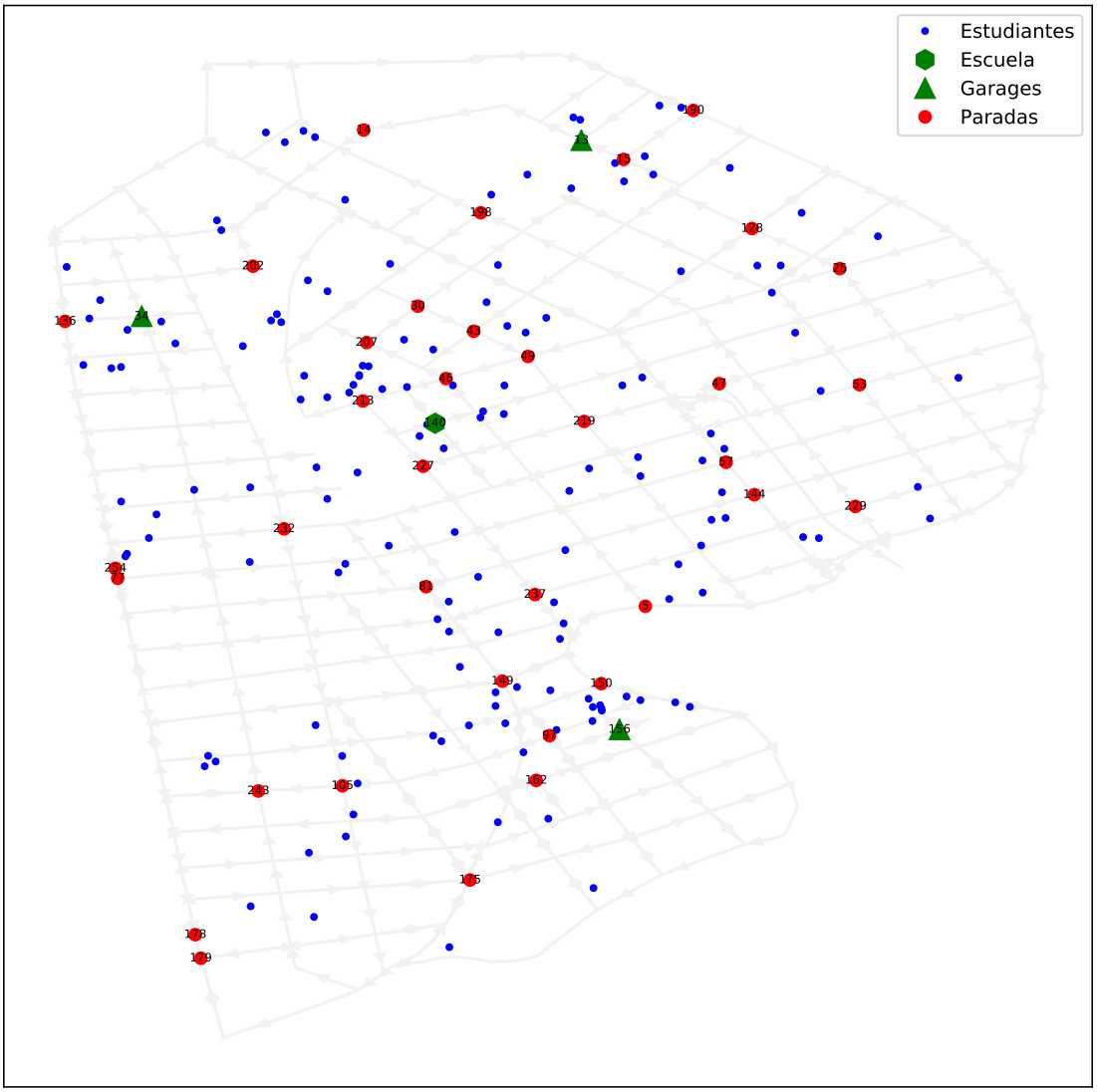

Fig. 1.1: Una instancia de prueba del barrio porteño de la Boca con 40 paradas, 150 estudiantes, 3 buses de capacidad 60 y una distancia máxima de caminata de 200 metros

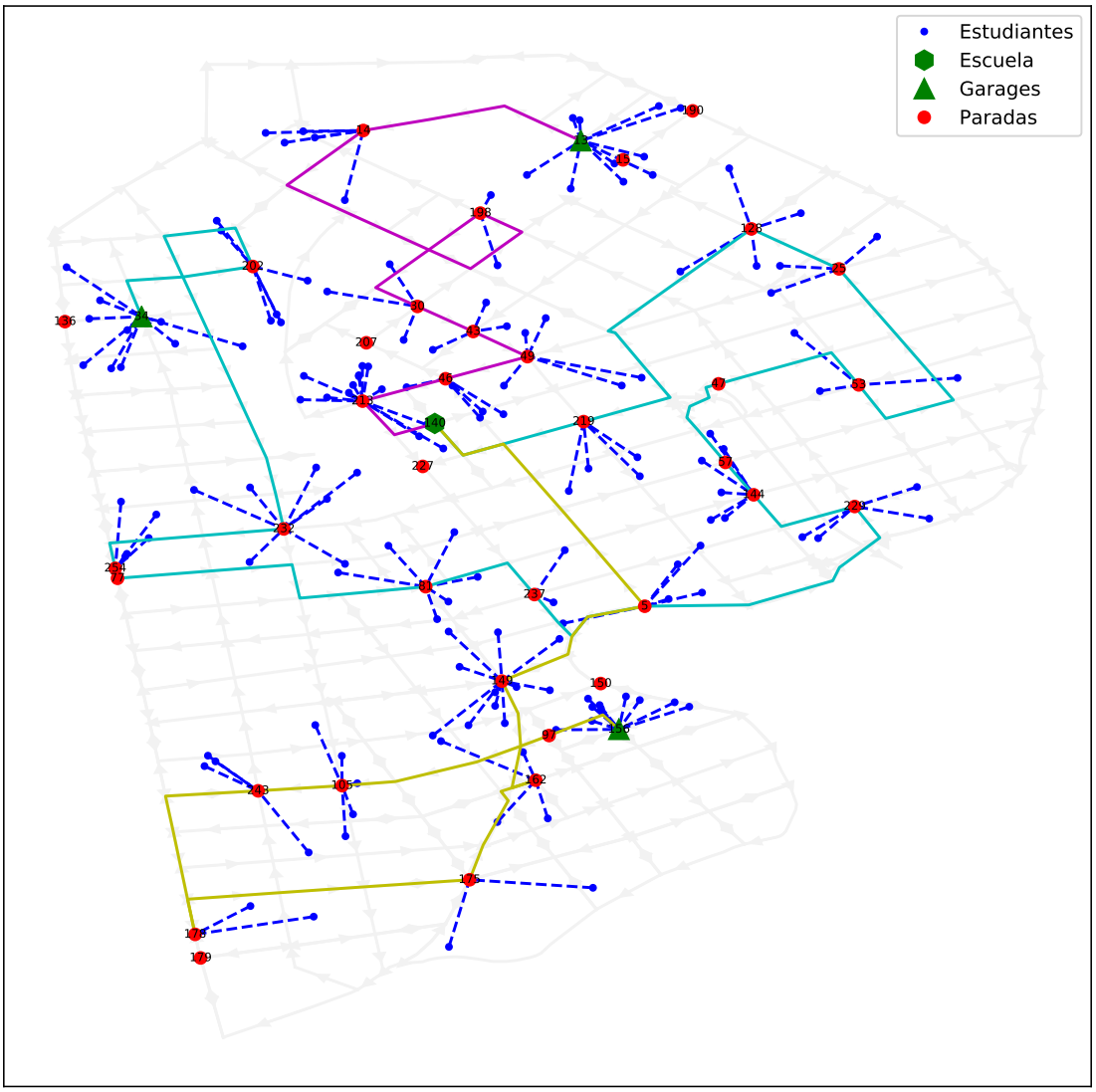

Fig. 1.2: La solución óptima al problema de la Figura 1.1

1.2. Literatura existente

En esta sección, haremos una breve reseña de la literatura existente sobre el problema de ruteo de buses escolares. Dos reseñas literarias que resumen la investigación sobre el problema son los trabajos de Park & Kim [2] en 2010 y Ellegood et al. [3] en 2019. El primer trabajo que encontramos sobre dicho problema es el de Newton & Thomas [1] en 1969. A partir de la fecha se vio modesto interés en el problema. Sin embargo, en la última década, el interés en el problema tuvo un incremento significativo, como se explica en Ellegood et al. [3]. Repasaremos algunos de los trabajos en los siguientes párrafos.

El primer trabajo del que tenemos registro es el de Newton & Thomas [1] en 1969. En este trabajo, los autores consideran una instancia de una escuela, un conjunto de paradas donde los estudiantes ya están asignados a una. Quieren asegurar que sean respetados el tiempo máximo de permanencia en el bus de los estudiantes y la capacidad de los buses a utilizar. Presentan un algoritmo que se compone de varias etapas. Primero, buscan una única ruta que sirva a todas las paradas sin importar las restricciones, utilizando una técnica golosa y posterior búsqueda local. Luego, separan esta ruta en varias etapas, con el fin de respetar las restricciones mencionadas. Finalmente, hacen algunos experimentos con instancias artificiales de hasta 80 paradas.

Durante los siguientes años hubo varios trabajos que trataron el problema, aún sin modelar la selección de paradas de bus. Es decir, todas las paradas debían estar en las rutas y la cantidad de alumnos asignados a cada una era un parámetro de entrada. Algunos ejemplos de esta época son de Angel et al. [4] en 1972, de Newton & Thomas [5] en 1974 y Bennett & Gazis [6] en 1972. Una mención especial merece el trabajo de Gavish & Shlifer [7] en 1979 que presenta una formulación matemática del problema, y resuelve a optimalidad una instancia de 21 paradas utilizando un modelo de programación lineal no entera y la técnica de *branch and bound*.

El primer trabajo que considera la posibilidad de seleccionar las paradas del que tenemos registro es de Bodin & Berman [8] en 1979. En este trabajo, los estudiantes son asignados a la potencial parada más cercana a su domicilio y luego son generadas las rutas. Este concepto es llamado *location-allocation-routing* (LAR) en la literatura. Otro trabajo que utliza esta técnica es de Dulac et al. [9] en 1980. En oposición, la técnica llamada *allocation-routing-location* (ARL) se basa en separar los estudiantes en grupos, armar una ruta para cada grupo y luego asignar los estudiantes a las paradas de su grupo. Un ejemplo de uso es el trabajo de Chapleau et al. [10] en 1985. Otro trabajo que utiliza la técnica es el de Bowerman et al. [11] en 1995, en el cual los autores presentan una formulación matemática de programación no lineal entera, aunque utilizan heurísticas para resolver el problema.

Más recientemente, encontramos en el trabajo de Schittekat et al. [12] en 2006 el primer modelo de programación lineal entera que resuelve un problema similar al nuestro. En este trabajo, los autores toman un problema de una única escuela, donde la asignación de paradas no está dada y los buses comienzan su recorrido en la misma escuela. El modelo presentado es similar a los usados para resolver el problema de ruteo de vehículos capacitado, con la salvedad de que en este caso no tienen que ser todas las paradas visitadas. Utilizan un solver comercial para experimentos en instancias pequeñas (10 paradas y 50 estudiantes). Este mismo modelo es utilizado en el trabajo de Schitekkat et al. [13] en 2013 para obtener soluciones óptimas y comparar con la metaheurística que presentan. Dicha metaheurística sirve como base para la heurística de solución inicial presentada en esta

tesis.

En la última década, la cantidad de investigación sobre el problema ha aumentado considerablemente. Sin embargo, la gran mayoría de los trabajos están enfocados en resolver el problema de forma no exacta, con heurísticas clásicas o metaheurísticas (ver Ellegood et al. [3]). Continuaremos nuestra reseña comentando solo los trabajos que tratan soluciones exactas para el problema.

En el trabajo de Martínez & Viegas [14] en 2011, podemos ver una resolución exacta del problema de forma secuencial. Asignan en primer lugar una parada a cada estudiante y luego, dada esa asignación, resuelven el problema de ruteo. Los autores presentan dos modelos de programación entera para el problema. Es importante notar que la solución del ruteo puede no ser óptima en general, dadas las decisiones tomadas previamente en la resolución de la asignación de paradas.

Un trabajo interesante es el de Riera-Ledesma & Salazar-González [15] en 2012. En este trabajo, los autores proponen un modelo de programación lineal entera basado en flujo de estudiantes, y presentan un algoritmo de *branch & cut* para resolverlo. Luego, los mismos autores extienden su trabajo en Riera-Ledesma & Salazar-González [16] en 2013 proponiendo una solución basada en generación de columnas.

Otro trabajo que utiliza la técnica de generación de columnas y *branch & price* es el de Kinable et al. [17] en 2014. Los autores además comparan resultados computacionales con el trabajo de Schittekat et al. [12] y muestran una mejora.

1.3. Definición del problema

Para el tratamiento de nuestro problema, vamos a modelarlo de la siguiente forma:

- Un grafo ponderado dirigido $G = \langle V, E \rangle$, en el cual existe un vértice v en V por cada esquina de nuestra región a estudiar y existe un arista (v_1, v_2) en E si y sólo si hay un segmento de calle que une las esquinas correspondientes a v_1 y v_2 y es posible recorrer la calle con un bus en ese sentido.
- Un conjunto $P \subseteq V$ que representa a las potenciales paradas que podremos utilizar en nuestras rutas. Dentro de P tendremos un nodo distinguido v_E que representa a la escuela. También tendremos un subconjunto $P_G \subseteq P$ que representa los potenciales puntos de partida de los buses.
- Un conjunto S que representa a los estudiantes.
- Una capacidad $C \in \mathbb{N}$ que representa la capacidad de cada bus.
- Una función $paradas : S \longrightarrow \mathcal{P}(P \setminus \{v_E\})$ que representa, para cada estudiante, las paradas a las cuales se lo puede asignar.
- Una función de costo $w : E \longrightarrow \mathbb{R}_{\geq 0}$ que representa el costo para un bus de ir entre dos esquinas.

Cada solución que consideraremos válida es un conjunto R de caminos en G, tal que todas las rutas comiencen en garages distintos, terminen en la escuela y una asignación $a : S \longrightarrow P \setminus \{v_E\}$ de estudiantes a paradas tal que cada parada en la imagen de la función sea recorrida por una ruta. Además, deben cumplirse las restricciones de capacidad. Nuestro enfoque será en minimizar la distancia recorrida por los buses.

Dado que nuestro grafo representa un plano urbano, es seguro asumir que $|E| \in \mathcal{O}(|V|)$

1.4. Complejidad computacional

El problema de ruteo de buses escolares, como lo definimos en la sección anterior, es un problema de optimización; es decir, hay una magnitud que debemos minimizar o maximizar. Sin embargo, las clases de complejidad computacional de problemas están definidas sobre los problemas de decisión. Entonces, definimos el problema de decisión de ruteo de buses escolares como el siguiente. Dados una instancia X del problema original de ruteo de buses escolares y $k \in \mathbb{R}$, el problema de decisión tiene respuesta positiva en el caso de que exista una solución del problema de optimización sobre la instancia X cuya función objetivo tome un valor menor o igual a k. Queremos demostrar que el problema de decisión de ruteo de buses escolares pertenece a la clase de complejidad computacional NP-completo. Para esto, demostraremos que el problema pertenece a las clases NP y NP-difícil.

NP La clase de complejidad computacional NP está definida como el conjunto de problemas de decisión para los cuales, cuando la solución es positiva, y dicha solución es provista, la verificación de la solución se puede realizar en tiempo polinomial. En el caso del problema de decisión de ruteo de buses escolares, dada una instancia del problema y su solución, simplemente hay que chequear que se cumplan las siguientes restricciones:

- Todos los estudiantes deben tener una parada asignada dentro de la distancia máxima de caminata.
- Todas las paradas con estudiantes asignados deben ser recorridas por alguna ruta
- La capacidad de los buses debe ser respetada
- La distancia recorrida por los buses es menor o igual a k

Todos estos chequeos se pueden realizar en tiempo polinomial. Por lo tanto, el problema de decisión de ruteo de buses escolares pertenece a NP.

NP-difícil Un problema de decisión X está en la clase de complejidad computacional NP-difícil si para todo problema $Y \in NP$, existe una reducción factible en tiempo polinomial de Y a X.

Equivalentemente, un problema de decisión X está en NP-difícil si existe un problema de decisión Y en NP-completo tal que existe una reducción factible en tiempo polinomial de Y a X. Probaremos que el problema en cuestión está en NP-difícil, tomando a Y como la versión de decisión del problema de viajante de comercio.

El problema del viajante de comercio toma un grafo completo $G = (V, V \times V)$, una función de longitud de arcos $w : V \times V \longrightarrow \mathbb{R}$ y un largo L. Tiene respuesta positiva si existe un ciclo que pase por todos los vértices del grafo una vez y tenga longitud menor o igual a L. Dados estos datos y tomando $V = \{v_0, v_1, \ldots v_{n-1}\}$, definimos nuestro conjunto de paradas $V' = V \cup \{v_n\}$, nuestro grafo $G' = (V', V' \times V')$ y nuestra función de longitud de arcos $w' : V' \times V' \longrightarrow \mathbb{R}$ definida de la siguiente forma:

$$w'(u,v) = \begin{cases} w(u,v) & \text{si } u \in V \wedge v \in V \\ w(u,v_0) & \text{si } u \in V \wedge v \notin V \\ w(v_0,v) & \text{si } u \notin V \wedge v \in V \\ 0 & \text{si no} \end{cases} \tag{1.1}$$

Es simple ver que generamos un vértice *copia* de v_0. Definimos nuestro conjunto de estudiantes $S' = \{s_0, s_1 \dots s_{n-1}\}$ y nuestra función de posibles paradas $paradas' : S' \longrightarrow \mathcal{P}(V')$ de tal forma que $(\forall s_i \in S)\ \ paradas'(s_i) = \{v_i\}$. Tomamos v_0 como único depósito y v_n como la escuela. Es simple ver que todas las definiciones que hicimos se pueden hacer en tiempo polinomial.

Entonces, el problema de decisión de ruteo de buses escolares tendrá respuesta positiva si existe una ruta de largo menor o igual a L que recoja a todos los estudiantes. Como toda parada en V tiene un estudiante del cual es la única opción, la ruta debe contener a todas estas paradas y terminar en v_n. Es simple ver que esta misma ruta representa a un ciclo equivalente en el problema original del viajante de comercio. Entonces, si existe una ruta de largo menor o igual a L que puede recoger a todos los estudiantes recorriendo todas las paradas, también existe un ciclo de largo menor o igual a L que recorre todos los vértices del grafo.

Hemos demostrado entonces que el problema de decisión del viajante de comercio es reducible al problema de decisión de ruteo de buses escolares. Por lo tanto, el problema de decisión de ruteo de buses escolares pertenece a NP-difícil.

NP-completo Como el problema de decisión de ruteo de buses escolares pertenece a NP y NP-difícil, pertenece a NP-completo.

1.5. Contenido de la tesis

El objetivo de esta tesis es presentar varias formas de resolver el problema de ruteo de buses escolares, experimentar sobre ellas y descubrir casos de uso de algunas de ellas.

En el Capítulo 2, presentaremos tres modelos de programación lineal entera. Sobre dos de estos modelos, presentaremos variantes que esperamos mejoren en algunos casos la performance de nuestro modelo. Explicaremos su motivación y algunas hipótesis sobre su performance en diferentes instancias del problema.

En el Capítulo 3, el enfoque estará puesto en mejorar la performance de los modelos presentados en el capítulo anterior. Presentaremos una optimización en particular que se puede hacer sobre uno de los modelos. Luego, presentamos una técnica de agrupación de estudiantes, a través de la cual reduciremos la simetría de nuestro modelo, con esperanzas de mejorar la performance. Al mismo tiempo, presentaremos dos heurísticas: la primera será una típica heurística golosa, para contar con una solución inicial y la segunda utilizará los resultados de la relajación lineal en cada nodo para mejorar la solución incumbente.

En el Capítulo 4, intentaremos comprobar la validez de algunas de nuestras hipótesis. Primero, experimentaremos aisladamente con las optimizaciones presentadas en el Capítulo 4, para verificar si proveen una mejora significativa en el tiempo de cómputo de nuestros problemas. Luego, haremos una comparación de las mejores versiones de cada modelo, variando los parámetros de entrada para evaluar las fortalezas de cada uno de los modelos.

En el Capítulo 5, presentaremos nuestras conclusiones y posibles direcciones de investigación futura.

En el Apéndice, haremos una breve reseña del marco teórico en el cual se basa esta tesis.

2. MODELADO CON PROGRAMACIÓN LINEAL ENTERA

En este capítulo plantearemos las diferentes formas encontradas de modelar el problema. En general, van incrementando en sofisticación, a medida que fuimos teniendo más entendimiento del problema, y disminuyendo en cantidad de variables.

2.1. Modelo con pre-cálculo de distancia

Este modelo es una implementación adaptada del presentado en el trabajo de Schittekat et al [12] en 2006. En este trabajo toman un modelo particular, en el cual todos los buses parten de la escuela. Nuestro modelo es el mismo adaptado a que los buses puedan partir de paradas arbitrarias. Para este modelo, es preciso pre-calcular la distancia mínima entre todos los pares de paradas en $P \times P$. Para esto, utilizamos el algoritmo de Dijkstra [18] desde cada una de las paradas. De esta forma, utilizando una representación del grafo como lista de adyacencia y una cola de prioridad podemos calcular estas distancias en $\mathcal{O}(|P||E|log|V|)$. Representaremos el costo de ir de una parada a otra con la función $w' : P \times P \to \mathbb{R}_{\geq 0}$

Para el modelo con pre-cálculo de distancia, utilizaremos las siguientes variables de decisión:

- Para cada $v_0 \in P_G$, una variable de decisión binaria $r_{v_0} \in \{0, 1\}$, de modo tal que $r_{v_0} = 1$ si y solo si se utiliza el bus que sale de v_0.
- Para cada $v_0 \in P_G$, y cada $p \in P \setminus \{v_E\}$, una variable de decisión $rp_{v_0,p} \in \{0, 1\}$, de modo tal que $rp_{v_0,p} = 1$ si y sólo si el bus que comienza su recorrido en v_0 tiene una parada en la esquina p.
- Para cada $v_0 \in P_G$, y cada $(u, v) \in P \times P$ tal que $u \neq v$, una variable de decisión $re_{v_0,(u,v)} \in \{0, 1\}$, de modo tal que $re_{v_0,(u,v)} = 1$ si y sólo si el bus que comienza su recorrido en v_0 va de la parada u a la parada v sin paradas intermedias.
- Para cada $v_0 \in P_G$, $s \in S$, y cada $p \in paradas(s)$, una variable de decisión $rsp_{v_0,s,p} \in \{0, 1\}$, de modo tal que $rsp_{v_0,s,p} = 1$ si y sólo si el estudiante s se toma el bus en la parada p, cubierta por el bus que comienza en v_0.

Para facilitar la escritura de las restricciones, definimos los grados de entrada y salida de los vértices de la siguiente forma:

$$g^+_{v_0}(v) = \sum_{\substack{u \in P \\ v \neq u}} re_{v_0,(v,u)} \qquad\qquad g^-_{v_0}(v) = \sum_{\substack{u \in P \\ v \neq u}} re_{v_0,(u,v)}$$

Entonces, nuestro modelo queda dado por:

$$\text{mín} \sum_{\substack{v_0 \in P_G \\ v \in P \\ u \in P \\ v \neq u}} w'(v, u) \times re_{v_0,(v,u)} \tag{2.1}$$

$$g^+_{v_0}(p) = g^-_{v_0}(p) \qquad \forall v_0 \in P_G, p \in P \setminus \{v_E, v_0\} \tag{2.2}$$

$$g^+_{v_0}(v_0) - g^-_{v_0}(v_0) = r_{v_0} \qquad \forall v_0 \in P_G \tag{2.3}$$

$$g^+_{v_0}(v_E) - g^-_{v_0}(v_E) = -r_{v_0} \qquad \forall v_0 \in P_G \tag{2.4}$$

$$g^+_{v_0}(p) = rp_{v_0,p} \qquad \forall v_0 \in P_G, p \in P \setminus \{v_E\} \tag{2.5}$$

$$\sum_{v_0 \in P_G} rp_{v_0,p} \leq 1 \qquad \forall p \in P \tag{2.6}$$

$$rp_{v_0,p} \leq r_{v_0} \qquad \forall v_0 \in P_G, p \in P \tag{2.7}$$

$$\sum_{\substack{v_0 \in P_G \\ p \in paradas(s)}} rsp_{v_0,s,p} = 1 \qquad \forall s \in S \tag{2.8}$$

$$rsp_{v_0,s,p} \leq rp_{v_0,p} \qquad \forall s \in S, v_0 \in P_G, p \in paradas(s) \tag{2.9}$$

$$\sum_{\substack{s \in S \\ p \in paradas(s)}} rsp_{v_0,s,p} \leq C \qquad \forall v_0 \in P_G \tag{2.10}$$

$$\sum_{\substack{v \in W \\ u \in W \\ v \neq u}} re_{v_0,v,u} \leq |W| - 1 \qquad \forall v_0 \in P_G, W \subset P \setminus \{v_E, v_0\} \tag{2.11}$$

Podemos ver que la función (2.1) minimiza nuestra distancia total entre todas las rutas. Luego, las restricciones (2.2) aseguran que para todas las rutas, los vértices que no sean garage o parada, tienen grado de entrada igual que el grado de salida. Por otro lado, las restricciones de (2.3) aseguran que la diferencia entre el grado de salida y de entrada sea 0 o 1, ya que está restringido por las cotas de r_{v_0} (a su vez, asegura que esta esté en 1 si la ruta está activa y en 0 si no). Análogamente, con las restricciones (2.4) se restringe la diferencia de grados de la escuela como -1 o 0. Con las restricciones (2.5), se impide que una parada se active si el vértice no tiene grado de salida. Luego, las restricciones (2.6) impiden que un vértice sea parada de 2 o más rutas. Con las restricciones (2.7) se impide que una parada se asigne a una ruta no activa. En (2.8), se fuerza que a cada estudiante se le asigne solamente una ruta y parada. Además, con las restricciones (2.9) se requiere que un estudiante se asigne a una parada y una ruta, solo si esa parada está activa en esa ruta. Las restricciones (2.10) fuerzan que las capacidades de los buses se respeten. Por último, las restricciones (2.11) impiden que se formen *subtours*, impidiendo que subgrafos de n vértices que contengan n o más ejes.

Es importante notar que la familia de restricciones (2.11) contiene una cantidad de restricciones de orden exponencial sobre la cantidad de paradas. Esto hace que la implementación de todas sea impráctico ya en modelos medianos. Por este motivo, en nuestra implementación estas restricciones se implementan como *lazy constraints*. Es decir, sólo se agregan a nuestro modelo si, encontrada una solución entera, se encuentra violada una de estas restricciones. Esta búsqueda se realiza algorítmicamente, recibiendo la solución entera encontrada como parámetro. Si es el caso, se agrega al modelo y se continúa con el cómputo.

Este modelo se plantea con $|P_G| \times (|P|^2 - 1 + \sum_{s \in S} |paradas(s)|)$ variables binarias y $|P_G| \times (2|P| + \sum_{s \in S} |paradas(s)| + 1) + |P| + |S|$ restricciones iniciales.

2.1.1. Variante con restricciones de Miller-Tucker-Zemlin

Como mencionamos, la familia de restricciones (2.11) crece exponencialmente con la cantidad de paradas de nuestro modelo. Esto hace que su inclusión en modelos medianos o grandes sea poco práctica. Si bien se pueden ir agregando las restricciones necesarias a medida que se efectúa el cómputo, esto puede llegar a generar un modelo muy grande hacia el final del proceso. Por lo tanto, reemplazaremos estas restricciones (2.11) por la familia introducida en Miller, Tucker, Zemlin [19]. Para esto, necesitaremos una nueva familia de variables $rr_{v_0,p} \in \mathbb{R}_{\geq 0}$ para cada $v_0 \in P_G$ y cada $p \in P$. Estas variables representarán el rango de la parada. Es decir, la cantidad de paradas que la ruta ya ha recorrido hasta llegar a esta parada inclusive. Agregaremos las siguientes restricciones:

$$rr_{v_0,v_0} = 1 \qquad \forall v_0 \in P_G \tag{2.12}$$

$$rr_{v_0,v} - rr_{v_0,u} - |P| re_{v_0,(u,v)} \geq 1 - |P| \quad \forall v_0 \in P_G, v \in P \setminus \{v_0\}, u \in P \setminus \{v_E\} \tag{2.13}$$

Las restricciones (2.12) aseguran que los garages tengan rango 1 en su propia ruta. Las restricciones (2.13) garantizan que si el bus que comienza su recorrido en v_0 recorre las paradas u y v consecutivamente, el rango se mantiene. Su funcionamiento es el siguiente:

- Si $re_{v_0,(u,v)} = 0$, la restricción se cumple trivialmente, ya que existen los suficientes rangos para que las paradas tomen y cumplan esa restricción.
- Si $re_{v_0,(u,v)} = 1$, entonces se debe cumplir $rr_{v_0,v} - rr_{v_0,u} \geq 1$. Esto implica que el rango de v supere en 1 o más al rango de u.

Con estas restricciones, podemos restringir los *subtours* y forzamos a que nuestra ruta quede conexa. De esta forma, el modelo queda con $|P_G| \times (|P|^2 + |P| - 1 + \sum_{s \in S} |paradas(s)|)$ variables binarias y $|P_G| \times (|P|^2 + \sum_{s \in S} |paradas(s)| + 3) + |P| + |S|$ restricciones fijas.

2.2. Modelo directo

Como vimos, el modelo de la Sección 2.1 utiliza una variable de decisión para cada par de paradas y ruta, representando la conexión inmediata de esas paradas en esa ruta. Esto hace que esa familia de variables sea de tamaño $|P_G||P|^2$, es decir, crece cuadráticamente con la cantidad de paradas. Dado que, en general, la cantidad de variables repercute perjudicialmente en la performance de nuestros modelos, es de nuestro interés usar la menor cantidad de variables posible. Con ese fin, presentamos este modelo directo que reemplaza esa familia de variables binarias, con una familia de variables enteras que representan la cantidad de veces que el bus recorre cada arista (cuadra) de nuestro grafo (ciudad). Nuestro modelo utiliza las siguientes variables de decisión:

- Para cada $v_0 \in P_G$, una variable de decisión binaria $r_{v_0} \in \{0, 1\}$, de modo tal que $r_{v_0} = 1$ si y solo si se utiliza el bus que sale de v_0.
- Para cada $v_0 \in P_G$, y cada $p \in P \setminus \{v_E\}$, una variable de decisión $rp_{v_0,p} \in \{0, 1\}$, de modo tal que $rp_{v_0,p} = 1$ si y sólo si el bus que comienza su recorrido en v_0 tiene una parada en la esquina p.

- Para cada $v_0 \in P_G$, y cada $e \in E$, una variable de decisión $re_{v_0,e} \in \mathbb{N}_0$, de modo tal que $re_{v_0,e} = k$ si y sólo si el bus que comienza su recorrido en v_0 recorre la cuadra representada por e k veces.
- Para cada $v_0 \in P_G$, $s \in S$, y cada $p \in paradas(s)$, una variable de decisión $rsp_{v_0,s,p} \in \{0,1\}$, de modo tal que $rsp_{v_0,s,p} = 1$ si y sólo si el estudiante s se toma el bus en la parada p, cubierta por el bus que comienza en v_0.

Para facilitar la lectura del modelo, definimos los grados de entrada y salida de los vértices:

$$g^+_{v_0}(v) = \sum_{\substack{u \in V \\ (v,u) \in E}} re_{v_0,(v,u)} \qquad g^-_{v_0}(v) = \sum_{\substack{u \in V \\ (u,v) \in E}} re_{v_0,(u,v)}$$

Luego, nuestro modelo queda dado por:

$$\text{mín} \sum_{\substack{v_0 \in P_G \\ e \in E}} w(e) \times re_{v_0,e} \tag{2.14}$$

$$g^+_{v_0}(v) = g^-_{v_0}(v) \qquad \forall v_0 \in P_G, v \in V \setminus \{v_E, v_0\} \tag{2.15}$$

$$g^+_{v_0}(v_0) - g^-_{v_0}(v_0) = r_{v_0} \qquad \forall v_0 \in P_G \tag{2.16}$$

$$g^+_{v_0}(v_E) - g^-_{v_0}(v_E) = -r_{v_0} \qquad \forall v_0 \in P_G \tag{2.17}$$

$$g^+_{v_0}(v) \geq rp_{v_0,v} \qquad \forall v_0 \in P_G, p \in P \setminus \{v_E\} \tag{2.18}$$

$$\sum_{v_0 \in P_G} rp_{v_0,p} \leq 1 \qquad \forall p \in P \tag{2.19}$$

$$rp_{v_0,p} \leq r_{v_0} \qquad \forall v_0 \in P_G, p \in P \tag{2.20}$$

$$\sum_{\substack{v_0 \in P_G \\ p \in paradas(s)}} rsp_{v_0,s,p} = 1 \qquad \forall s \in S \tag{2.21}$$

$$rsp_{v_0,s,p} \leq rp_{v_0,p} \qquad \forall s \in S, v_0 \in P_G, p \in paradas(s) \tag{2.22}$$

$$\sum_{\substack{s \in S \\ p \in paradas(s)}} rsp_{v_0,s,p} \leq C \qquad \forall v_0 \in P_G \tag{2.23}$$

$$\sum_{\substack{v \in W \\ u \in V \setminus W}} re_{v_0,v,u} \geq rp_{v_0,p} \qquad \forall v_0 \in P_G, W \subset V \setminus (P_G \cup \{v_E\}), p \in W \cap P \tag{2.24}$$

El modelo es bastante similar al modelo detallado con pre-cálculo de distancias, con una excepción que debemos explicitar. Previamente, impedíamos los *subtours* evitando ciclos en nuestro grafo. En este caso, no podremos abordar el problema de esa forma, ya que necesitamos permitir ciclos para que nuestro recorrido pueda usar varias veces el mismo eje, como en el caso de una vuelta manzana. Entonces, las restricciones (2.24) requieren que, para cualquier asignación de una parada a una ruta, todos los subconjuntos de los vértices que no contengan a la escuela y al garage de esa ruta tengan un vértice que salga de ese subconjunto. Esta familia de restricciones, de la misma forma que en el

modelo con pre-cálculo, tiene tamaño de orden exponencial en la cantidad de paradas. Por lo tanto, se agregan en forma de *lazy constraints*

Este modelo tiene $|P_G| \times (|P| + |E| + |S| \times (|P| - 1))$ variables y $|P_G| \times (|V| + |P| \times (|S| + 2) + 1) + |S|$ restricciones fijas. La cantidad de *lazy constraints* que tiene este modelo está acotada superiormente por $|P_G| \times |P| \times 2^{|V|-|P_G|-1}$.

2.2.1. Variante de conectividad por variables de flujo

El modelo desarrollado anteriormente asegura la conectividad de la ruta con una familia de restricciones de tamaño de orden exponencial en la cantidad de vértices. De forma similar a lo explicado para el modelo con pre-cálculo de distancias, esto no es deseable, ya que no permite utilizar múltiples hilos de procesamiento manteniendo el determinismo, tiene un *overhead* por la necesidad de programar una llamada en Python, y puede suceder que agregue muchas restricciones que hagan pesado el modelo. Con esto en mente, intentaremos reemplazar esta familia de restricciones con familias de restricciones y variables de orden polinomial en la cantidad de paradas. Para cada $v_0 \in P_G$, y cada $e \in E$, agregaremos una variable $ref_{v_0,e} \in \mathbb{N}_0$. Esta variable va a ser usada como una forma de garantizar la conectividad de nuestro camino; es decir, eliminar los subtours desconexos. La idea se basa en crear una red de flujo subyacente al camino generado por las variables $re_{v_0,e}$. Para mejor entendimiento, usaremos la siguiente notación para flujo de salida y flujo de entrada de un vértice de nuestro modelo:

$$f^+_{v_0}(v) = \sum_{\substack{u \in V \\ (v,u) \in E}} ref_{v_0,(v,u)} \qquad f^-_{v_0}(v) = \sum_{\substack{u \in V \\ (u,v) \in E}} ref_{v_0,(u,v)}$$

En cada garage de bus crearemos una fuente de flujo de cantidad igual a la cantidad de paradas atendidas por esa ruta. En cada parada que esté activa, crearemos un sumidero de flujo de cardinalidad unitaria. Con esta estrategia, la única forma de que un camino sea solución es que toda parada atendida por la ruta tenga un camino desde el garage, justamente lo que necesitabamos. Luego, para esta variante, eliminamos las restricciones (2.24), y las reemplazamos con las siguientes restricciones:

$$ref_{v_0,e} \leq |P| \times re_{v_0,e} \qquad \forall e \in E, v_0 \in P_G \qquad (2.25)$$

$$f^+_{v_0}(v) = f^-_{v_0}(v) \qquad \forall v \in V \setminus P, v_0 \in P_G \qquad (2.26)$$

$$f^+_{v_0}(p) = f^-_{v_0}(p) + rp_{v_0,p} \qquad \forall v_0 \in P_G, p \in P \setminus \{v_E, v_0\} \qquad (2.27)$$

$$f^+_{v_0}(v_0) + r_{v_0} = f^-_{v_0}(v_0) + \sum_{p \in P \setminus \{v_E\}} rp_{v_0,p} \qquad \forall v_0 \in P_G \qquad (2.28)$$

Entonces, las restricciones (2.25) aseguran que un arco puede tener flujo positivo si y sólo si el bus pasa por ese arco. Luego, las restricciones (2.26) aseguran que el flujo se conserve en las esquinas de nuestro modelo que no sean potenciales paradas. Las restricciones (2.27) aseguran que, si la parada pertenece a esa ruta, el flujo de salida de la parada es menor al de entrada en una unidad. Veamos como funcionan las restricciones (2.28): si la ruta no está activa, el flujo de salida es igual al flujo de entrada, ya que $r_{v_0} = \sum_{p \in P \setminus \{v_E\}} rp_{v_0,p} = 0$; si la ruta está activa, entonces la diferencia entre entrada y salida de flujo es igual a la cantidad de paradas de la ruta, menos uno (por el garage del

bus que sirve como parada). De esta forma, se garantiza que todas las soluciones enteras de este problema sean caminos conexos hacia la escuela. Con las familias presentadas, se agregan $|E|$ variables y $|P_G| \times (|E| + |V| - 1)$ restricciones.

2.3. Modelo plano

Este modelo se trata de una optimización por sobre el modelo con pre-cálculo de distancias. Un modelo de estilo similar es visto en Riera-Ledesma y Salazar-González [15]. En nuestros modelos anteriores, se multiplican las variables y restricciones por cada una de las rutas que los buses deberían cubrir. Si tuviéramos muchos buses, esto puede hacer que la cantidad de variables y restricciones torne impráctico el cómputo. Luego, para el modelo plano, condensamos muchas de estas variables. Este enfoque complica el seguimiento de las cargas de los buses y su consecuente restricción por capacidad. Por lo tanto, introducimos el concepto de carga de parada. Dichas variables representan la cantidad de alumnos que tiene el bus cuando deja cada parada. En más detalle, utilizaremos las siguientes variables:

- Para cada $v_0 \in P_G$, una variable de decisión binaria $r_{v_0} \in \{0, 1\}$, de modo tal que $r_{v_0} = 1$ si y solo si se utiliza el bus que sale de v_0.
- Para cada $p \in P \setminus \{v_E\}$, una variable de decisión $p_p \in \{0, 1\}$, de modo tal que $p_p = 1$ si y sólo si existe un bus que tiene una parada en la esquina p.
- Para cada $(u, v) \in P \times P$ tal que $u \neq v$, una variable de decisión $e_{(u,v)} \in \{0, 1\}$, de modo tal que $e_{(u,v)} = 1$ si y sólo si existe un bus que va de la parada u a la parada v sin paradas intermedias.
- Para cada $s \in S$ y cada $p \in \mathit{paradas}(s)$, una variable de decisión $sp_{s,p} \in \{0, 1\}$, de modo tal que $sp_{s,p} = 1$ si y sólo si el estudiante s se toma algún bus en la parada p.
- Para cada $p \in P$, una variable de decisión $cp_p \in \mathbb{N}_0$, de modo tal que cp_p represente la cantidad de alumnos que hay en el bus cuando deja la parada p.

Para facilitar la escritura de las restricciones, redefinimos los grados de entrada y salida de los vértices de la siguiente forma:

$$g^+(v) = \sum_{\substack{u \in P \\ v \neq u}} e_{(v,u)} \qquad\qquad g^-(v) = \sum_{\substack{u \in P \\ v \neq u}} e_{(u,v)}$$

Luego, este modelo queda dado por:

$$\text{mín} \sum_{\substack{v \in P_G \\ u \in P_G \\ v \neq u}} w'(v, u) \times e_{(v,u)} \tag{2.29}$$

$$g^+(v) = g^-(v) \qquad \forall v \in P \setminus (P_G \cup \{v_E\}) \quad (2.30)$$

$$g^+(v_0) - g^-(v_0) = r_{v_0} \qquad \forall v_0 \in P_G \quad (2.31)$$

$$g^+(v_E) - g^-(v_E) = -\sum_{v_0 \in P_G} r_{v_0} \quad (2.32)$$

$$g^+(v) = p_v \qquad \forall p \in P \setminus \{v_E\} \quad (2.33)$$

$$\sum_{p \in paradas(s)} sp_{s,p} = 1 \qquad \forall s \in S \quad (2.34)$$

$$sp_{s,p} \leq p_p \qquad \forall s \in S, p \in paradas(s) \quad (2.35)$$

$$\sum_{\substack{s \in S \\ p \in paradas(s)}} sp_{s,p} \leq cp_p \qquad \forall p \in P \quad (2.36)$$

$$\sum_{\substack{s \in S \\ p \in paradas(s)}} sp_{s,p} + cp_u + C \times (e_{(u,v)} - 1) \leq cp_v \qquad \forall v, u \in P, v \neq u \quad (2.37)$$

$$cp_p \leq C \qquad \forall p \in P \quad (2.38)$$

El modelo sigue una lógica similar al de la Sección 2.1. La función objetivo (2.29) es análoga a la del modelo con pre-cálculo. Las restricciones (2.30) aseguran que las paradas tengan igual grado de salida que de entrada. Al mismo tiempo, las restricciones (2.31) y (2.32) definen las diferencias de grados de los garages y escuela, respectivamente. Es importante ver que están restringidas ya que r_i es una variable binaria. Con las restricciones (2.33) impedimos que se active una parada que no está en una ruta. Las restricciones de asignación de parada a estudiante (2.34) y (2.35) son análogas a los modelos anteriores. Por otro lado, las restricciones (2.36) aseguran que la carga del bus saliendo de cada parada sea como mínimo la cantidad de estudiantes asignados a esta parada. Las restricciones (2.37) funcionan de la siguiente forma: si $e_{(u,v)} = 1$, entonces asegura que la carga de la parada v sea mayor o igual a la carga de la parada u más los estudiantes propios de v. Si $e_{(u,v)} = 0$, el lado izquierdo de la inecuación es tan bajo que es cumplida trivialmente. Las restricciones (2.38) aseguran que se cumplan las capacidades estipuladas.

Este modelo tiene $|P_G| + |P| \times (|P| + |S| + 1)$ variables y $|P| \times (|S| + |P| + 3) + |S| - 1$ restricciones.

3. MEJORAS A LOS MODELOS DE PLE

En el capítulo anterior, planteamos varios modelos que pueden representar nuestro problema a resolver. Si bien con sus características alcanzan para computar una solución, hay mejoras que se les pueden agregar, con el fin de disminuir el tiempo de cómputo y, si no es posible encontrar el óptimo, mejorar las soluciones que las implementaciones llegan a encontrar.

3.1. Corte de flujo máximo sobre el modelo directo

Como planteamos el modelo directo previamente, nuestro problema queda bien definido. Sin embargo, podemos plantear algunas restricciones adicionales que no modifican la solución óptima de nuestro problema, pero mejoran la relajación lineal de nuestro modelo, mejoran la cota dual y por consiguiente la performance. En esta sección plantearemos una familia de esas restricciones.

En nuestro modelo, la variable $re_{v_0,e} \forall v_0 \in P_G, e \in E$ representa la cantidad de veces que la ruta que comienza en v_0 pasa por el arco e. Es fácil ver que si el arco no es parte del camino mínimo entre algún par de paradas, el arco no será utilizado por ningún bus, i.e. $\sum_{v_0 \in P_G} re_{v_0,e} = 0$. Definiremos entonces a $\delta(u, v)$ como el camino mínimo del vértice u al vértice v. Entonces, utilizaremos la siguiente familia de restricciones:

$$\sum_{v_0 \in P_G} re_{v_0,e} = 0 \qquad \forall e \in E, (\nexists\ p_1, p_2 \in P : e \in \delta(p_1, p_2)) \tag{3.1}$$

Al mismo tiempo, veamos si podemos generalizar la noción de las restricciones planteadas. Sabemos que todos los pesos de los arcos son positivos. Por lo tanto, cualquier camino mínimo entre dos vértices de nuestro grafo será libre de ciclos. Entonces, si un arco está en el camino mínimo entre solamente un par de vértices, el arco será usado a lo sumo una vez, i.e., $\sum_{v_0 \in P_G} re_{v_0,e} \leq 1$. Generalizando, podemos ver que el valor de la variable está acotado superiormente por la cantidad de pares de paradas que lo tienen en su camino mínimo. Entonces, definimos las siguientes restricciones:

$$\sum_{v_0 \in P_G} re_{v_0,e} \leq |\{(p_1, p_2) \in P \times P : e \in \delta(p_1, p_2)\}| \qquad \forall e \in E \tag{3.2}$$

Sin embargo, podemos ajustar más está cota; por las restricciones (2.19), sabemos que cada parada puede ser visitada por un bus sólo. Por lo tanto, si el arco pertenece a dos caminos mínimos con mismo punto de partida, i.e. $e \in \delta(p_1, p_2) \cap \delta(p_1, p_3)$, sólo podrá tomar uno de los dos. Veamos si podemos generalizar esta idea. Hacemos las siguientes definiciones:

$$\mathcal{F}(e) = \{(p_1, p_2) \in P \times P : e \in \delta(p_1, p_2)\} \tag{3.3}$$
$$\mathcal{F}^+(e) = \{p_1 : (p_1, p_2) \in \mathcal{F}(e)\} \tag{3.4}$$
$$\mathcal{F}^-(e) = \{p_2 : (p_1, p_2) \in \mathcal{F}(e)\} \tag{3.5}$$

Luego, podemos acotar por la máxima cantidad de pares de paradas que pueden pasar al mismo tiempo por el arco. Esto es, no hay dos caminos que compartan puntos de partida o puntos de llegada. Es importante notar que la escuela es un caso especial; no se admiten caminos de salida, pero se admiten a lo sumo $|P_G|$ caminos de entrada. Definimos la función $\Psi : \mathcal{P}(P \times P) \to \mathbb{B}$ que evalúa si un subconjunto de $P \times P$ cumple con la condición:

$$\Psi(X) = (\forall\ (p_1, p_2) \in X : p_1 \neq v_E) \wedge \tag{3.6}$$
$$(\forall\ (p_1, p_2), (p_3, p_4) \in X : p_1 = p_3 \Rightarrow p_2 = p_4) \wedge \tag{3.7}$$
$$(\forall\ (p_1, p_2), (p_3, p_4) \in X : p_2 = p_4 \Rightarrow (p_1 = p_3 \vee p_2 = v_E) \wedge \tag{3.8}$$
$$|\{(p_1, p_2) \in X : p_2 = v_E\}| \leq |P_G| \tag{3.9}$$

En definitiva, $\Psi(X)$ evalúa que los caminos mínimos entre los pares de vértices de X puedan suceder al mismo tiempo. Esto es: no hay caminos salientes de la escuela, no hay dos caminos con puntos de partida iguales, no hay caminos con puntos de llegada iguales, salvo por la escuela que tiene a lo sumo $|P_G|$. Entonces, podemos plantear la siguiente restricción:

$$\sum_{v_0 \in P_G} re_{v_0,e} \leq \max_{\substack{X \subset \mathcal{F}(e) \\ \Psi(X)}} |X| \qquad \forall e \in E \tag{3.10}$$

Si la restricción definida fuera implementada de forma directa, es decir, iterando sobre los subconjuntos de $\mathcal{F}(e)$ y buscando el más grande validado por Ψ, sería muy costoso computacionalmente ya que existen $2^{|\mathcal{F}(e)|}$ subconjuntos. Este problema es muy similar a un problema muy estudiado que es el máximo apareamiento de aristas en un grafo bipartito. En este problema, se nos da un grafo bipartito $G = \langle V_1 \cup V_2, E \rangle$ en el cual se nos pide encontrar el máximo subconjunto de aristas independiente (no tienen vértices en común). Una forma eficiente de resolver este problema es creando una red de flujo y computando el flujo máximo. Para resolver nuestro problema, utilizamos un enfoque similar. Definimos nuestro grafo:

$$N_{\mathcal{F}(e)} = \langle \{s, t\} \cup \{p^+ : p \in \mathcal{F}^+(e)\} \cup \{p^- : p \in \mathcal{F}^-(e)\}, f_{\mathcal{F}(e)} \rangle \tag{3.11}$$

Es decir, los vértices de nuestra red de flujo son una fuente, un sumidero, un vértice por cada parada de salida en $\mathcal{F}(e)$, uno por cada uno de entrada. Si un vértice se encuentra en ambos conjuntos tiene un vértice de salida y uno de entrada (p^+ y p^-, respectivamente). Luego, definimos nuestra función de capacidad:

$$f_{\mathcal{F}(e)}(s, p^+) = 1 \qquad \forall p \in \mathcal{F}^+(e) \tag{3.12}$$
$$f_{\mathcal{F}(e)}(p^+, q^-) = 1 \qquad \forall (p, q) \in \mathcal{F}(e) \tag{3.13}$$
$$f_{\mathcal{F}(e)}(q^-, t) = 1 \qquad \forall q \in \mathcal{F}^-(e) \setminus \{v_E\} \tag{3.14}$$
$$f_{\mathcal{F}(e)}(q^-, t) = |V_G| \qquad \forall q \in \mathcal{F}^-(e) \cap \{v_E\} \tag{3.15}$$
$$f_{\mathcal{F}(e)}(p, q) = 0 \qquad \text{en cualquier otro caso} \tag{3.16}$$

De esta forma, podemos utilizar el algoritmo de Dinics en $\mathcal{O}(V^2 E)$ y tener nuestra cota en un tiempo razonable.

3.2. Clústers de estudiantes

En todos los modelos descritos en el capítulo anterior, existía un conjunto de variables de decisión binarias para cada estudiante, a través de las cuales el modelo asignaba una parada (y ruta en algunos casos) a cada estudiante. Sin embargo, es fácil ver que dados dos estudiantes s y t tal que $paradas(s) = paradas(t)$, es indiferente intercambiar las asignaciones de los estudiantes entre sí. Entonces, podemos tener conjuntos de estudiantes con conjuntos iguales de posibles paradas, asignarlos indiferentemente con variables enteras. Llamaremos clústers a estos conjuntos de estudiantes. Esto nos permitiría hacer una optimización a nuestros modelos. Por ejemplo, en el modelo con pre-cálculo y el directo tenemos la variable de decisión:

$$rsp_{v_0,s,p} \in \{0,1\} \qquad \forall v_0 \in P_G, s \in S, p \in paradas(s) \tag{3.17}$$

Se cumple que $rsp_{v_0,s,p} = 1$ si y sólo si el estudiante s se toma el bus en la parada p, cubierta por el bus que comienza en v_0. Es posible intercambiar esta variable por

$$rcp_{v_0,c,p} \in [0, |\{s \in S : paradas(s) = c\}|] \cap \mathbb{Z} \quad \forall v_0 \in P_G, c \in Im(paradas), p \in c \tag{3.18}$$

De este modo $rcp_{v_0,c,p}$ toma el valor de la cantidad de estudiantes s tal que $paradas(s) = c$ que toman el bus de la ruta que parte de v_0 en la parada p. Es importante notar que, dado que $|Im(paradas)| \leq |S|$, la cantidad de variables disminuye o se mantiene constante.

Al mismo tiempo, debemos modificar las restricciones (2.8), (2.9) y (2.10), ya que utilizan las variables que removimos del modelo. Son reemplazadas por las siguientes:

$$\sum_{\substack{v_0 \in P_G \\ p \in c}} rcp_{v_0,c,p} = |\{s \in S : paradas(s) = c\}| \qquad \forall c \in Im(paradas) \tag{3.19}$$

$$rcp_{v_0,c,p} \leq |\{s \in S : paradas(s) = c\}| \times rp_{v_0,p} \quad \forall v_0 \in P_G, c \in Im(paradas), p \in c \tag{3.20}$$

$$\sum_{\substack{c \in Im(paradas) \\ p \in c}} rcp_{v_0,c,p} \leq C \qquad \forall v_0 \in P_G \tag{3.21}$$

De esta forma, las restricciones (3.19) fuerzan que la cantidad de estudiantes asignados como parte de nuestro clúster sea efectivamente el tamaño del clúster. Las restricciones (3.20) impiden que se asignen estudiantes a una parada y ruta, si alguna de estás dos no se encuentra activa. Por último, las restricciones (3.21) aseguran que se respeten las capacidades de los buses.

El cambio de variables para el modelo plano es análogo, removiendo la diferenciación por ruta.

3.2.1. Asignación de estudiantes individuales

Como fue explicado anteriormente, se pueden condensar las decisiones de asignación de estudiantes a paradas, siempre y cuando el conjunto de estudiantes tenga el mismo

conjunto de paradas posibles. Sin embargo, esto no explicita asignaciones particulares de estudiantes a paradas. Si bien es trivial, dado el resultado de un modelo agrupado, asignar estudiantes que cumplan con la interpretación, se puede iterar sobre esta idea y asignar los estudiantes a las rutas y paradas de tal forma que la distancia total caminada por los estudiantes sea mínima. Tomamos $\mathcal{R}$ como el conjunto de rutas resultado de la optimización de cualquiera de los modelos agrupados explicados anteriormente y $d(s,p)$ la distancia del estudiante s a la parada p. En esta sub-sección, formularemos un modelo que solucione este problema. Nuestro modelo tendrá un sólo tipo de variables de decisión:

- Para cada $s \in S$ y cada $p \in paradas(s)$ tal que $(\exists R \in \mathcal{R})\ p \in R$, tendremos $sp_{s,p} \in \{0,1\}$, de tal modo que $sp_{s,p} = 1$ si y sólo si el estudiante s es asignado la parada p.

Luego, este modelo queda dado por:

$$\text{mín} \sum_{\substack{s \in S \\ p \in paradas(S) \\ (\exists R \in \mathcal{R})\ p \in R}} d(s,p) \times sp_{s,p} \tag{3.22}$$

$$\sum_{\substack{p \in paradas(s) \\ (\exists R \in \mathcal{R})\ p \in R}} sp_{s,p} = 1 \qquad \forall s \in S \tag{3.23}$$

$$\sum_{\substack{s \in S \\ p \in R\ \cap\ paradas(s)}} sp_{s,p} \leq C \qquad \forall R \in \mathcal{R} \tag{3.24}$$

La función objetivo (3.22) minimiza la suma de distancias caminadas por los estudiantes. Las restricciones (3.23) aseguran que cada estudiante tenga exactamente una parada asignada y las restricciones (3.24) aseguran que se respeten las capacidades de las rutas. Este problema tiene la estructura de un problema de asignación. Por lo tanto, se pueden relajar las condiciones de integralidad, y el óptimo será nuestra asignación

3.3. Heurística para solución inicial

Cuando se utiliza una estrategia de *branch and bound* para resolver un programa de programación lineal entera, se efectúa, a grandes rasgos, la siguiente estrategia:

- Se resuelve el problema principal, relajando las restricciones de integralidad. Esto se puede resolver en un tiempo razonable utilizando el algoritmo símplex y se evalúa el resultado X^*:
  - Si el resultado no tiene variables fraccionarias, el resultado es la solución entera óptima.
  - Si el resultado tiene por lo menos una variable fraccionaria, se elige una x_i^* y se resuelven dos subproblemas: uno con la restricción adicional $x_i \leq \lfloor x_i^* \rfloor$ y el otro con la restricción $x_i \geq \lceil x_i^* \rceil$. Luego, se resuelven recursivamente estos dos problemas y se devuelve la mejor solución.

Esta estrategia mencionada anteriormente tiene el problema de que puede ser muy costosa computacionalmente. Para abordar este problema, cuando se encuentra una solución entera en alguno de los subproblemas, cualquier subproblema cuya relajación lineal tenga óptimo peor que la solución entera es descartado. De esta forma, se puede podar el árbol de recursión y ahorrar valioso tiempo de cómputo. Creamos una heurística con el fin de tener una solución razonable que nos permitiera poder descartar subproblemas. En esta sección, explicaremos su funcionamiento.

3.3.1. Construcción inicial

Para esto nos basamos en una heurística para el problema del viajante de comercio presentada en Clarke & Wright [20]. La diferencia es que en esta heurística, el viaje comienza y termina en el mismo vértice. En C&W, la solución inicial consiste en n caminos de ida y vuelta a cada uno de los n vértices, y estos se van concatenando según el ahorro de costos posible. Nosotros adaptamos esta estrategia; empezamos con un camino de cada depósito y vamos agregando cada una de las paradas conforme a los ahorros. Es importante aclarar que, cada vez que hacemos un cambio en nuestra solución, necesitamos chequear que existe una distribución de estudiantes entre las rutas que podría funcionar con la solución por la cual pretendemos reemplazarlas (es decir, que las capacidades de los buses es respetada). Los conjuntos de rutas que cumplen con esta condición los llamamos *factibles*. La forma de comprobar la factibilidad de distribuir los estudiantes la veremos más tarde. Presentamos el algoritmo de construcción de la solución inicial:

```
1: procedure ConstruirSoluciónInicial(P, P_G, v_E)
2:     R ← {[v_0, v_E] : v_0 ∈ P_G}                ▷ Rutas iniciales solo del depósito a la escuela
3:     U ← P \ (P_G ∪ {v_E})
4:     if ¬ Factible(R, ∅) then
5:         return
6:     end if
7:     while |U| > 0 do
8:         (∀ v ∈ U, r ∈ R, i ∈ [0 ... |r| − 2])
9:                 sv(v, r, i) ← dist(r[i], v) + dist(v, r[i + 1]) − dist(r[i], r[i + 1])
10:        while |sv| < 0 do
11:            v, r, i ← argmín_{(v,r,i)∈Dom(sv)} sv(v, r, i)
12:            R_n ← (R ∪ {r[0 ... i] + [v] + r[i ... |r − 1|]}) \ r
13:            if Factible(R_n, ∅) then
14:                R ← R_n
15:                U ← U \ {v}
16:                break
17:            end if
18:            sv(v, r, i) ← indefinido                ▷ Eliminamos esa opción
19:        end while
20:    end while
21:    return R
22: end procedure
```

Repasemos el funcionamiento del algoritmo. En la línea 2 creamos nuestro conjunto de rutas, con una ruta desde cada depósito hacia la escuela, sin paradas intermedias. Luego, en la línea 3, iniciamos el conjunto de paradas todavía no asignadas a una ruta. Si uno de los depósitos tuviera una cantidad de estudiantes asignados indefectiblemente a él que fuera mayor a la capacidad del bus, esta instancia no tendría solución; en las líneas 3 y 4 realizamos este chequeo, y abortamos la ejecución si es el caso. El ciclo que comienza en la línea 7 es donde asignamos las paradas a las rutas, este corre mientras el conjunto de paradas no ruteadas sea no vacío. En las líneas 8 y 9 creamos nuestra función de costos. La función evalúa, para cada parada no ruteada, para cada ruta y para cada posición en esa ruta, cuando aumentaría la distancia recorrida por nuestros buses. Como en nuestros grafos siempre se cumple la desigualdad triangular, estos costos son siempre positivos. Entonces, en el ciclo que comienza en la línea 10, recorreremos esa función en búsqueda de la operación que conlleve menor costo. En la líneas 11 y 12, buscamos la operación que tenga menor costo, y generamos nuestro nuevo conjunto de rutas. Luego, en la línea 13, chequeamos si esta operación puede llevar a un conjunto de rutas completo al que se le pudieran asignar los estudiantes y no se violan las capacidades de los buses. Si es el caso, se reemplaza el conjunto de rutas por el nuevo, y se quita la parada del conjunto de paradas por asignar. Si no lo es, eliminamos la operación del conjunto de operaciones posibles y evaluamos la operación siguiente.

Luego del paso constructivo, tenemos un conjunto de rutas que incluye a todas las paradas de nuestro modelo, y es factible. Un ejemplo del paso constructivo se puede ver en la Figura 3.1.

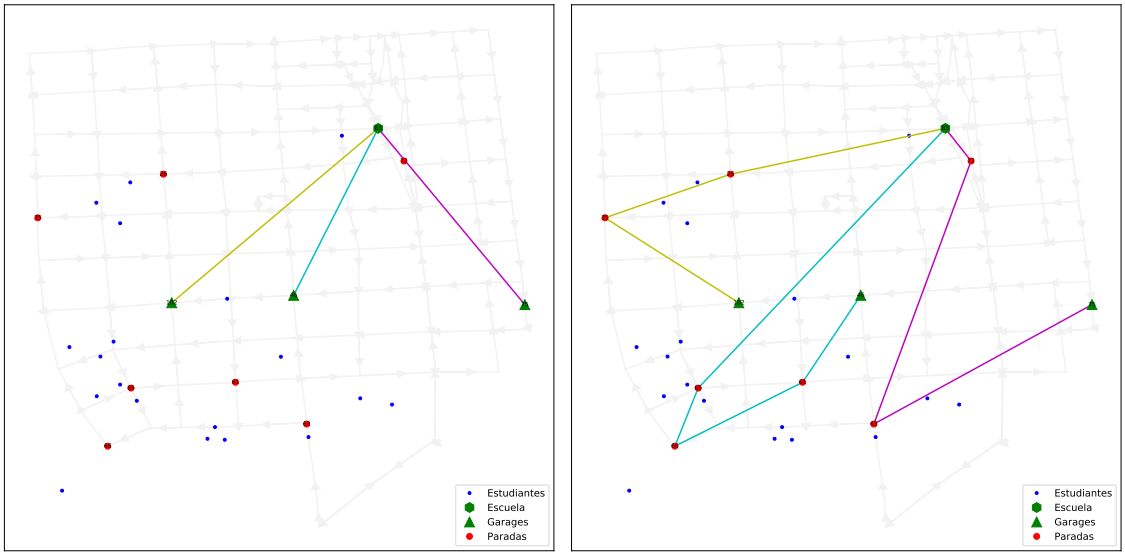

Fig. 3.1: Construcción de solución inicial de la heurística. Barrio de San Telmo, 10 paradas, 20 estudiantes

3.3.2. Búsqueda local

Como vimos en la sub-sección anterior, tenemos un conjunto de rutas que es parte de una solución completa de nuestro problema. Ahora, definiremos vecindarios que serán explorados en búsqueda de una mejor solución:

- Dado un conjunto de rutas $\mathcal{R}$ definimos el vecindario $Rem(\mathcal{R})$ como el conjunto de soluciones dado al quitar una parada de alguna de las rutas.
- Dado un conjunto de rutas $\mathcal{R}$ definimos el vecindario $RemInsDentro(\mathcal{R})$ como el conjunto de soluciones dado al quitar una parada de una de las rutas e insertarlo en la misma ruta en otro lugar.
- Dado un conjunto de rutas $\mathcal{R}$ definimos el vecindario $RemInsEntre(\mathcal{R})$ como el conjunto de soluciones dado al quitar una parada de una de las rutas e insertarlo en la otra ruta.
- Dado un conjunto de rutas $\mathcal{R}$ definimos el vecindario $Reempl(\mathcal{R})$ como el conjunto de soluciones dado al quitar una parada de una de las rutas e insertar una parada que no se encuentra en ninguna ruta.
- Dado un conjunto de rutas $\mathcal{R}$ definimos el vecindario $Concat(\mathcal{R})$ como el conjunto de soluciones dado al concatenar dos rutas.

Ejemplos sobre los vecindarios de una solución factible se pueden encontrar en la Figura 3.2. El proceso de la búsqueda local propuesta consiste en, basándose en una solución factible (rutas y paradas no utilizadas), pasar a una solución factible dentro de los vecindarios que cause el mayor ahorro. Cuando esto no sea posible porque no existe solución factible en el vecindario o todas tienen ahorro negativo, se da por finalizada la búsqueda local. El funcionamiento se puede ver en detalle en el siguiente algoritmo:

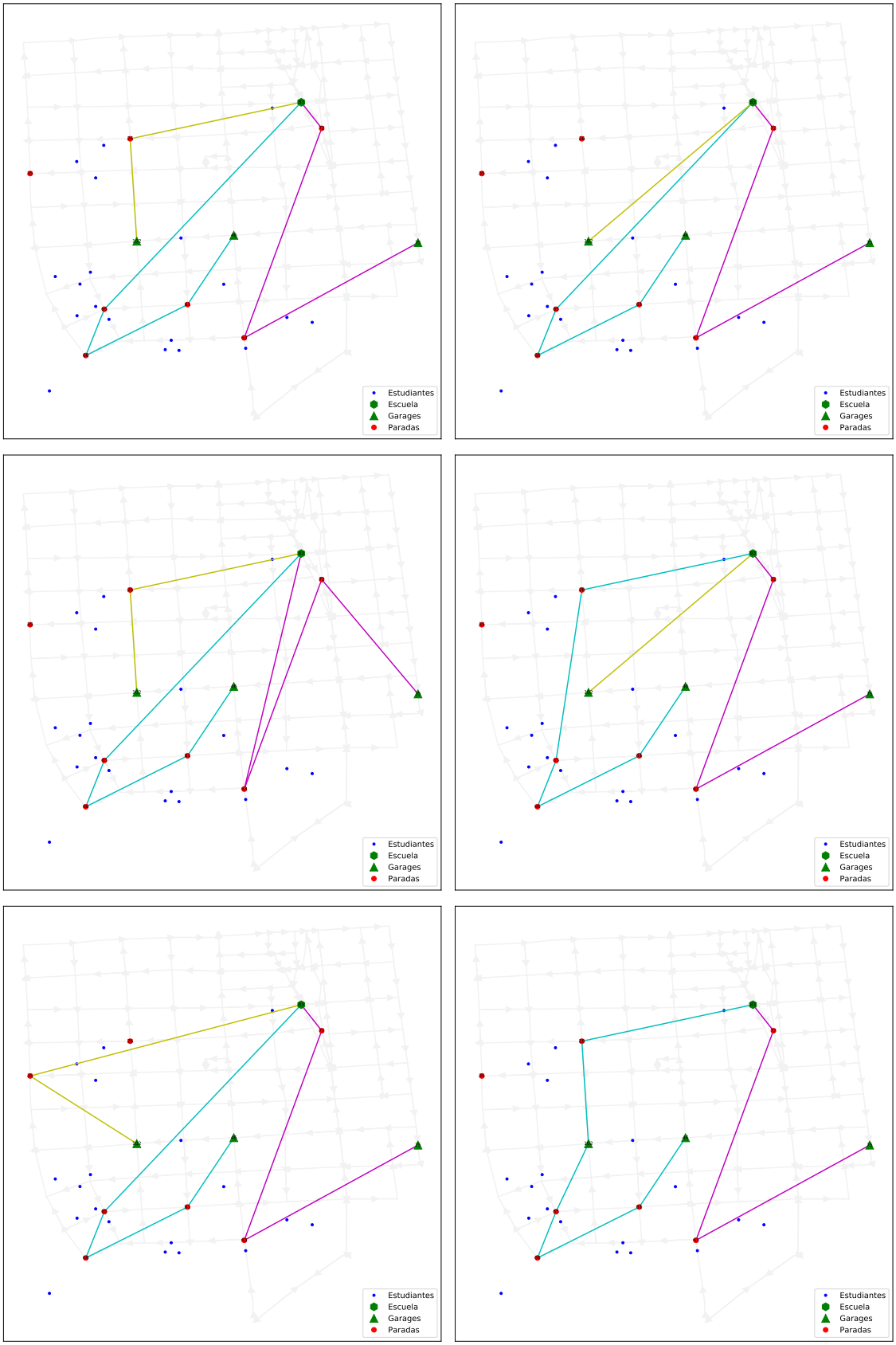

Fig. 3.2: Vecindarios de búsqueda local.
Arriba: Solución inicial y vecindad creada quitando una parada de una ruta.
Centro: Vecindades creadas quitando una parada de una ruta y insertando en la misma ruta a la izquierda y en otra ruta a la derecha.
Abajo: Vecindades creadas reemplazando una parada por una no utilizada a la izquierda y concatenando dos rutas a la derecha

```
1: procedure BúsquedaLocal(R)
2:     U ← ∅
3:     ch ← true
4:     while ch do
5:         (∀ R_n ∈ (Rem(R) ∪ RemInsDentro(R) ∪ RemInsEntre(R)∪
6:                                 Reempl(R) ∪ Concat(R)))
7:                 sv(R) ← costo(R_n) − costo(R_n)
8:         ch ← false
9:         while |sv| > 0 do
10:            R_n ← argmín_{R_n∈Dom(sv)} sv(R_n)
11:            if sv(R_n) ≥ 0 then
12:                break
13:            end if
14:            U_n ← U ∪ v(R) \ v(R_n)
15:            if Factible(R_n, U_n) then
16:                R ← R_n
17:                U ← U_n
18:                ch ← true
19:                break
20:            end if
21:            sv(R_n) ← indefinido
22:        end while
23:    end while
24:    return R
25: end procedure
```

Repasemos el funcionamiento del algoritmo. La función de búsqueda local toma el conjunto de rutas con todas las paradas de la heurística constructiva de la sección anterior. Entonces, el conjunto de paradas no enrutadas es vacío, como se define en la línea 2. Luego, en la línea 3, se le asigna verdadero a la variable centinela que utilizaremos para salir del ciclo principal cuando no se pueda mejorar una solución. En la línea 4 empieza dicho ciclo. En las líneas 5, 6 y 7, definimos nuestra función de ahorros (o costos), para cada una de las soluciones en el vecindario de nuestra solución actual. En la línea 8, asignamos falso a la variable centinela. Empezamos el ciclo que recorre la vecindad en la línea 9. En la línea 10, se busca el conjunto de rutas que signifique el mayor ahorro. En las líneas 11 y 12, chequeamos que el mayor ahorro sea positivo. Si este es no positivo, nuestra solución actual es la óptima del vecindario actual. Luego, chequeamos que, dadas las nuevas rutas, se puedan asignar los estudiantes a las paradas, respetando las restricciones de capacidad. Si este es el caso, nos quedamos con esta solución y repetimos el proceso. Si no lo es, eliminamos esta operación y buscamos con el mayor ahorro siguiente.

3.3.3. Subproblema de factibilidad

Como vimos en esta sección, en cada paso de una solución a otra, queremos chequear si, dadas las rutas (parciales o totales) de la nueva solución, es posible asignar las paradas no asignadas y los estudiantes, de forma que todas las restricciones sean cumplidas. En este problema, si bien se cuenta con bastantes características del problema general que queremos estudiar, podemos aprovechar la falta de una función objetivo y la consecuente

innecesidad de formular sobre arcos de ruta u orden de las paradas para usar una formulación reducida que nos permita saber lo que necesitamos. Nuestra forma de resolverlo es planteando un problema de programación lineal entera que toma como parámetro las rutas actuales $\mathcal{R}$, y las paradas que no van a ser utilizadas (porque fueron eliminadas en la búsqueda local) U. Estas deben ser diferenciadas de las que resta asignarles ruta ya que en el proceso de construcción es importante saber si a una parada no le vamos a poder asignar estudiantes o sí. El modelo se compone de las siguientes variables:

- Para cada $R \in \mathcal{R}$, y cada $p \in P \setminus \{v_E\}$, una variable de decisión $rp_{v_0,p} \in \{0,1\}$, de modo tal que $rp_{R,p} = 1$ si y sólo si la parada p está en la ruta R.
- Para cada $R \in \mathcal{R}$, cada $p \in P \setminus \{v_E\}$ y cada $c \in Im(paradas)$ tal que $p \in c$, una variable de decisión $rcp_{R,c,p} \in \mathbb{N}_0$ de modo tal que toma el valor de la cantidad de estudiantes s tal que $paradas(s) = c$ que toman el bus de la ruta R en la parada p.

Luego, este modelo se completa con las siguientes restricciones:

$$\sum_{r \in \mathcal{R}} rp_{R,p} \leq 1 \qquad \forall p \in P \setminus \{v_E\} \tag{3.25}$$

$$\sum_{\substack{R \in \mathcal{R} \\ p \in c}} rcp_{R,c,p} = |\{s \in S : paradas(s) = c\}| \qquad \forall c \in Im(paradas) \tag{3.26}$$

$$rcp_{R,c,p} \leq |\{s \in S : paradas(s) = c\}| \times rp_{R,p} \quad \forall R \in \mathcal{R}, c \in Im(paradas), p \in c \tag{3.27}$$

$$\sum_{\substack{c \in Im(paradas) \\ p \in c}} rcp_{R,c,p} \leq C \qquad \forall R \in \mathcal{R} \tag{3.28}$$

$$rp_{R,p} = 1 \qquad \forall R \in \mathcal{R}, p \in R \tag{3.29}$$

$$rp_{R,p} = 0 \qquad \forall p \in U \tag{3.30}$$

Las restricciones (3.25) aseguran que cada parada sea asignada a 1 ruta, como máximo. Luego, las restricciones (3.26) nos aseguramos de que todos los estudiantes del clúster estén asignados a una parada y una ruta. Las restricciones (3.27) aseguran que cuando se asigna un estudiante a una parada y ruta, la parada esté asignada a la ruta. Además, las restricciones (3.28) aseguran que se respeten las capacidades de los buses. Por último, las restricciones (3.29) y (3.30) aseguran que las variables tomen los valores adecuados para las paradas ya asignadas a una ruta y las que no serán asignadas, respectivamente.

3.4. Heurística de redondeo en nodos

Como explicamos anteriormente, utilizando la estrategia *branch and bound*, podemos utilizar la solución de nuestra relajación lineal para acercarnos a la solución entera de

nuestro problema. Esto implica que la eventual solución fraccionaria encontrada puede tener información sobre la solución óptima que podemos aprovechar. Para esto, desarrollamos una heurística basada en la solución lineal fraccionaria en cada nodo. A diferencia de la heurística anterior, que sirve para construir una solución inicial, como esta heurística corre en cada nodo, no queremos que sea muy costosa computacionalmente, ya que podría empeorar la performance del modelo, aún encontrando soluciones mejores que la incumbente. Utilizaremos este tipo de heurística sobre el modelo con pre-cálculo de distancia, y si obtenemos buenos resultados lo adaptaremos al resto. A continuación veremos el funcionamiento de nuestro algoritmo:

procedure HEURÍSTICALINEAL(r^*, rp^*, rsp^*, re^*)
  $(\forall v \in \overline{r} \cup \overline{rp} \cup \overline{rsp} \cup \overline{re})\ v \leftarrow 0$
  for $p \in P$ **do**
    $mv_0 \leftarrow \underset{v_0 \in P_G}{\text{argmáx}}\ rp^*_{v_0,p}$
    $(\forall v_0 \in |P_G| \setminus \{mv_0\})\ \ \overline{rp}_{v_0,p} \leftarrow 0$
    $\overline{rp}_{mv_0,p} \leftarrow \lceil rp^*_{mv_0,p} \rceil$
    $\overline{r}_{mv_0} \leftarrow \text{máx}(\overline{r}_{mv_0}, \lceil rp^*_{mv_0,p} \rceil)$
  end for
  for $s \in S$ **do**
    $mv_0, mp \leftarrow \underset{\substack{v_0 \in P_G, p \in P \setminus \{v_E\} \\ \overline{rp}_{v_0,p}=1}}{\text{argmáx}}\ rsp^*_{v_0,s,p}$
    $(\forall (v_0, p) \in (|P_G| \times (|P| \setminus \{v_E\})) \setminus \{(mv_0, mp)\})\ \ \overline{rsp}_{v_0,s,p} \leftarrow 0$
    $\overline{rsp}_{mv_0,s,mp} \leftarrow 1$
  end for
  $(\forall v_0 \in |P_G|, p_1, p_2 \in |P|, p_1 \neq p_2))\ \ c(v_0, p_1, p_2) \leftarrow (1 - re^*_{v_0,p_1,p_2}) \times w'(p_1, p_2)$
  for $v_0 \in |P_G|$ **do**
    if $\overline{r}_{mv_0} = 1$ **then**
      $R \leftarrow$ INSERCIÓN$(v_0, \{v | v \in |P|, \overline{rp}_{v_0,p} = 1\}, v_E, c)$ ▷ Inserción según c
      for $i \in [1 \dots |R| - 1]$ **do**
        $\overline{re}_{v_0,r_i,r_{i+1}} \leftarrow 1$
      end for
    end if
  end for
  return $\overline{r}, \overline{rp}, \overline{rsp}, \overline{re}$
end procedure

En síntesis, el algoritmo tiene tres pasos; primero, para cada parada, evalúa las variables de asignación a rutas y elige la que tenga mayor valor. Si este valor es cero, toma la parada como no activa. Luego, decide para cada estudiante, la parada y ruta a la cual asignar. Es importante comprobar que la combinación de ruta y parada estén el línea con lo asignado en el paso anterior. Por último, debemos programar el orden de las paradas de cada ruta. Creamos una función de costos sobre los arcos y ruta que nos dé alguna idea de que arcos conviene elegir. Esta función es el largo de la ruta multiplicado por el inverso de la variable de ese arco y ruta. De esta forma, si el valor de la variable es 1, el costo es 0. Si por el contrario es 0, el costo de esta ruta es el largo del arco. Basados en esta función, utilizamos un simple algoritmo de inserción de las paradas para cada ruta. Es importante destacar que es posible que utilizando la heurística descripta se llegue a una solución que no cumpla las restricciones de capacidad de nuestro problema. Esto en la implementación

es controlado simplemente manteniendo un contador de cantidad de estudiantes asignados a cada ruta. En el caso de que se esté utilizando el modelo con restricciones MTZ, se calculan los valores apropiados para que se cumplan.

4. EXPERIMENTACIÓN

4.1. Creación de casos de prueba

4.1.1. Conjunto de datos base

Para la experimentación, decidimos utilizar barrios reales de la Ciudad Autónoma de Buenos Aires. Decidimos tomar el conjunto de datos "Calles" [21] provisto por el Gobierno de la Ciudad Autónoma de Buenos Aires. Este conjunto de datos provee un archivo de valores separados por comas o csv (por *comma-separated values*), en la cual existe una entrada para cada segmento de calle entre dos esquinas (los que vendran a ser los arcos de nuestro grafo). A partir de esto, podemos obtener, para cada segmento, las coordenadas geográficas que la definen como un objeto `LINESTRING` del formato *Well-known text* [22], el nombre oficial de la calle, las alturas de numeración, la longitud del segmento, el sentido, el barrio (o barrios si es división) y varios atributos más que no nos son de utilidad para nuestro objetivo.

4.1.2. Generación del grafo G básico

Decidimos que nuestras instancias estarán basadas en calles de diferentes barrios porteños. Esta decisión nos permite tener instancias pequeñas como San Telmo de 1,23 km^2 y grandes como Palermo con 15,85 km^2. Basándonos en el conjunto de datos mencionado, podemos crear nuestro grafo de entrada de la siguiente forma:

1: **procedure** GenerarGrafoBarrial($callejeroCSV, barrio$)
2: $V, E, w \leftarrow \emptyset, \emptyset, \emptyset$
3: **for** $fila \in callejeroCSV$ **do**
4: **if** $barrio \in \{fila.barrio\} \cup \{fila.barrio\_impar\} \cup \{fila.barrio\_par\}$ **then**
5: $p_0, \ldots, p_k \leftarrow fila.WKT$
6: $V \leftarrow V \cup \{p_0, p_k\}$
7: **if** $fila.sentido =$ CRECIENTE **then**
8: $E \leftarrow E \cup \{(p_0, p_k)\}$
9: $w \leftarrow w \cup \{((p_0, p_k), fila.longitud)\}$
10: **else if** $fila.sentido =$ DECRECIENTE **then**
11: $E \leftarrow E \cup \{(p_k, p_0)\}$
12: $w \leftarrow w \cup \{((p_k, p_0), fila.longitud)\}$
13: **else if** $fila.sentido =$ DOBLE **then**
14: $E \leftarrow E \cup \{(p_0, p_k), (p_k, p_0)\}$
15: $w \leftarrow w \cup \{((p_0, p_k), fila.longitud), ((p_k, p_0), fila.longitud)\}$
16: **end if**
17: **end if**
18: **end for**
19: **return** $\langle V, E, w \rangle$
20: **end procedure**

Luego, tenemos nuestro grafo con pesos asociados. Durante nuestra experimentacion, nos dimos cuenta de que había casos en los cuales se genereraban instancias infactibles, como

por ejemplo, el bus salía de un vértice que no tenía aristas de salida ya que habían sido filtrados por nuestra generación de grafo. Por lo tanto, decidimos quedarnos en cada caso con la componente fuertemente conexa más grande que tuviera nuestro grafo. Para esto utilizamos el algoritmo de Kosaraju [18]. Este algoritmo encuentra las componentes conexas en $\mathcal{O}(|V| + |E|)$, luego de lo cual eliminamos de V los vértices que no se encuentran en la componente mayor y de E y w los arcos que conectaban vértices eliminados.

4.1.3. Elección de paradas y estudiantes

Dado nuestro grafo de entrada, necesitamos elegir los vértices que serán potenciales paradas y ubicar los estudiantes. Esto se hace de forma aleatoria. Dado $|P|$, se eligen aleatoria y equiprobablemente el conjunto P, dentro del cual se elige aleatoriamente el vértice v_E que representa a la escuela y el subconjunto P_G que representa a los potenciales puntos de partida de un bus.

Luego, es momento de elegir los estudiantes. Para simplificar la generación, nuestro criterio de si a un estudiante s se le puede asignar una parada p (es decir, $p \in paradas(s)$) es si la distancia entre la parada y el estudiante no supera un valor aleatorio c_{max}. Entonces, ubicamos los estudiantes de la siguiente forma:

procedure GenerarEstudiantes(n, P, c_{max})
  $c^{\circ}_{max} \leftarrow \frac{c_{max}}{78710}$ ▷ Conversión burda de metros a grados en Buenos Aires
  $S \leftarrow \emptyset$
  for $i \in [1 \dots n]$ **do**
    $p \leftarrow \mathcal{U}(P)$
    repeat
      $s_{lat} \leftarrow \mathcal{U}(p_{lat} - c^{\circ}_{max}, p_{lat} + c^{\circ}_{max})$
      $s_{lon} \leftarrow \mathcal{U}(p_{lon} - c^{\circ}_{max}, p_{lon} + c^{\circ}_{max})$
    until $Distancia(p, s) \leq c_{max}$
    $S \leftarrow S \cup \{s\}$
  end for
  return S
end procedure

Es importante notar que la función $Distancia(x, y)$ no representa la distancia euclideana clásica. Como estamos tratando con puntos sobre la Tierra, es necesario usar la fórmula del semiverseno para cálcular la distancia de círculo máximo entre los dos puntos. Esta se define de la siguiente forma:

procedure Distancia(x, y)
  $\Delta_{lon} = x_{lon} - y_{lon}$
  $\Delta_{lat} = x_{lat} - y_{lat}$
  $a \leftarrow sen(\frac{\Delta_{lat}}{2})^2 + cos(x_{lat}) \times cos(y_{lat}) \times sen(\frac{\Delta_{lon}}{2})^2$
  $c \leftarrow 2 \times atan2(\sqrt{a}, \sqrt{1-a})$
  return $R \times c$ ▷ R es el radio de la Tierra
end procedure

4.2. Metodología de experimentación

Para las experiencias descritas en este capítulo, usamos una computadora hogareña de escritorio con un procesador Intel Core i7-7700K corriendo a 4.2 GHz, y 32 GB de memoria RAM. Para todos los modelos, creamos un script de Python 3 (corriendo con versión 3.6.8) que lee las instancias de un archivo de texto, hace el preprocesamiento acorde, y arma el modelo utilizando la API de CPLEX (corriendo en su versión 12.8.0). Luego, llama a CPLEX para resolver el problema. Para todas las instancias, dimos un tiempo límite de 1 hora para resolver el problema. Si el problema no está resuelto para entonces, la ejecución es terminada. En estos casos, utilizaremos la mejor solución encontrada y la mejor cota inferior encontrada por CPLEX para comparar los modelos. Cuanto más chica es esta brecha, se considera que el modelo tuvo mejor performance. CPLEX también provee una medida determinística de trabajo necesaria para resolver el problema llamda *ticks*. La utilizaremos como medida complementaria además del tiempo. Es importante notar que el tiempo presentado en los experimentos es solamente el tiempo de ejecución de CPLEX, y no incluye el tiempo de lectura, preprocesamiento y escritura de las instancias.

4.3. Variante MTZ para modelo con pre-cálculo de distancias

En esta sección haremos una comparación de las dos variantes de nuestro modelo con pre-cálculo de distancias. La primera variante tiene una familia de restricciones de tamaño de orden exponencial sobre la cantidad de paradas. Como es impráctico incluir a todas ellas en el modelo, se van agregando a medida que una solución entera encontrada las viola. Si bien este modelo, en general, presenta una relajación lineal más acotada, si $|P|$ es grande, podemos tener una cantidad de restricciones de este tipo muy grande, haciendo a nuestro modelo pesado para procesar en cada subnodo. Para esto, desarrollamos una variante que utiliza las restricciones presentadas en el trabajo de Miller, Tucker y Zemlin [19], cuya familia tiene orden cuadrático sobre la cantidad de paradas. Nuestra hipótesis es que este modelo va a resultar conveniente, especialmente cuando la cantidad de rutas posibles sea grande. Otra ventaja de esta variante es que, al no necesitar configurar *lazy constraints*, es posible utilizar varios hilos de procesamiento y eliminar el *overhead* de escribir la llamada en Python. Para comparar estas variantes, generamos instancias aleatorias. Como la cantidad de variables y restricciones del modelo no cambia según el tamaño del barrio a evaluar, decidimos que con un barrio (Barracas) alcanza para sacar conclusiones. Al mismo tiempo, decidimos dejar constantes la capacidad de los buses (60) y la distancia máxima de caminata de los estudiantes (200). Pusimos un tiempo límite de una hora para la resolución de la instancia. Si no es suficiente, consideraremos la mejor solución encontrada y la cota dual como indicativos de performance. En la Tabla 4.1 se pueden ver los resultados de instancias resueltas con el tiempo necesario para resolver y luego, en la Tabla 4.2, los resultados de las instancias que no fueron resueltas.

Evaluemos los resultados obtenidos. De las 21 instancias generadas, 16 fueron resueltas por ambas variantes en el término de una hora y 4 no fueron resueltas por ninguna de las dos variantes. La instancia 11, por otra parte, es resuelta por la variante MTZ, pero no por la variante DFJ. De las instancias resueltas, sacando instancias excepcionales (9, 16 y 19), usando las restricciones MTZ se obtiene una mejor performance: en promedio, reduce 18.52 % el tiempo de procesamiento y consume 57.44 % menos ticks. De las instancias no resueltas, podemos ver que en todos los casos, la variante MTZ obtiene una mejor cota

id	$\|P\|$	$\|S\|$	$\|P_G\|$	Sol. óptima	T. DFJ	Ticks DFJ	T. MTZ	Ticks MTZ	Δ_T	Δ_{ticks}
1	15	100	2	12,018.60	0.25	345.13	**0.20**	**144.86**	-20.00 %	-58.03 %
2	15	200	4	15,558.30	1.64	2,692.71	**0.63**	**565.47**	-61.59 %	-79.00 %
3	15	300	6	18,551.12	113.48	168,942.25	**5.38**	**4,029.77**	-95.26 %	-97.61 %
4	15	400	8	22,508.90	24.98	35,809.58	**0.78**	**787.47**	-96.88 %	-97.80 %
5	20	100	2	17,173.73	0.42	606.10	**0.41**	**301.48**	-2.38 %	-50.26 %
6	20	200	4	18,766.08	0.95	1,610.33	**0.58**	**518.18**	-38.95 %	-67.82 %
7	20	300	6	22,684.30	1,948.33	2,934,320.67	**245.06**	**183,038.37**	-87.42 %	-93.76 %
8	20	400	8	27,614.83	2,598.31	3,328,083.12	**209.92**	**142,824.63**	-91.92 %	-95.71 %
9	25	100	2	20,839.40	**1.80**	**2,327.77**	5.06	3,740.54	181.11 %	60.69 %
10	25	200	4	22,769.20	13.48	20,507.72	**3.27**	**2,806.37**	-75.74 %	-86.32 %
11	25	300	6	25,657.61	—	—	**1,651.89**	**1,208,047.28**	—	—
13	30	100	2	21,195.49	**8.81**	**11,211.70**	16.33	12,347.36	85.36 %	10.13 %
14	30	200	4	25,390.56	513.66	640,809.90	**102.38**	**74,969.84**	-80.07 %	-88.30 %
16	35	100	2	22,963.88	**5.47**	**8,735.48**	6.45	4,895.00	17.92 %	-43.96 %
17	35	200	4	24,524.01	25.47	36,353.16	**8.98**	**6,380.70**	-64.74 %	-82.45 %
19	40	100	2	23,370.67	**1.08**	**1,636.52**	3.33	2,250.30	208.33 %	37.51 %
20	40	200	4	27,566.62	1,970.70	2,643,629.52	**511.66**	**361,039.48**	-74.04 %	-86.34 %

Tab. 4.1: Instancias resueltas por una variante del modelo con pre-cálculo de distancia

id	$\|P\|$	$\|S\|$	$\|P_G\|$	C. Dual DFJ	Sol. DFJ	C. Dual MTZ	Sol. MTZ	Δ_{cota}	$\Delta_{sol.}$
11	25	300	6	23,833.31	25,819.89	—	—	—	—
12	25	400	8	24,863.55	31,529.95	**27,275.09**	**27,910.82**	9.70 %	-11.48 %
15	30	300	6	23,532.80	33,897.50	**24,901.84**	**28,379.42**	5.82 %	-16.28 %
18	35	300	6	25,157.66	34,932.15	**26,018.10**	**29,515.05**	3.42 %	-15.51 %
21	40	300	6	25,851.77	41,701.83	**26,889.88**	**33,290.47**	4.02 %	-20.17 %

Tab. 4.2: Instancias no resueltas por una variante del modelo con pre-cálculo de distancia. Resultados después de 3600 segundos de cómputo.

dual y una mejor solución. En promedio, las cotas duales y las soluciones MTZ son 5.74 % y 15.86 % mejores, respectivamente. Es importante notar que las instancias donde la variante DFJ tiene mejor performance tienen $|P_G| = 2$, y que a medida que la cantidad de rutas posibles es mayor, la diferencia de performance aumenta a favor de la variante MTZ, tal como pensábamos. Basados en estos resultados, utilizaremos la variante con restricciones MTZ para toda experimentación posterior.

4.4. Variante de variables de flujo y cota de caminos mínimos para modelo directo

En esta sección evaluaremos la mejora en performance del modelo directo cuando se utiliza la variante de conectividad por variables de flujo, cuando se utiliza la cota de caminos mínimos propuesta en la Sección 3.1 y cuando se utilizan ambas estrategias simultáneamente. Creemos que la cota de caminos mínimos tendrá una gran mejora en la performance, en especial cuando la instancia sea muy poco densa[1], ya que esto permitiría acotar mucho (potencialmente por cero) gran cantidad de arcos. Por otra parte, creemos que la variante de conectividad por flujo debería tener mejor performance que la versión con *lazy constraints* por las mismas razones discutidas en la sección anterior. Como en este caso la cantidad de cuadras y esquinas de nuestra instancia es un factor clave en la performance de nuestro modelo, utilizaremos instancias basadas en tres barrios de la Ciudad de Buenos Aires, en orden ascendente de tamaño: San Telmo (205 cuadras y 114 esquinas), La Boca (591 cuadras y 268 esquinas) y Barracas (1231 cuadras y 568 esquinas). Al mismo tiempo, variaremos la cantidad de paradas, estudiantes y rutas de manera similar al experimento previo. Dejaremos fijas la capacidad de los buses (60) y la distancia de caminata máxima (200). En las Tablas 4.3 y 4.4, podemos ver los resultados de las instancias resueltas en menos de una hora y luego los resultados de las instancias que no fueron resueltas en ese tiempo.

Evaluemos los resultados obtenidos. Dado que todas las instancias del barrio de San Telmo fueron resueltas en menos de 2 segundos, no las analizaremos ya que creemos que empañaran un análisis valioso de los modelos. De las 12 instancias restantes, el modelo original resuelve 4, también resueltas por las otras variantes. En estas 4 instancias (2, 7, 8 y 9), la variante CM precisa en promedio 90.43 % menos tiempo y 92.96 % ticks para resolver la misma instancia. La variante CF por el contrario, empeora su performance en estas instancias, necesitando en promedio 427.03 % más tiempo y 610.90 % más ticks. Cuando se combinan estas dos estrategias, también el resultado es peor que el original, precisando en promedio 29.35 % más tiempo y 82.16 % más ticks. De las 8 instancias no resueltas por el modelo original, la cota por CM permite resolver 5 instancias (1, 3, 4, 10 y 11). Al mismo tiempo, en las 3 instancias restantes, la variante mejora en promedio la cota dual en 7.18 % y la mejor solución encontrada en 35.21 %. De manera similar, la variante de conectividad por flujo, permite resolver 2 de estas instancias. Al mismo tiempo, en las no resueltas, permite mejorar las cuotas duales y soluciones encontradas en promedio, 5.51 % y 32.18 %, respectivamente. Por otra parte, cuando combinamos ambas optimizaciones, nuestro modelo tiene la mejor performance. En la única instancia que no resuelve (6), mejora la cota dual y la solución encontrada contra cualquier otra alternativa. Salvando pocas excepciones en donde los tiempos de ejecución no superan los 5 segundos,

[1] La intuición de densidad es la proporción de las esquinas de nuestra instancia que son posibles paradas ($\frac{|P|}{|V|}$).

id	Barrio	$\|P\|$	$\|S\|$	$\|P_G\|$	Sol. óptima	T. Orig.	Ticks Orig.	T. Cam.Mín.	Ticks Cam.Mín.	T. Flujo	Ticks Flujo	T. CM+F	Ticks CM+F
1	Barracas	10	50	1	10,329.12	—	—	0.14	15.99	134.16	104,914.53	**0.08**	**53.20**
2	Barracas	20	50	1	14,650.44	2,386.84	1,730,608.51	**1.11**	**524.60**	608.42	470,029.95	3.03	2,417.18
3	Barracas	30	50	1	18,488.28	—	—	309.30	402,551.83	—	—	**265.72**	**218,711.76**
4	Barracas	10	100	2	11,335.47	—	—	2.50	1,408.60	—	—	**0.88**	**672.41**
5	Barracas	20	100	2	15,676.19	—	—	—	—	—	—	**94.81**	**75,698.80**
7	La Boca	10	50	1	6,765.13	4.09	2,654.21	**0.08**	**31.67**	9.20	7,182.15	0.09	51.48
8	La Boca	20	50	1	9,739.21	1.09	590.89	**0.13**	**47.52**	7.70	5,671.43	1.38	1,004.94
9	La Boca	30	50	1	10,453.30	0.78	390.96	**0.19**	**73.86**	8.98	6,200.70	3.03	2,175.66
10	La Boca	10	100	2	7,979.76	—	—	1.97	1,346.43	—	—	**0.69**	**489.65**
11	La Boca	20	100	2	11,572.07	—	—	59.06	57,615.44	—	—	**21.59**	**16,659.24**
12	La Boca	30	100	2	12,691.57	—	—	—	—	298.36	222,121.57	**58.61**	**42,979.86**
13	San Telmo	10	50	1	4,429.16	0.02	1.33	$<$ **0.01**	**0.98**	0.02	11.03	$<$ **0.01**	3.58
14	San Telmo	20	50	1	5,853.60	0.02	2.27	$<$ **0.01**	**2.04**	0.03	18.83	0.05	19.01
15	San Telmo	30	50	1	6,906.49	0.13	88.52	0.14	80.16	0.09	71.09	**0.08**	**72.03**
16	San Telmo	10	100	2	5,420.93	0.20	196.34	0.16	169.67	0.20	158.00	**0.13**	**109.60**
17	San Telmo	20	100	2	6,847.73	**0.13**	**112.62**	0.28	351.93	0.25	206.57	0.23	212.70
18	San Telmo	30	100	2	8,137.27	0.95	1,105.23	1.25	1,428.99	1.20	998.08	**0.50**	**426.92**

Tab. 4.3: Instancias resueltas por una variante del modelo directo

id	Barrio	$\|P\|$	$\|S\|$	$\|P_G\|$	Sol. óptima	C. Dual Orig.	Sol. Orig.	CD Cam.Mín.	Sol. Cam.Mín.	CD Flujo	Sol. Flujo	CD CM+F	Sol. CM+F
1	Barracas	10	50	1	10,329.12	9,656.36	10,416.93	—	—	—	—	—	—
3	Barracas	30	50	1	18,488.28	17,538.37	19,023.36	—	—	17,659.33	18,507.22	—	—
4	Barracas	10	100	2	11,335.47	7,746.03	28,217.43	—	—	9,918.69	11,335.47	—	—
5	Barracas	20	100	2	15,676.19	12,908.98	284,469.91	14,183.12	18,148.75	14,127.09	16,172.16	—	—
6	Barracas	30	100	2	20,270.30	16,434.46	23,190.64	18,192.43	21,294.04	17,422.73	20,828.44	**18,685.11**	**20,300.75**
10	La Boca	10	100	2	7,979.76	7,090.96	10,000.22	—	—	7,646.64	7,988.55	—	—
11	La Boca	20	100	2	11,572.07	11,317.26	12,069.37	—	—	11,175.85	11,572.07	—	—
12	La Boca	30	100	2	12,691.57	12,194.06	13,330.52	12,313.79	12,819.27	—	—	—	—

Tab. 4.4: Instancias no resueltas por una variante del modelo directo

esta alternativa es estrictamente mejor al resto en tiempo de cómputo. Por estos motivos, utilizaremos esta versión de nuestro modelo para experimentos posteriores.

4.5. Heurística para solución inicial

En esta sección evaluaremos la mejora en performance que se obtiene cuando se provee al modelo con una solución inicial. Esta solución inicial estará provista por la heurística descrita en la Sección 3.3. Esta heurística provee un conjunto de rutas y una asignación de estudiantes a paradas. Esta información es traducida a las variables adecuadas para cada uno de los tres modelos y provista a CPLEX. Creemos que esta técnica puede proveer alguna pequeña mejora, especialmente en casos donde la instancia sea grande y no sea fácil conseguir una solución razonable en poco tiempo. Para este experimento, utilizaremos combinaciones de un barrio grande (Barracas) y un barrio mediano (La Boca), 15, 25 y 35 paradas, 100, 200 y 300 estudiantes (para 2, 4 y 6 rutas de buses de capacidad 60, respectivamente).

Para el modelo con pre-cálculo de distancias, el modelo con la heurística que provee una solución inicial resuelve 15 de las 18 instancias. Por otro lado, el modelo sin la solución inicial resuelve 16. En los 15 casos resueltos por ambas variantes, el modelo con heurística consume en promedio 12.72 % menos tiempo y 11.14 % menos ticks. En las instancias no resueltas, el modelo con heurística mejora en promedio en 1.5 % la cota inferior del óptimo y en 1.13 % la mejor solución obtenida.

Para el modelo directo, ambas variantes resuelven 13 de las 18 instancias. En estos casos, el modelo con heurística consume en promedio 0.94 % más tiempo y 0.94 % más ticks. En las instancias no resueltas, el modelo con heurística mejora en promedio en 0.36 % la cota inferior del óptimo y en 0.51 % la mejor solución obtenida.

Para el modelo plano, ambas variantes resuelven 13 de las 18 instancias. En estos casos, el modelo con heurística consume en promedio 14.8 % menos tiempo y 10.17 % menos ticks. En las instancias no resueltas, el modelo con heurística mejora en promedio en 1.75 % la cota inferior del óptimo y en 0.69 % la mejor solución obtenida.

Podemos ver que según el modelo, la adición de una solución inicial utilizando una heurística constructiva puede mejorar la performance en varios niveles. Por un lado, creemos que la mejora brindada por la técnica para el modelo directo, no es suficiente para que valga la pena su implementación. En especial, teniendo en cuenta la performance en instancias como la 4, cuyo tiempo de cómputo con la heurística es más del doble. En el modelo con pre-cálculo de distancias, la implementación de la heurística obtiene resultados mejores, aunque hay una instancia que deja de resolver cuando esta es utilizada. Por este motivo, creemos que tampoco vale la pena utilizarla para este modelo, Por último, en el modelo plano pudimos ver una mejora relativamente consistente en los tiempos de cómputo y en las cotas para las instancias que no resolvió. Por este motivo, utilizaremos la heurística solamente para el modelo plano.

En las próximas páginas se pueden ver en detalle los resultados de las corridas de los tres modelos para las 18 instancias generadas.

id	Barrio	$\|P\|$	$\|S\|$	$\|P_G\|$	Sol. óptima	Sol. heur.	Δ_{heur}	T. s/heur	Ticks s/heur	T. c/heur	Ticks c/heur	Δ_T	Δ_{ticks}
1	Barracas	15	100	2	12,018.60	12,363.87	2.87 %	0.17	98.98	**0.13**	**91.22**	-23.53 %	-7.84 %
2	Barracas	15	200	4	15,558.30	15,904.34	2.22 %	0.81	685.00	**0.55**	**535.35**	-32.10 %	-21.85 %
3	Barracas	15	300	6	18,500.27	20,032.98	8.28 %	7.55	6,272.75	**2.92**	**2,329.27**	-61.32 %	-62.87 %
4	Barracas	25	100	2	20,571.37	20,835.56	1.28 %	**3.58**	**2,585.14**	3.88	2,845.70	8.38 %	10.08 %
5	Barracas	25	200	4	21,498.98	24,616.16	14.50 %	2.41	1,935.11	**1.72**	**1,373.67**	-28.63 %	-29.01 %
7	Barracas	35	100	2	21,556.71	22,411.32	3.96 %	7.34	6,102.56	**5.83**	**5,022.51**	-20.57 %	-17.70 %
8	Barracas	35	200	4	25,732.53	28,421.10	10.45 %	**12.19**	**8,383.80**	12.94	9,539.25	6.15 %	13.78 %
10	Boca	15	100	2	10,289.46	10,315.28	0.25 %	0.13	97.66	**0.09**	**65.67**	-30.77 %	-32.76 %
11	Boca	15	200	4	10,742.01	10,779.47	0.35 %	0.73	657.41	**0.70**	**536.91**	-4.11 %	-18.33 %
12	Boca	15	300	6	11,902.74	12,937.83	8.70 %	**4.26**	**3,200.98**	5.13	3,782.08	20.42 %	18.15 %
13	Boca	25	100	2	12,326.22	14,271.40	15.78 %	6.01	4,769.64	**4.47**	**3,429.50**	-25.62 %	-28.10 %
14	Boca	25	200	4	13,409.93	15,132.51	12.85 %	**70.13**	**50,616.22**	82.94	64,375.73	18.27 %	27.18 %
15	Boca	25	300	6	14,027.60	15,340.40	9.36 %	761.09	546,508.99	**685.13**	**479,789.63**	-9.98 %	-12.21 %
16	Boca	35	100	2	12,952.64	13,441.94	3.78 %	13.56	**9,922.65**	**13.08**	10,580.53	-3.54 %	6.63 %
17	Boca	35	200	4	15,050.07	17,238.65	14.54 %	171.20	120,819.56	**164.73**	**106,026.53**	-3.78 %	-12.24 %
18	Boca	35	300	6	15,822.47	17,993.10	13.72 %	**2,436.38**	**1,420,617.27**	—	—	—	—

Tab. 4.5: Instancias resueltas por el modelo con pre-cálculo de distancia sin y con heurística para solución inicial

id	Barrio	$\|P\|$	$\|S\|$	$\|P_G\|$	Sol. óptima	Sol. heur.	Δ_{heur}	C. Dual s/heur	Sol. s/heur	C. Dual c/heur	Sol c/heur	Δ_{cota}	$\Delta_{sol.}$
6	Barracas	25	300	6	—	29,562.73	—	24,311.80	26,491.21	**25,069.99**	**26,441.43**	3.12 %	-0.19 %
9	Barracas	35	300	6	—	31,180.29	—	**26,394.36**	30,867.05	26,363.13	**30,224.65**	-0.12 %	-2.08 %
18	Boca	35	300	6	15,822.47	17,993.10	13.72 %	—	—	15,475.17	15,822.47	—	—

Tab. 4.6: Instancias no resueltas por el modelo con pre-cálculo de distancia sin y con heurística para solución inicial

id	Barrio	$\|P\|$	$\|S\|$	$\|P_G\|$	Sol. óptima	Sol. heur.	Δ_{heur}	T. s/heur	Ticks s/heur	T. c/heur	Ticks c/heur	Δ_T	Δ_{ticks}
1	Barracas	15	100	2	12,018.60	12,363.87	2.87 %	11.94	8,585.52	**10.19**	**7,785.47**	-14.66 %	-9.32 %
2	Barracas	15	200	4	15,558.30	15,904.34	2.22 %	**14.02**	**10,720.82**	14.77	11,701.40	5.35 %	9.15 %
3	Barracas	15	300	6	18,500.27	20,032.98	8.28 %	1,279.02	732,533.18	**801.89**	**447,390.25**	-37.30 %	-38.93 %
4	Barracas	25	100	2	20,571.37	20,835.56	1.28 %	**348.28**	**263,266.23**	776.09	560,161.09	122.84 %	112.77 %
5	Barracas	25	200	4	21,498.98	24,616.16	14.50 %	60.33	43,516.07	**52.09**	**38,504.90**	-13.66 %	-11.52 %
7	Barracas	35	100	2	21,556.71	22,411.32	3.96 %	1,829.28	1,082,305.73	**1,716.61**	**885,837.14**	-6.16 %	-18.15 %
8	Barracas	35	200	4	25,732.53	28,421.10	10.45 %	2,567.20	1,174,100.87	**2,248.78**	**1,049,994.57**	-12.40 %	-10.57 %
10	Boca	15	100	2	10,289.46	10,315.28	0.25 %	6.94	5,207.55	**5.98**	**4,565.75**	-13.83 %	-12.32 %
11	Boca	15	200	4	10,742.01	10,779.47	0.35 %	15.03	12,102.08	**15.47**	**12,620.86**	2.93 %	4.29 %
12	Boca	15	300	6	11,902.74	12,937.83	8.70 %	557.33	388,981.39	**468.45**	**337,984.06**	-15.95 %	-13.11 %
13	Boca	25	100	2	12,326.22	14,271.40	15.78 %	35.44	30,691.80	**27.25**	**22,622.48**	-23.11 %	-26.29 %
14	Boca	25	200	4	13,409.93	15,132.51	12.85 %	**2,329.28**	**1,350,401.35**	3,117.61	1,842,191.81	33.84 %	36.42 %
16	Boca	35	100	2	12,952.64	13,441.94	3.78 %	450.81	341,744.19	**380.36**	**306,191.26**	-15.63 %	-10.40 %

Tab. 4.7: Instancias resueltas por el modelo directo sin y con heurística para solución inicial

id	Barrio	$\|P\|$	$\|S\|$	$\|P_G\|$	Sol. óptima	Sol. heur.	Δ_{heur}	C. Dual s/heur	Sol. s/heur	C. Dual c/heur	Sol c/heur	Δ_{cota}	$\Delta_{sol.}$
6	Barracas	25	300	6	—	29,562.73	—	23,024.79	26,496.46	**23,313.32**	**26,491.21**	1.25 %	-0.02 %
9	Barracas	35	300	6	—	31,180.29	—	25,813.48	**30,401.09**	**26,016.23**	30,833.82	0.79 %	1.42 %
15	Boca	25	300	6	14,027.60	15,340.40	9.36 %	**13,207.99**	14,672.38	13,119.08	**14,113.80**	-0.67 %	-3.81 %
17	Boca	35	200	4	15,050.07	17,238.65	14.54 %	14,751.86	**15,050.07**	**14,791.44**	15,050.94	0.27 %	0.01 %
18	Boca	35	300	6	16,171.83	17,993.10	11.26 %	15,080.31	16,193.68	**15,108.78**	**16,171.83**	0.19 %	-0.13 %

Tab. 4.8: Instancias no resueltas por el modelo directo sin y con heurística para solución inicial

id	Barrio	$\|P\|$	$\|S\|$	$\|P_G\|$	Sol. óptima	Sol. heur.	Δ_{heur}	T. s/heur	Ticks s/heur	T. c/heur	Ticks c/heur	Δ_T	Δ_{ticks}
1	Barracas	15	100	2	12,018.60	12,363.87	2.87 %	0.13	78.85	**0.09**	**54.08**	-30.77 %	-31.41 %
2	Barracas	15	200	4	15,558.30	15,904.34	2.22 %	8.17	5,710.91	**6.28**	**4,484.92**	-23.13 %	-21.47 %
3	Barracas	15	300	6	18,500.27	20,032.98	8.28 %	**2.22**	**1,622.51**	2.31	1,654.61	4.05 %	1.98 %
4	Barracas	25	100	2	20,571.37	20,835.56	1.28 %	2.31	1,662.18	**1.80**	**1,238.77**	-22.08 %	-25.47 %
5	Barracas	25	200	4	21,498.98	24,616.16	14.50 %	19.23	15,625.58	**10.67**	**8,099.40**	-44.51 %	-48.17 %
7	Barracas	35	100	2	21,556.71	22,411.32	3.96 %	213.30	188,605.06	**168.05**	**146,523.72**	-21.21 %	-22.31 %
10	Boca	15	100	2	10,289.46	10,315.28	0.25 %	0.19	69.67	**0.09**	**57.91**	-52.63 %	-16.88 %
11	Boca	15	200	4	10,742.01	10,779.47	0.35 %	**0.09**	**55.64**	0.13	95.29	44.44 %	71.26 %
12	Boca	15	300	6	11,902.74	12,937.83	8.70 %	0.39	240.03	**0.33**	**230.77**	-15.38 %	-3.86 %
13	Boca	25	100	2	12,326.22	14,271.40	15.78 %	14.55	11,556.99	**13.31**	**10,574.73**	-8.52 %	-8.50 %
14	Boca	25	200	4	13,409.93	15,132.51	12.85 %	269.28	233,370.96	**244.81**	**208,713.98**	-9.09 %	-10.57 %
15	Boca	25	300	6	14,027.60	15,340.40	9.36 %	319.05	290,973.46	**298.20**	**257,516.01**	-6.54 %	-11.50 %
16	Boca	35	100	2	12,952.64	13,441.94	3.78 %	759.88	595,256.43	**706.70**	**563,381.29**	-7.00 %	-5.35 %

Tab. 4.9: Instancias resueltas por el modelo plano sin y con heurística para solución inicial

id	Barrio	$\|P\|$	$\|S\|$	$\|P_G\|$	Sol. óptima	Sol. heur.	Δ_{heur}	C. Dual s/heur	Sol. s/heur	C. Dual c/heur	Sol c/heur	Δ_{cota}	$\Delta_{sol.}$
6	Barracas	25	300	6	—	29,562.73	—	24,910.18	26,491.21	**25,521.26**	**26,154.28**	2.45 %	-1.27 %
8	Barracas	35	200	4	25,732.53	28,421.10	10.45 %	24,472.78	**25,732.53**	**25,006.36**	**25,732.53**	2.18 %	0.00 %
9	Barracas	35	300	6	—	31,180.29	—	24,503.09	**29,862.13**	**24,901.37**	**29,862.13**	1.63 %	0.00 %
17	Boca	35	200	4	15,050.07	17,238.65	14.54 %	14,430.14	**15,050.07**	**14,494.46**	**15,050.07**	0.45 %	0.00 %
18	Boca	35	300	6	15,822.47	17,993.10	13.72 %	14,325.45	16,171.83	**14,619.49**	**15,822.47**	2.05 %	-2.16 %

Tab. 4.10: Instancias no resueltas por el modelo plano sin y con heurística para solución inicial

4.6. Heurística de redondeo en nodos

En esta sección evaluaremos la mejora en performance que provee la heurística de redondeo en nodos. Dada su relativa simplicidad de implementación sobre el modelo con pre-cálculo de distancias, decidimos unicamente adaptarlo a este último y si viéramos una mejora considerable, la extenderíamos a los modelos restantes. Utilizamos instancias sobre los dos barrios mas grandes (Barracas y La Boca), 15, 25 y 35 paradas y 100, 200 y 300 estudiantes con 2, 4 y 6 rutas, respectivamente. Las instancias son las mismas que utilizamos en la sección anterior para evaluar la performance de la heurística para generar una solución inicial. Creemos que no vamos a ver una mejora muy grande, dado el overhead de las llamadas escritas en Python, que ralentizan el cómputo.

De las 18 instancias evaluadas, el modelo con heurística de redondeo resuelve 13 instancias, que son también resueltas por el modelo base. Además de estas, el modelo base resuelve 3 instancias extra. En las instancias resueltas por ambos modelos, el modelo con nuestra heurística consume 11.74 veces más tiempo y 8.44 veces más ticks. Al mismo tiempo, en las instancias no resueltas por ninguno de los dos modelos, la versión con nuestra heurística tiene una cota inferior 3.86 % peor, y la solución obtenida es 1.53 % mejor.

Si bien la heurística de redondeo en nodos mejora ocasionalmente la solución incumbente, creemos que estos resultados están muy lejos de justificar su implementación. Por lo tanto, no extenderemos su implementación a los otros dos modelos y no utilizaremos esta heurística para futuras comparaciones del modelo con pre-cálculo de distancias.

En la siguiente página se pueden observar los resultados de los experimentos en detalle.

id	Barrio	$\|P\|$	$\|S\|$	$\|P_G\|$	Sol. óptima	T. s/heur	Ticks s/heur	T. c/heur	Ticks c/heur	Δ_T	Δ_{ticks}
1	Barracas	15	100	2	12,018.60	**0.17**	**98.98**	0.99	424.93	5.82x	4.29x
2	Barracas	15	200	4	15,558.30	**0.81**	**685.00**	1.19	1,362.07	1.47x	1.99x
3	Barracas	15	300	6	18,500.27	**7.55**	**6,272.75**	49.22	23,664.64	6.52x	3.77x
4	Barracas	25	100	2	20,571.37	**3.58**	**2,585.14**	121.81	41,964.36	34.03x	16.23x
5	Barracas	25	200	4	21,498.98	**2.41**	**1,935.11**	4.55	4,768.11	1.89x	2.46x
7	Barracas	35	100	2	21,556.71	**7.34**	**6,102.56**	37.16	20,165.02	5.06x	3.3x
8	Barracas	35	200	4	25,732.53	**12.19**	**8,383.80**	318.45	182,551.29	26.12x	21.77x
10	Boca	15	100	2	10,289.46	**0.13**	**97.66**	0.23	209.52	1.77x	2.15x
11	Boca	15	200	4	10,742.01	**0.73**	**657.41**	2.80	2,381.27	3.84x	3.62x
12	Boca	15	300	6	11,902.74	**4.26**	**3,200.98**	63.74	57,362.17	14.96x	17.92x
13	Boca	25	100	2	12,326.22	**6.01**	**4,769.64**	88.95	41,216.08	14.8x	8.64x
14	Boca	25	200	4	13,409.93	**70.13**	**50,616.22**	1,328.17	730,040.77	18.94x	14.42x
15	Boca	25	300	6	14,027.60	**761.09**	**546,508.99**	—	—	—	—
16	Boca	35	100	2	12,952.64	**13.56**	**9,922.65**	188.25	90,844.29	13.88x	9.16x
17	Boca	35	200	4	15,050.07	**171.20**	**120,819.56**	—	—	—	—
18	Boca	35	300	6	16,171.83	**2,436.38**	**1,420,617.27**	—	—	—	—

Tab. 4.11: Instancias resueltas por una variante del modelo con pre-cálculo de distancia con y sin heurística de redondeo en nodos

id	Barrio	$\|P\|$	$\|S\|$	$\|P_G\|$	Sol. óptima	C. Dual s/heur	Sol. s/heur	C. Dual c/heur	Sol c/heur	Δ_{cota}	$\Delta_{sol.}$
6	Barracas	25	300	6	—	**24,311.80**	**26,491.21**	23,141.10	**26,491.21**	-4.82 %	0.00 %
9	Barracas	35	300	6	—	**26,394.36**	30,867.05	25,627.27	**29,923.32**	-2.91 %	-3.06 %
15	Boca	25	300	6	14,027.60	—	—	12,537.78	14,113.80	—	—
17	Boca	35	200	4	15,050.07	—	—	14,227.51	15,050.07	—	—
18	Boca	35	300	6	16,171.83	—	—	14,771.75	16,171.83	—	—

Tab. 4.12: Instancias no resueltas por una variante del modelo con pre-cálculo de distancia con y sin heurística de redondeo en nodos

4.7. Clústers de estudiantes

En esta sección evaluaremos la mejora en performance que se obtiene cuando se modifica al modelo agrupando los estudiantes según su imagen de la función *paradas*. Esperamos obtener una mejora sustancial, en especial cuando la cantidad de estudiantes sea grande, y potencialmente podamos colapsar muchas variables binarias de asignación de cada estudiante a paradas en variables enteras de cantidad de estudiantes. Para este experimento, consideramos que los parámetros más convenientes a variar son la cantidad de estudiantes (100, 200 y 300 con 2, 4 y 6 buses, respectivamente), la distancia máxima de caminata (200m, 400m y 800m) y el barrio (Barracas y La Boca). Dejaremos la cantidad de paradas constante en 25 y la capacidad de los buses constante en 60.

Para el modelo con pre-cálculo de distancias, la versión con clusterización de estudiantes resolvió las 18 instancias en el lapso de una hora. Por otro lado, la versión base no llegó a resolver dos de ellas. En las 16 instancias resueltas por ambas variantes, la versión con la optimización precisó en promedio 4.21 % menos tiempo y 5.14 % más ticks para resolverlas. Si bien esto no parece una mejora considerable, encontramos dos casos en los cuales la mejora empeora muy significativamente la performance (2 y 18). Creemos que estos casos pueden ser no representativos de nuestra mejora, ya que en los otras instancias, la mejora es consistente. Si removiéramos estos dos casos de la muestra, la mejora sería del 32.32 % en tiempo y 29.60 % en términos de ticks.

Para el modelo directo, la versión básica resolvió 16 de las 18 instancias en el lapso de una hora. Por otro lado, la versión con nuestra optimización logró resolver solo 15 de esas 16. En las 15 instancias resueltas por ambas variantes, la versión con la optimización precisó en promedio 17.59 % menos tiempo y 12.43 % menos ticks para resolverlas. En las 2 instancias no resueltas por ninguna versión, la versión optimizada mejoró 0.99 % la cota inferior y 1.58 % la solución obtenida, en promedio. Aún tomando en cuenta que hubo una instancia que la versión básica resuelve y la optimizada no, creemos que la mejora de esta optimización es sustancial.

Para el modelo plano, la versión básica de nuestro modelo resuelve 12 instancias de las 18. De estas 12 instancias, la versión optimizada logra resolver 11. En estas instancias, con la optimización se consigue una mejora del 26.28 % en tiempo y 28.26 % en ticks. Luego, la versión optimizada resuelve otras 3 instancias que no fue posible resolver con la versión básica. 3 de las 18 instancias no fueron resueltas por ninguna de las variantes. En estas, cuando utilizamos la versión optimizada, la cota inferior se mejora en un 0.53 % y la solución obtenida en un 0.27 %, en promedio. Creemos que estos resultados son muy significativos tanto en instancias resueltas, como en tiempo de cómputo y en cotas obtenidas.

Creemos que la optimización presenta una mejora sustancial en la performance de nuestros modelos. Por lo tanto, utilizaremos esta optimización en todos los casos de experimentación posterior.

En las siguientes páginas se pueden observar los resultados en detalle.

id	Barrio	$\|S\|$	$\|P_G\|$	d_c	Sol. óptima	T. s/clús.	Ticks s/clús.	T. c/clús.	Ticks c/clús.	Δ_T	Δ_{ticks}
1	Barracas	100	2	200	20,145.50	5.84	4,337.41	**5.30**	**4,096.02**	-9.25 %	-5.57 %
2	Barracas	100	2	400	17,842.53	**2.08**	**1,647.80**	4.36	3,133.86	109.62 %	90.18 %
3	Barracas	100	2	800	15,331.95	0.66	587.30	**0.64**	**502.66**	-3.03 %	-14.41 %
4	Barracas	200	4	200	22,482.19	**2.72**	2,232.31	2.77	**2,098.37**	1.84 %	-6.00 %
5	Barracas	200	4	400	20,814.59	2.50	2,398.73	**1.91**	**1,720.86**	-23.60 %	-28.26 %
6	Barracas	200	4	800	16,616.79	4.74	4,498.97	**3.73**	**3,619.99**	-21.31 %	-19.54 %
7	Barracas	300	6	200	26,160.62	—	—	**2,255.42**	**1,565,153.05**	—	—
8	Barracas	300	6	400	25,653.49	—	—	**2,922.45**	**2,149,575.82**	—	—
9	Barracas	300	6	800	19,715.60	299.45	162,807.03	**30.00**	**26,903.52**	-89.98 %	-83.48 %
10	Boca	100	2	200	12,150.18	3.75	2,795.70	**2.64**	**1,948.96**	-29.60 %	-30.29 %
11	Boca	100	2	400	10,088.59	13.09	10,744.73	**9.55**	**7,401.25**	-27.04 %	-31.12 %
12	Boca	100	2	800	6,911.96	4.63	3,370.06	**1.97**	**1,503.92**	-57.45 %	-55.37 %
13	Boca	200	4	200	13,409.93	84.97	59,481.43	**54.03**	**39,825.06**	-36.41 %	-33.05 %
14	Boca	200	4	400	12,072.12	29.14	21,796.70	**21.27**	**17,087.60**	-27.01 %	-21.60 %
15	Boca	200	4	800	9,800.79	**30.56**	**21,485.27**	35.38	31,549.65	15.77 %	46.84 %
16	Boca	300	6	200	14,195.12	564.64	381,975.56	**173.61**	**131,814.18**	-69.25 %	-65.49 %
17	Boca	300	6	400	12,174.62	174.88	97,266.27	**41.66**	**32,003.76**	-76.18 %	-67.10 %
18	Boca	300	6	800	10,619.63	**82.16**	**42,118.54**	308.50	213,321.61	275.49 %	406.48 %

Tab. 4.13: Instancias resueltas por una variante del modelo con pre-cálculo de distancia con y sin clusterización de estudiantes

id	Barrio	$\|S\|$	$\|P_G\|$	d_c	Sol. óptima	C. Dual s/clús.	Sol. s/clús.	C. Dual c/clús.	Sol c/clús.	Δ_{cota}	$\Delta_{sol.}$
7	Barracas	300	6	200	26,160.62	24,937.61	26,418.09	—	—	—	—
8	Barracas	300	6	400	25,653.49	23,844.01	25,863.33	—	—	—	—

Tab. 4.14: Instancias no resueltas por una variante del modelo con pre-cálculo de distancia con y sin clusterización de estudiantes

id	Barrio	$\|S\|$	$\|P_G\|$	d_c	Sol. óptima	T. s/clús.	Ticks s/clús.	T. c/clús.	Ticks c/clús.	Δ_T	Δ_{ticks}
1	Barracas	100	2	200	20,145.50	848.89	508,428.62	**655.70**	**430,702.69**	-22.76 %	-15.29 %
2	Barracas	100	2	400	17,842.53	**187.72**	**139,774.94**	276.63	225,190.98	47.36 %	61.11 %
3	Barracas	100	2	800	15,331.95	120.78	95,731.48	**55.52**	**45,008.95**	-54.03 %	-52.98 %
4	Barracas	200	4	200	22,482.19	140.03	94,391.94	**113.09**	**73,279.57**	-19.24 %	-22.37 %
5	Barracas	200	4	400	20,814.59	62.28	47,603.32	**32.84**	**26,312.96**	-47.27 %	-44.72 %
6	Barracas	200	4	800	16,616.79	57.88	43,522.49	**12.55**	**9,738.91**	-78.32 %	-77.62 %
10	Boca	100	2	200	12,150.18	25.06	21,063.13	**18.02**	**14,754.37**	-28.09 %	-29.95 %
11	Boca	100	2	400	10,088.59	**55.02**	**45,478.41**	61.94	53,963.25	12.58 %	18.66 %
12	Boca	100	2	800	6,911.96	72.77	59,967.56	**53.50**	**44,090.43**	-26.48 %	-26.48 %
13	Boca	200	4	200	13,409.93	**2,844.63**	**1,473,927.28**	—	—	—	—
14	Boca	200	4	400	12,072.12	**216.97**	**143,903.72**	252.33	183,314.07	16.30 %	27.39 %
15	Boca	200	4	800	9,800.79	**943.55**	**477,971.14**	1,727.53	907,479.46	83.09 %	89.86 %
17	Boca	300	6	400	12,174.62	1,012.81	479,911.46	**385.72**	**223,020.04**	-61.92 %	-53.53 %
18	Boca	300	6	800	10,619.63	1,658.83	669,408.07	**831.45**	**430,877.40**	-49.88 %	-35.63 %

Tab. 4.15: Instancias resueltas por una variante del modelo directo con y sin clusterización de estudiantes

id	Barrio	$\|S\|$	$\|P_G\|$	d_c	Sol. óptima	C. Dual s/clús.	Sol. s/clús.	C. Dual c/clús.	Sol c/clús.	Δ_{cota}	$\Delta_{sol.}$
7	Barracas	300	6	200	26,160.62	22,982.04	26,676.62	**23,206.90**	**26,333.06**	0.98 %	-1.29 %
8	Barracas	300	6	400	25,653.49	22,251.19	26,160.97	**22,270.45**	**25,675.92**	0.09 %	-1.85 %
9	Barracas	300	6	800	19,715.60	18,682.18	20,359.15	**19,073.54**	**19,715.60**	2.09 %	-3.16 %
13	Boca	200	4	200	13,409.93	—	—	13,202.83	13,409.93	—	—
16	Boca	300	6	200	14,195.12	13,216.70	**14,195.12**	**13,324.57**	**14,195.12**	0.82 %	0.00 %

Tab. 4.16: Instancias no resueltas por una variante del modelo directo con y sin clusterización de estudiantes

id	Barrio	$\|S\|$	$\|P_G\|$	d_c	Sol. óptima	T. s/clús.	Ticks s/clús.	T. c/clús.	Ticks c/clús.	Δ_T	Δ_{ticks}
1	Barracas	100	2	200	20,145.50	11.06	9,023.77	**9.25**	**7,162.00**	-16.37 %	-20.63 %
2	Barracas	100	2	400	17,842.53	2.36	1,806.22	**1.66**	**1,226.77**	-29.66 %	-32.08 %
3	Barracas	100	2	800	15,331.95	**0.84**	**753.02**	1.05	894.69	25.00 %	18.81 %
4	Barracas	200	4	200	22,482.19	98.36	93,940.51	**76.53**	**73,866.38**	-22.19 %	-21.37 %
5	Barracas	200	4	400	20,814.59	34.17	32,183.20	**17.55**	**15,354.30**	-48.64 %	-52.29 %
6	Barracas	200	4	800	16,616.79	2134.08	2,181,299.36	**749.20**	**769,695.88**	-64.89 %	-64.71 %
7	Barracas	300	6	200	26,160.62	—	—	**1,839.48**	**1,880,740.17**	—	—
9	Barracas	300	6	800	19,715.60	—	—	**1,075.23**	**1,167,706.81**	—	—
10	Boca	100	2	200	12,150.18	10.92	9,001.76	**6.72**	**5,070.46**	-38.46 %	-43.67 %
11	Boca	100	2	400	10,088.59	21.69	20,538.43	**18.95**	**17,674.39**	-12.63 %	-13.94 %
12	Boca	100	2	800	6,911.96	**73.38**	**75,610.43**	73.58	78,093.20	0.27 %	3.28 %
13	Boca	200	4	200	13,409.93	316.92	316,578.09	**237.17**	**227,674.52**	-25.16 %	-28.08 %
14	Boca	200	4	400	12,072.12	**3,500.81**	**3,377,557.93**	—	—	—	—
16	Boca	300	6	200	14,195.12	795.66	706,301.84	**347.14**	**309,497.49**	-56.37 %	-56.18 %
17	Boca	300	6	400	12,174.62	—	—	**2,243.44**	**2,334,211.56**	—	—

Tab. 4.17: Instancias resueltas por una variante del modelo plano con y sin clusterización de estudiantes

id	Barrio	$\|S\|$	$\|P_G\|$	d_c	Sol. óptima	C. Dual s/clús.	Sol. s/clús.	C. Dual c/clús.	Sol c/clús.	Δ_{cota}	$\Delta_{sol.}$
7	Barracas	300	6	200	26,160.62	23,837.28	26,575.28	—	—	—	—
8	Barracas	300	6	400	25,653.49	**22,392.15**	25,863.33	22,079.61	**25,653.49**	-1.40 %	-0.81 %
9	Barracas	300	6	800	19,715.60	18,342.83	19,715.60	—	—	—	—
14	Boca	200	4	400	12,072.12	—	—	11,185.87	12,072.12	—	—
15	Boca	200	4	800	9,800.79	8,697.17	**9,800.79**	**8,950.32**	**9,800.79**	2.91 %	0.00 %
17	Boca	300	6	400	12,174.62	11,246.84	12,174.62	—	—	—	—
18	Boca	300	6	800	10,619.63	8,859.95	**10,619.63**	**8,865.94**	**10,619.63**	0.07 %	0.00 %

Tab. 4.18: Instancias no resueltas por una variante del modelo plano con y sin clusterización de estudiantes

4.8. Comparación general

Habiendo ya evaluado la performance de todas nuestras mejoras, decidimos comparar entre sí todos los modelos obtenidos con todas las que consideramos mejoran su performance. Recordemos entonces, que tenemos las siguientes variantes:

- Modelo con pre-cálculo de estudiantes. Variante con restricciones MTZ y clusterización de estudiantes.
- Modelo directo. Variante de conectividad por variables de flujo, corte de flujo máximo y clusterización de estudiantes.
- Modelo plano. Variante con heurística para solución inicial y clusterización de estudiantes.

En esta sección realizaremos dos experimentos, variando diferentes parámetros de las instancias a resolver. Esperamos entonces ver fortalezas de algunos modelos en instancias particulares.

4.8.1. Cantidad de paradas

Sabemos que la dificultad de una instancia aumenta con la cantidad de posibles paradas disponibles. La mayor cantidad de potenciales paradas implica que nuestro modelo tiene que "decidir" entre más paradas para asignar un estudiante, además de separar paradas por rutas y luego encontrar el camino más corto entre ellas. Esperamos que esto tenga un impacto negativo en la performance de todos nuestros modelos. Sin embargo, consideramos que hay modelos que se ven menos impactados por esto que otros. Por ejemplo, para el modelo con pre-cálculo de distancias y el modelo plano, el número de variables de decisión en ellos aumenta cuadráticamente con la cantidad de potenciales paradas. Por otro lado, en el modelo directo, sólo crece linealmente. Pensamos que en casos con gran cantidad de paradas y un barrio chico, el modelo directo podría presentar una ventaja. Creemos que esta ventaja se acentuaría en modelos donde la cantidad de calles y esquinas es pequeña, como San Telmo o La Boca en menor medida.

Para estos experimentos, utilizamos los tres barrios que separamos (Barracas, La Boca y San Telmo), variamos la cantidad de paradas (20, 40, 60, 80 y 100). Por otro lado, dejamos constantes la cantidad de estudiantes en 200, la cantidad de buses en 4, la capacidad de ellos en 50 y la distancia máxima de caminata en 200m.

Podemos ver varios resultados interesantes. En las instancias del barrio de Barracas, el modelo con pre-cálculo de distancias es el único que puede resolver las instancias de 40 y 60 paradas. Aunque todos pueden resolver la instancia de 20 paradas, este modelo presenta la mejor performance por cerca de un orden de magnitud. Aún así, el modelo directo encuentra la solución óptima en las instancias de 40 y 60 paradas, aunque no logra demostrar su optimalidad. Ninguno de los tres modelos resuelve las instancias de 80 o 100 paradas. En ambos casos, el modelo con pre-cálculo de distancias obtiene la mejor cota inferior de la solución. Sin embargo, en el caso de 80 paradas, la mejor solución es encontrada por el modelo directo y en el de 100, por el modelo plano. En ambos casos, la brecha entre la cota inferior y la mejor solución encontrada es mejor para el modelo directo. Creemos que esto va en línea con la hipótesis de que la performance del modelo directo superaría a los otros conforme aumentáramos la cantidad de paradas.

En las instancias del barrio de La Boca, el modelo con pre-cálculo de distancias es el único que puede resolver las instancias de 20 y 40 paradas. Si bien el modelo plano también resuelve la instancia de 20 paradas, su performance es empeorada por un factor cercano a 7. En la instancia de 60 paradas, el modelo con pre-cálculo de distancias obtiene ambas mejores cota inferior y solución. Naturalmente, también obtiene la mejor brecha entre ellas, seguida por la del modelo directo. En la instancia de 80 paradas, este modelo también obtiene la mejor solución, pero la mejor cota es encontrada por el modelo directo. Este último modelo también consigue la mejor brecha, seguido del modelo con pre-cálculo. En la instancia de 100 paradas, el modelo directo obtiene ambas mejores cota inferior y solución. Naturalmente, también obtiene la mejor brecha.

Por último, en las instancias del barrio de San Telmo, el modelo directo puede resolver las instancias de 20, 40, 60 y 80 paradas. Los otros dos modelos también resuelven la instancia de 20 paradas. El modelo con pre-cálculo de distancias presenta una performance similar en este caso, pero el modelo plano tiene una performance por lo menos un orden de magnitud peor. Ningún modelo logra resolver la instancia de 100 paradas, aunque el modelo directo obtiene ambas mejores cota inferior y solución obtenida. Naturalmente, también obtiene la mejor brecha.

Podemos obtener varias conclusiones sobre este experimento. Como pensábamos, la performance de todos los modelos sufriría un fuerte impacto negativo cuando aumentáramos la cantidad de paradas. Sin embargo, vimos que el modelo directo mantenía una mejor performance cuando había muchas paradas. Vimos que este efecto se acentúa cuando el barrio es chico, como es el caso de San Telmo. En las instancias de este barrio, el modelo directo permitió resolver instancias para las cuáles otros modelos dejaron brechas considerables. Por otro lado, vimos que para cantidades bajas de paradas, el modelo con pre-cálculo de distancias obtiene una performance muy buena en comparación al resto.

En la siguiente página se pueden ver en detalle los resultados obtenidos.

id	Barrio	$\|P\|$	Sol. óptima	T. pre-cálc.	Ticks pre-cálc.	T. directo	Ticks directo	T. plano	Ticks plano
1	Barracas	20	19,052.24	**0.47**	**380.97**	21.03	15,780.61	3.44	2,554.30
2	Barracas	40	27,231.37	**107.66**	**78,379.75**	—	—	—	—
3	Barracas	60	29,444.54	**1,765.33**	**873,672.32**	—	—	—	—
6	Boca	20	13,195.07	**10.61**	**8,585.69**	—	—	77.28	65,803.84
7	Boca	40	15,190.50	**1,814.20**	**661,866.81**	—	—	—	—
11	San Telmo	20	7,822.84	**0.95**	**858.35**	1.56	1,091.64	35.22	29,615.46
12	San Telmo	40	10,193.11	—	—	**1,005.67**	**820,061.15**	—	—
13	San Telmo	60	9,209.76	—	—	**891.94**	**622,185.77**	—	—
14	San Telmo	80	9,174.15	—	—	**1,690.16**	**1,091,591.62**	—	—

Tab. 4.19: Instancias resueltas por el modelo con pre-cálculo, el modelo directo o el modelo plano

id	Barrio	$\|P\|$	Sol. óptima	C.D. pre-cálc.	Sol. pre-cálc.	C.D. directo	Sol. directo	C.D. plano	Sol. plano
2	Barracas	40	27,231.37	—	—	26,644.19	27,231.37	24,513.62	27,282.83
3	Barracas	60	29,444.54	—	—	28,740.40	29,444.54	26,642.00	30,540.62
4	Barracas	80	—	**28,917.85**	31,915.93	28,675.26	**29,812.25**	27,199.16	30,641.90
5	Barracas	100	—	**26,103.96**	29,629.41	25,966.40	28,662.70	25,233.64	**28,310.64**
6	Boca	20	13,195.07	—	—	13,075.50	13,195.07	—	—
7	Boca	40	15,190.50	—	—	14,447.56	15,795.57	13,964.93	15,238.40
8	Boca	60	—	**15,039.31**	**15,261.21**	14,720.65	15,432.21	13,188.13	15,721.05
9	Boca	80	—	14,161.47	**16,721.65**	**14,481.71**	16,900.37	12,841.62	17,188.73
10	Boca	100	—	12,248.19	16,256.08	**12,644.63**	**16,102.49**	11,304.10	16,746.04
12	San Telmo	40	10,193.11	9,749.72	10,193.11	—	—	8,299.39	10,399.10
13	San Telmo	60	9,209.76	8,891.74	9,209.76	—	—	6,688.44	9,265.97
14	San Telmo	80	9,174.15	8,048.00	9,468.39	—	—	6,387.80	9,468.39
15	San Telmo	100	—	8,386.57	10,750.29	**9,088.04**	**9,697.52**	7,103.81	10,242.35

Tab. 4.20: Instancias no resueltas por el modelo con pre-cálculo, el modelo directo o el modelo plano

4.8.2. Cantidad de estudiantes

En el experimento anterior decíamos que la complejidad de una instancia incrementaba con la cantidad de posibles paradas. Cuando incrementamos la cantidad de estudiantes y rutas (ya que no podemos agrandar los buses arbitrariamente), sucede algo similar; nuestro modelo tiene que "decidir" entre más rutas para asignar la parada, tiene más estudiantes para asignar y tiene más permutaciones de paradas para considerar. Esperamos que esto también tenga un efecto negativo sobre la performance. Recordemos que si bien la cantidad de variables crece linealmente con la cantidad de estudiantes en todos los modelos, no así con la cantidad de rutas. El modelo con pre-cálculo de distancias y el directo, tiene muchas variables que son replicadas una vez para cada ruta posible. Por otro lado, el modelo plano "aplana" estas variables (de ahí su nombre), por lo cual esperamos que tenga mejor performance en casos con muchas rutas posibles.

Para estos experimentos, utilizaremos nuevamente los tres barrios que separamos (Barracas, La Boca y San Telmo) y variamos la cantidad de estudiantes (200, 300, 400, 500 y 600 con 4, 6, 8, 10 y 12 buses de capacidad 60, respectivamente). Dejamos constante en 25 la cantidad de paradas y en 200m la distancia de caminata máxima.

Podemos ver varios resultados interesantes. En las instancias del barrio de Barracas, el modelo plano resuelve todas las instancias. El directo puede resolver sólo la de 4 rutas y consumiendo casi dos órdenes de magnitud más que el modelo con pre-cálculo de distancias. En esta instancia, este último modelo tiene una mejor performance que el plano por más de un orden de magnitud. En la instancia de 6 rutas, aún sigue esta relación, pero podemos ver como esta diferencia se achica bastante. El modelo con pre-cálculo de distancias no llega a resolver la instancia con 8 rutas, mientras que el modelo plano la resuelve en cerca de media hora. Para el caso de 10 instancias, tienen una performance similar. Finalmente, para el caso de 12 rutas, la diferencia es muy notoria, el modelo plano tarda cerca de 13 veces menos en resolver la instancia. Podemos ver que existe una tendencia, en donde las ventajas del modelo plano se van haciendo más notorias conforme aumentamos la cantidad de rutas.

En las instancias del barrio de La Boca, se puede ver una tendencia similar. El modelo directo no puede resolver instancias más allá de las 4 rutas. El modelo con pre-cálculo de distancia tiene mejor performance en la instancia de 4 rutas, pero el modelo plano comienza a tener mejor performance. Por ejemplo, en la instancia con 8 rutas, el modelo con pre-cálculo de distancias no logra resolver la instancia, mientras que el modelo plano logra resolverla en menos de 40 minutos. Aún en la instancia de 12 rutas, que no es resuelto por ninguna variante, el modelo plano obtiene ambas mejores cota inferior y solución obtenida; naturalmente, también tiene la menor brecha entre ellas.

En las instancias del barrio de San Telmo, se puede ver una tendencia diferente. Ambos el modelo con pre-cálculo de distancias y el modelo directo logran resolver las instancias con 4, 6 y 12 rutas, mientras que el modelo plano no lo hace. Aún en las instancias de 8 y 10 rutas, el modelo directo obtiene la menor brecha, seguido del modelo con pre-cálculo de distancias. Creemos que este comportamiento se debe a la forma en la cual el modelo plano garantiza que se cumplan las restricciones de capacidad de los buses. Este modelo es el único que no acota directamente sobre las variables de asignación, dado que no diferencia por ruta. Sospechamos que esto, en instancias donde los estudiantes pueden ser asignadas a muchas paradas diferentes, puede traer este tipo de comportamiento. Recordemos que la cantidad de paradas y la distancia de caminata máxima son constantes, y el tamaño de San Telmo es mucho menor al de La Boca o Barracas.

Podemos obtener una importante conclusión sobre este experimento. Como era nuestra hipótesis, el modelo plano obtiene una mejor performance relativa cuando se aumenta la cantidad de estudiantes y rutas, con la salvedad de que si los estudiantes tienen muchas paradas posibles de asignación, esto no sucede.

En la página siguiente, podemos ver los resultados de los experimentos realizados en detalle.

id	Barrio	$\|S\|$	$\|P_G\|$	Sol. óptima	T. pre-cálc.	Ticks pre-cálc.	T. directo	Ticks directo	T. plano	Ticks plano
1	Barracas	200	4	21,989.90	**0.98**	**797.50**	45.50	29,503.69	16.72	12,348.40
2	Barracas	300	6	25,162.90	**997.84**	**508,423.72**	—	—	1,863.56	1,519,521.14
3	Barracas	400	8	31,296.03	—	—	—	—	**1,908.66**	**1,796,024.92**
4	Barracas	500	10	32,539.75	**381.78**	**254,819.30**	—	—	479.22	501,481.74
5	Barracas	600	12	38,432.39	1,465.86	601,590.63	—	—	**111.64**	**109,107.21**
6	Boca	200	4	13,406.10	**24.48**	**17,013.58**	3,101.97	1,702,539.49	118.39	106,493.35
7	Boca	300	6	13,899.75	183.77	**130,488.08**	—	—	**176.28**	173,154.28
8	Boca	400	8	15,647.54	—	—	—	—	**694.61**	**685,989.01**
9	Boca	500	10	17,266.56	3,602.47	707,052.07	—	—	**2,328.19**	**1,555,517.53**
11	San Telmo	200	4	8,334.77	**1.98**	**1,676.86**	2.09	1,833.82	—	—
12	San Telmo	300	6	9,804.51	70.44	54,995.43	**54.20**	**47,381.70**	—	—
15	San Telmo	600	12	13,393.96	**1,892.50**	**1,064,324.95**	2,568.80	1,286,502.73	—	—

Tab. 4.21: Instancias resueltas por el modelo con pre-cálculo, el modelo directo o el modelo plano

id	Barrio	$\|S\|$	$\|P_G\|$	Sol. óptima	C.D. pre-cálc.	Sol. pre-cálc.	C.D. directo	Sol. directo	C.D. plano	Sol. plano
2	Barracas	300	6	25,162.90	—	—	23,104.63	25,816.34	—	—
3	Barracas	400	8	31,296.03	27,904.82	31,833.06	25,524.16	32,175.09	—	—
4	Barracas	500	10	32,539.75	—	—	29,306.17	34,222.77	—	—
5	Barracas	600	12	38,432.39	—	—	29,394.68	40,660.82	—	—
7	Boca	300	6	13,899.75	—	—	13,321.31	14,077.32	—	—
8	Boca	400	8	15,647.54	14,999.98	15,647.54	14,203.02	15,965.06	—	—
9	Boca	500	10	17,266.56	—	—	14,675.39	18,032.24	—	—
10	Boca	600	12	—	14,141.76	18,588.10	14,480.23	18,267.39	**17,375.82**	**18,255.43**
11	San Telmo	200	4	8,334.77	—	—	—	—	7,744.62	8,334.77
12	San Telmo	300	6	9,804.51	—	—	—	—	8,771.86	9,804.51
13	San Telmo	400	8	—	11,134.44	**11,854.24**	**11,741.69**	**11,854.24**	9,905.88	**11,854.24**
14	San Telmo	500	10	—	11,746.29	13,425.22	**12,209.35**	13,423.96	9,328.81	**13,401.71**
15	San Telmo	600	12	13,393.96	—	—	—	—	10,885.92	13,393.96

Tab. 4.22: Instancias no resueltas por el modelo con pre-cálculo, el modelo directo o el modelo plano

5. CONCLUSIONES

El problema de ruteo de buses escolares ha sido foco de investigación ya hace varias décadas. Esto se debe en parte al costo variable que significa mantener una flota de buses, lo cual hace su profunda optimización útil.

En esta tesis, presentamos tres modelos que abordan el problema: el modelo con pre-cálculos de distancias, quizás la formulación más "pura", el modelo directo, que espera utilizar la estructura de las calles y esquinas con la esperanza de mejorar el tiempo de cómputo en ciertas instancias, y el modelo plano, que espera ser más eficiente en casos donde la cantidad de rutas haga inmanejable la cantidad de variables para los otros dos modelos.

Luego, proponemos algunas variantes sobre los modelos, intentando eliminar las *lazy constraints*, que impactan negativamente en la performance. Con experimentación, encontramos que estas variantes tienen mérito y las implementamos en los modelos para posterior experimentación.

Después, planteamos una optimización sobre el modelo directo, en la cual acotamos la cantidad de veces que puede pasar un bus por una cuadra, basándonos en los caminos más cortos entre pares de paradas. Experimentamos y vemos que esta optimización sirve bastante, permitiéndonos resolver instancias que no era posible resolver sin ella.

Luego, implementamos varias ideas que aplicamos a todos los modelos. Inicialmente, presentamos una idea que apunta a reducir la simetría de nuestra formulación. Intenta aprovechar la idea de que es indiferente si a estudiante se lo envía a una parada u otra, siempre que los estudiantes compartan el conjunto de paradas posibles de asignación. Experimentamos y vemos que esta técnica tiene una mejora significativa para todos los casos, entonces la adoptamos para nuestros modelos. Luego, creamos una heurística constructiva con posterior búsqueda local con el fin de proveerle al *solver* una solución inicial razonable. Verificamos con experimentación que esta técnica no otorga mejoras significativas, salvo por el caso del modelo plano que se ve beneficiado marginalmente. Decidimos no mantenerla en los otros casos. Por último, implementamos una heurística primal de redondeo en nodos sobre uno de los modelos, que trata de utilizar la solución fraccionaria de la relajación lineal para general una solución entera a nuestro modelo, de forma de ir mejorando la solución incumbente. Vemos que el impacto de esta heurística es negativo, por lo cual la descartamos.

Posteriormente, realizamos una experimentación donde comparamos los diferentes modelos con todas las optimizaciones que tuvieron impacto positivo. Vemos que cuando la cantidad de paradas es grande y el barrio no es tan grande, el modelo directo puede funcionar mejor que una solución más convencional como es el modelo con pre-cálculo de distancias. También, encontramos que el modelo plano puede ser una buena alternativa cuando se desea rutear una cantidad grande de buses, siempre y cuando se hagan algunas salvedades.

5.1. Trabajo futuro

Creemos que este problema, si bien ha recibido creciente atención, sobre todo en la última década, puede ser foco de bastante investigación futura. Algunas de los posibles

caminos que vemos son los siguientes:

- En este trabajo, no exploramos planos de corte. Creemos que hay mucho potencial en esta dirección. Dada la similitud de este problema con el problema de viajante de comercio o el problema de ruteo de vehículos, muchas de las ideas sobre estos problemas ampliamente conocidos pueden ser aplicadas al problema de esta tesis.
- Una optimización que presentamos en el trabajo es la reducción de simetría en base a la clusterización de estudiantes y su asignación a las paradas en conjunto. Creemos que este tipo de ideas pueden llevar a una mejor formulación matemática, que lleven a una formulación con menor grado de simetría. Un ejemplo de eso es el siguiente: la asignación de un estudiante a una parada en sí no es importante. Sí lo es respetar las capacidades de los buses. De esta forma, podríamos asignar los estudiantes directamente a las rutas, y no a las paradas. Podríamos restringir que para que un estudiante esté asignado a una ruta, que las paradas de la ruta y las posibles paradas para el estudiante tengan intersección no vacía.
- Creemos también que una combinación de los modelos directo y plano podría funcionar mejor en algunos casos como los casos de alta cantidad de rutas pero muchas paradas posibles para cada estudiante. Estos son las instancias donde la ventaja del modelo plano para las instancias con muchas rutas no se ve más.
- Creemos también que el problema es un muy buen candidato para la implementación de heurísticas clásicas y metaheurísticas, sean para proveer una solución inicial a un *solver* o como soluciones independientes del problema.
- Se puede investigar también la posibilidad de utilizar la técnica de generación de columnas, creando una variable de decisión por cada posible ruta para cada bus.
- Por último, en este trabajo implementamos una muy básica heurística primal por redondeo que no tuvo buenos resultados. Sin embargo, creemos que la técnica no debe ser desechada y que puede ser posible encontrar parámetros que obtengan mejores resultados, tales como cambiar la frecuencia con la cual es llamada la heurística, implementar las llamadas en un lenguaje con mejor performance o implementar técnicas completamente diferentes.

Apéndice

A. MARCO TEÓRICO

A.1. Complejidad computacional

La complejidad computacional de un algoritmo es una medida de la cantidad de recursos requeridos para correrlo. Esta medida puede referirse a recursos de diferente naturaleza. Sin embargo, en general se refieren a la cantidad de operaciones elementales necesarias para resolver el problema o el espacio utilizado en memoria. En este trabajo, a menos que esté aclarado, estamos hablando de la cantidad de operaciones elementales necesarias para correr el algoritmo. Dicha medida suele tener una relación proporcional con el tiempo necesario para correr el algoritmo en una misma computadora.

Esta medida generalmente se expresa como una cota superior asintótica utilizando la notación de Landau. Informalmente, si una función $f \in \mathcal{O}(g(x))$, la función está acotada superiormente por $k \times g(x)$ con k constante y x tendiendo al infinito. Cuando decimos que un algoritmo corre en tiempo lineal, cuadrático, polinomial o exponencial, hablamos sobre la naturaleza de la función g.

Problema de decisión Se denomina problema de decisión a un problema que tiene una respuesta de sí o no. Ejemplos de estos problemas son decidir si un número es par, si un número es primo o si un número divide a otro. En general, las clases de complejidad computacional están definidas sobre los problemas de decisión.

P Un problema de decisión pertenece a la clase de complejidad computacional P si existe un algoritmo que puede resolver el problema en tiempo polinomial en una computadora convencional. Ejemplos de problemas en P son verificar si un número es primo y las versiones de decisión de encontrar el camino mínimo entre dos vértices de un grafo y encontrar el máximo divisor común de dos números.

NP Un problema de decisión pertenece a la clase de complejidad computacional NP si para instancias donde la respuesta del problema es positiva y la solución nos es dada, la prueba puede realizarse en tiempo polinomial. Tomemos como ejemplo la versión de decisión de camino mínimo (encontrar si entre dos vértices de un grafo existe un camino de largo menor o igual a L). Si existe dicho camino y este nos es dado, podemos comprobar de forma polinomial la validez de la solución. Algunos ejemplos de problemas en NP son las versiones de decisión de camino mínimo y el problema del viajante de comercio. Es trivial demostrar que $P \subset NP$. Sin embargo, el problema de si $P = NP$ es uno de los grandes problemas abiertos de la teoría de la computación hace años.

NP-difícil Un problema de decisión X pertenece a la clase de complejidad computacional NP-difícil si para todo problema $Y \in NP$ existe una reducción factible en tiempo polinomial que reduce Y a X. Esto es, para cada instancia de Y, puedo transformarla en tiempo polinomial a una instancia de X. Informalmente, esto significa que X es por lo menos tan difícil como Y.

NP-completo Un problema de decisión pertenece a la clase de complejidad computacional NP-completo si pertenece a las clases NP y NP-difícil. Estos problemas se conocen informalmente como los más difíciles dentro de la clase NP.

A.2. Programación lineal

Un problema de programación lineal (PL) está definido como un modelo matemático donde, dado un conjunto de variables $x \in \mathbb{R}^n$, se busca optimizar una función objetivo lineal $c : \mathbb{R}^n \longrightarrow \mathbb{R}$, restringida con inecuaciones no estrictas también lineales. Canónicamente, los problemas de programación lineal se pueden expresar de la siguiente forma:

$$\begin{aligned} \text{maximizar} \quad & c^T x \\ \text{sujeto a} \quad & Ax \leq b \end{aligned}$$

donde x representa al vector de variables a optimizar, $c \in \mathbb{R}^n$ y $b \in \mathbb{R}^m$ son vectores dados, $A \in \mathbb{R}^{m \times n}$ es una matriz dada. Es importante ver que cualquier problema de programación lineal se puede expresar de la forma anterior; si se quisiera minimizar, se podría reemplazar c por su inverso y si alguna restricción de A se quisiera expresar como una combinación lineal de x mayor o igual a una constante b_i, se podrían reemplazar la fila de A y b por sus inversos e invertir el signo, de forma de quedar consistente con la versión canónica.

Está demostrado que existen algoritmos que resuelven este problema en tiempo polinomial, es decir, el problema está en P. Un ejemplo es el algoritmo de Karmarkar presentado por el autor homónimo [23]. Sin embargo, en la práctica se suele utilizar el algoritmo símplex propuesto por Dantzig [18]. Este método, aún requiriendo tiempo de orden exponencial en el peor caso, suele funcionar mejor en la práctica, tomando ciertas precauciones.

Es importante ver que existe una biyección entre cada conjunto de restricciones, tuplas de la forma $(A, b) \in (\mathbb{R}^{m \times n}, \mathbb{R}^m)$ y poliedros convexos $P \subset \mathbb{R}^n$. Dichos poliedros se denominan regiones factibles de los problemas asociados. Un resultado importante de la investigación sobre el tema es que el óptimo de la función objetiva siempre es uno de los vértices del poliedro. Tomemos como ejemplo el siguiente problema:

$$\begin{aligned} \text{maximizar} \quad & \frac{9}{4}x_1 + 3x_2 \\ \text{sujeto a} \quad & -\frac{7}{2}x_1 + x_2 \leq 0 \\ & -\frac{1}{2}x_1 + x_2 \leq \frac{5}{2} \\ & \frac{1}{3}x_1 + x_2 \leq 4 \\ & x_1 \leq \frac{7}{2} \\ & \frac{1}{2}x_1 - x_2 \leq 0 \end{aligned}$$

En la figura A.1 se puede ver la región factible en $\mathbb{R}^2$ del problema. También se pueden ver los vértices marcados en azul y el óptimo en rojo.

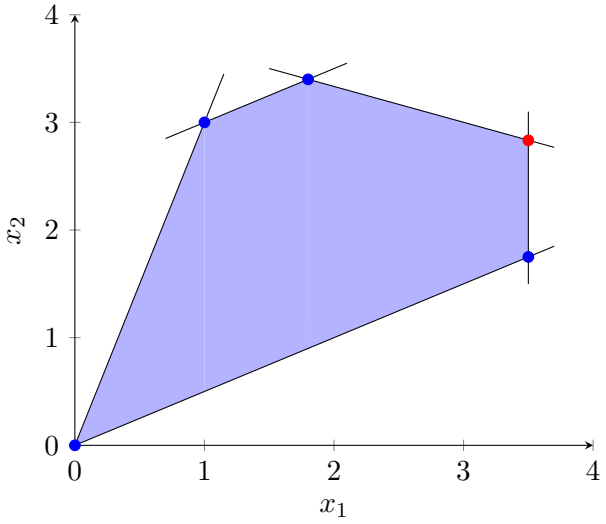

Fig. A.1: Región factible del problema de programación lineal

A.3. Programación lineal entera

Análogamente, podemos definir un problema de programación lineal entera (PLE) como un problema de programación lineal en el cual requerimos que las variables de la solución sean enteras. Canónicamente, podemos expresar el problema de la siguiente forma:

$$\begin{aligned} \text{maximizar} \quad & c^T x \\ \text{sujeto a} \quad & Ax \leq b \\ & x \in \mathbb{Z}^n \end{aligned}$$

Para este problema, si bien muy similar al expuesto en la sección anterior, no se ha encontrado un algoritmo capaz de resolverlo en tiempo polinomial. Si $P \neq NP$, dicho algoritmo no existiría. El problema pertenece a la clase de complejidad NP-completo.

Como este problema tiene restricciones que no son una inecuación lineal, la región factible de nuestro problema ya no se trata de un poliedro convexo, sino de un conjunto de puntos en el espacio. Sin embargo, si nosotros pudiéramos encontrar el polígono convexo más pequeño que contuviera a todos los puntos de la región factible, podríamos usar el algoritmo de programación lineal común para encontrar nuestro óptimo, ya que este se encontraría en un vértice de dicho polígono. Este polígono se llama cápsula convexa. Tomando como base el problema ejemplificado en la sección previa y forzando la integralidad de ambas variables, tenemos un problema de programación lineal entera cuya cápsula convexa se puede ver en la figura A.2.

Una variante de problema común son los modelos de programación lineal entera mixta. En este tipo de problemas, las restricciones de integralidad no están aplicadas sobre todas las variables, sino sobre un subconjunto propio no vacío.

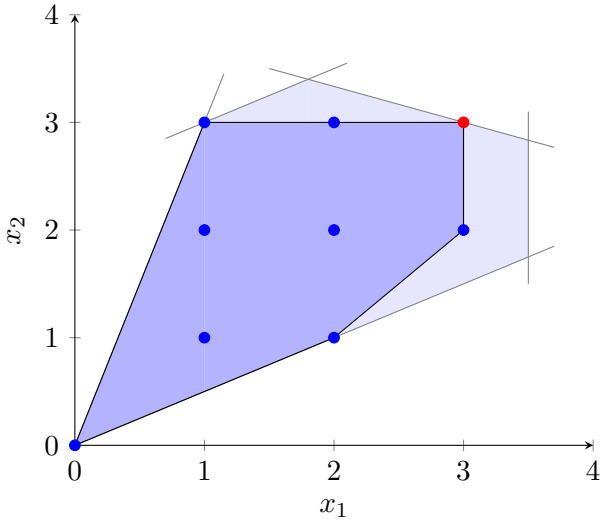

Fig. A.2: Región factible y cápsula convexa del problema de programación lineal entera

A.4. *Branch & Bound*

Como vimos en la sección anterior, el problema de encontrar el óptimo en un modelo PLE reside en que los algoritmos que tenemos pueden recorrer los vértices de nuestro poliedro. Dadas las restricciones de integralidad, el óptimo puede que no se encuentre en la frontera del poliedro. La técnica de *branch and bound* aborda este problema aprovechando que tenemos un algoritmo que resuelve en tiempo razonable la relajación lineal de nuestro modelo. Se compone de dos técnicas: *branching* y *bounding*, ramificar y acotar, respectivamente, en español.

La técnica de *branching* consiste en, dado un modelo PLE, resolver la relajación lineal, obteniendo como resultado los coeficientes x^*. Si esta solución tiene todos sus coeficientes enteros, esta es la solución óptima al modelo PLE, se retornan los valores y finaliza la ejecución. Si no los tiene, se escoge un coeficiente fraccionario x_i y se generan dos nuevos modelos PLE. Ambos contienen las restricciones y función objetivo de nuestro modelo original, pero a uno se le agrega la restricción $x_i \leq \lfloor x_i^* \rfloor$ y al otro se le agrega $x_i \geq \lceil x_i^* \rceil$. De esta forma, ninguno de los dos subproblemas contiene en su relajación lineal a x^*. Estos dos modelos se resuelven recursivamente, y cuando devuelven los resultados, se devuelve el mejor de los dos.

Si bien la técnica anterior alcanza para resolver el problema, tiene que recorrer innecesariamente todo el espacio de búsqueda para resolver el problema. La técnica de *bounding* intenta prevenir esto. Esta técnica consiste en mantener la mejor solución entera ya encontrada durante el cómputo. Si la relajación lineal de algún nodo presenta un óptimo peor que el de la mejor solución entera ya encontrada, en las ramificaciones de dicho nodo, no vamos a encontrar una mejor solución. Por lo tanto, no vale la pena seguir desarrollando estas ramas, por lo que estas son podadas. Esta técnica se ve muchas veces combinada con una heurística inicial para proveer una cota inicial razonable.

Bibliografía

[1] R. M. Newton and W. H. Thomas, "Design of school bus routes by computer," *Socio-Economic Planning Sciences*, vol. 3, no. 1, pp. 75 – 85, 1969.

[2] J. Park and B.-I. Kim, "The school bus routing problem: A review," *European Journal of Operational Research*, vol. 202, no. 2, pp. 311 – 319, 2010.

[3] W. A. Ellegood, S. Solomon, J. North, and J. F. Campbell, "School bus routing problem: Contemporary trends and research directions," *Omega*, 2019.

[4] R. D. Angel, W. L. Caudle, R. Noonan, and A. Whinston, "Computer-assisted school bus scheduling," *Management Science*, vol. 18, pp. B–279–B–288, Feb 1972.

[5] R. M. Newton and W. H. Thomas, "Bus routing in a multi-school system," *Computers & Operations Research*, vol. 1, p. 213–222, Aug 1974.

[6] B. T. Bennett and D. C. Gazis, "School bus routing by computer," *Transportation Research*, vol. 6, 1972.

[7] B. Gavish and E. Shlifer, "An approach for solving a class of transportation scheduling problems," *European Journal of Operational Research*, vol. 3, p. 122–134, Mar 1979.

[8] L. D. Bodin and L. Berman, "Routing and scheduling of school buses by computer," *Transportation Science*, vol. 13, no. 2, pp. 113–129, 1979.

[9] G. Dulac, J. A. Ferland, and P. A. Forgues, "School bus routes generator in urban surroundings," *Computers & Operations Research*, vol. 7, p. 199–213, Jan 1980.

[10] L. Chapleau, J.-A. Ferland, and J.-M. Rousseau, "Clustering for rom:lngin densely populated areas," p. 10, 1985.

[11] R. Bowerman, B. Hall, and P. Calamai, "A multi-objective optimization approach to urban school bus routing: Formulation and solution method," *Transportation Research Part A: Policy and Practice*, vol. 29, p. 107–123, Mar 1995.

[12] P. Schittekat, M. Sevaux, and K. Sorensen, "A mathematical formulation for a school bus routing problem," in *2006 International Conference on Service Systems and Service Management*, vol. 2, pp. 1552–1557, Oct 2006.

[13] P. Schittekat, J. Kinable, K. Sörensen, M. Sevaux, F. Spieksma, and J. Springael, "A metaheuristic for the school bus routing problem with bus stop selection," *European Journal of Operational Research*, vol. 229, no. 2, pp. 518 – 528, 2013.

[14] L. M. Martínez and J. M. Viegas, "Design and deployment of an innovative school bus service in lisbon," *Procedia - Social and Behavioral Sciences*, vol. 20, p. 120–130, 2011.

[15] J. Riera-Ledesma and J.-J. Salazar-González, "Solving school bus routing using the multiple vehicle traveling purchaser problem: A branch-and-cut approach," *Computers & Operations Research*, vol. 39, no. 2, pp. 391 – 404, 2012.

[16] J. Riera-Ledesma and J. J. Salazar-González, "A column generation approach for a school bus routing problem with resource constraints," *Computers Operations Research*, vol. 40, no. 2, pp. 566 – 583, 2013.

[17] J. Kinable, F. Spieksma, and G. Vanden Berghe, "School bus routing—a column generation approach," *International Transactions in Operational Research*, vol. 21, no. 3, pp. 453–478, 2014.

[18] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms, Third Edition*. The MIT Press, 3rd ed., 2009.

[19] C. E. Miller, A. W. Tucker, and R. A. Zemlin, "Integer programming formulation of traveling salesman problems," *J. ACM*, vol. 7, pp. 326–329, Oct. 1960.

[20] G. Clarke and J. W. Wright, "Scheduling of vehicles from a central depot to a number of delivery points," *Operations Research*, vol. 12, no. 4, pp. 568–581, 1964.

[21] Ministerio de Educación e Innovación. Secretaría de Ciencia, Tecnología e Innovación. SS de Ciudad Inteligente. DG Ciencias de la Información. Unidad de Sistemas de Información Geográfica (USIG), "Calles." https://data.buenosaires.gob.ar/dataset/calles. Accedido: 2019-05-03.

[22] Open Geospatial Consortium Inc., "OpenGIS® Implementation Standard for Geographic information - Simple feature access." http://www.opengeospatial.org/standards/sfa.

[23] N. Karmarkar, "A new polynomial-time algorithm for linear programming," *Combinatorica*, vol. 4, pp. 373–395, Dec 1984.