ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE CIENCIAS

ALGORITMO DE GENERACIÓN DE COLUMNAS PARA LA ASIGNACIÓN DE TAREAS EN EL SISTEMA METROBUS-Q.

PROYECTO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO MATEMÁTICO

ELIZABETH CAROLINA ZÚÑIGA RIOFRÍO

elizabeth_zuniga@hotmail.com

DIRECTOR: LUIS MIGUEL TORRES CARVAJAL, PhD.

luis.torres@epn.edu.ec

Quito, Agosto 2015

DECLARACIÓN

Yo, Elizabeth Carolina Zúñiga Riofrío, declaro bajo juramento que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

La Escuela Politécnica Nacional puede hacer uso de los derechos correspondientes a este trabajo, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normatividad institucional vigente.

ELIZABETH CAROLINA ZÚÑIGA RIOFRÍO

CERTIFICACIÓN

Certifico que el presente trabajo fue desarrollado por Elizabeth Carolina Zúñiga Riofrío, bajo mi supervisión.

Luis Miguel Torres Carvajal, PhD.
DIRECTOR DE PROYECTO

AGRADECIMIENTOS

Agradezco a Dios, por darme la fortaleza para alcanzar este logro.

Gracias al Dr. Luis Miguel Torres, quien con su acertada dirección y apoyo, me ha permitido presentar este proyecto.

Gracias al Dr. Ramiro Torres, Miguel Flores y Raúl Pineda, quienes me ayudaron en el desarrollo de este trabajo.

Gracias a la Facultad de Ciencias, por inculcarme todos los conocimientos que me servirán en mi vida profesional.

DEDICATORIA

A mis padres Rita y Juan, por su amor, sacrificio, entrega y ejemplo que me han brindado a lo largo de mi vida académica. Les dedico esta tesis, en la cual está plasmado no simplemente mi esfuerzo sino también el de ustedes.

A mis hermanas Francisca, Janneth y Nathaly, las mejores hermanas que la vida me ha podido dar, por su cariño y apoyo.

Índice general

Índice de figuras **3**

Índice de cuadros **4**

1. Introducción **1**

1.1. Movilidad dentro del Distrito Metropolitano de Quito 2

1.1.1. Sistema de Transporte Metrobús-Q 4

1.1.2. Corredor Central Trolebús 5

2. Marco Teórico **10**

2.1. Modelo lineal entero para el problema de asignación de tareas 13

2.1.1. Definición del problema y formulación matemática 14

2.1.2. Modelo de Particionamiento de Conjuntos 15

2.1.3. Generación de Columnas en Particionamiento de Conjuntos . 17

2.1.4. Heurística DS-OPT para la Planificación de Tareas 18

2.1.4.1. Subrutina cgen 20

2.1.4.1.1. Subrutina price 21

2.1.4.2. Subrutina aug 21

2.2. Método de Generación de Columnas 23

2.2.1. Problema de Corte de Material 25

2.2.2. Un modelo de generación de columnas 25

2.2.2.1. El Problema Maestro 26

2.2.2.2. Subproblema 27

2.3. El Problema de Camino más Corto con Restricciones de Recursos(SPPRC) 29

2.3.1. Algoritmos para resolver SPPRC 31

2.3.1.1. Algoritmo de corrección de etiquetas 32
2.3.1.2. Algoritmo de fijación de etiquetas 34
2.3.1.3. Algoritmo de arrastre 36

3. Asignación de Tareas en el Sistema Metrobús-Q 38
3.1. Planteamiento del Problema 38
3.2. Formulación del Modelo 42
3.3. Modelo de Generación de Columnas 43
3.4. Generación de nuevas jornadas 47
3.4.1. Algoritmo de Fijación de Etiquetas 53
3.5. Asignando jornadas a rutas 59

4. Implementación del Problema de Asignación de Tareas en el Trolebús 62
4.1. Pseudocódigo del Problema de Asignación de Tareas 64
4.1.1. Instancia inicial de RMLP-MBQ 66
4.1.2. El problema de costeo 66
4.1.2.1. Problema de camino de más corto con dos recursos 67
4.1.3. Fijación de Variables 70
4.2. Estructura de Datos 70
4.2.1. Clase Viajes 71
4.2.2. Clase Arco 72
4.2.3. Clase Nodo 72
4.2.4. Clase Etiqueta 73
4.2.5. Clase Duty 74
4.2.6. Clase Score 75
4.2.7. Clase Jornadas 77
4.2.8. Clase Problema MIP 79

5. Resultados Computacionales 82

6. Conclusiones 95

Bibliografía 99

Índice de figuras

1.1. Evolución del parque vehicular en el DMQ. 2
1.2. Principales motivos de los viajes en el DMQ. 3
1.3. Sistema de Transporte Metrobús-Q. 4
1.4. Esquema de las terminales del Trolebús. 7

3.1. Formulación del problema de costeo. 50
3.2. Instancia del problema de caminos más cortos con restricciones de recursos. 54
3.3. Primera iteración del algoritmo de fijación de etiquetas. 54
3.4. Segunda iteración del algoritmo de fijación de etiquetas. 55
3.5. Tercera iteración del algoritmo de fijación de etiquetas. 55
3.6. Iteración final del algoritmo sin considerar la etiqueta dominada. . . . 56
3.7. Iteración final del algoritmo considerando la etiqueta dominada. 56
3.8. Red tomando al nodo 1 como inicio de la jornada laboral. 57
3.9. Red tomando al nodo 2 como inicio de la jornada laboral. 58
3.10. Iteración final del algoritmo tomando como viaje inicial el nodo 1. . . . 58
3.11. Iteración final del algoritmo tomando como viaje inicial el nodo 2. . . . 58

4.1. Tiempo de ejecución de diferentes solvers 63

5.1. Número de viajes vs Tiempos de ejecución. 90
5.2. Número de viajes vs Tiempo de ejecución 92
5.3. Número de viajes vs Tiempo de ejecución 92
5.4. Número de viajes vs Tiempo de ejecución 93
5.5. Número de viajes vs Tiempo de ejecución 93
5.6. Número de viajes vs Tiempo de ejecución 94

Índice de cuadros

1.1. Comportamiento de la demanda en el subsistema Metrobús-Q. 5
1.2. Circuitos que operan en el Corredor Central Trolebús. 6

3.1. Circuitos del Trolebús modificados. 39
3.2. Características de los viajes. 53

5.1. Características de las instancias de prueba. 83
5.2. Parámetros de las pruebas computacionales. 84
5.3. Resultados del modelo para la instancia Lunes-Jueves. 86
5.4. Resultados del modelo para la instancia Viernes. 87
5.5. Resultados del modelo para la instancia Sábado. 88
5.6. Resultados del modelo para la instancia Domingo. 89
5.7. Mejores soluciones obtenidas por nuestro algoritmo en cada instancia. 91
5.8. Descripción detallada de los mejores resultados. 91

Índice de algoritmos

2.1. Heurística dsopt para la planificación de tareas. 19
2.2. Algoritmo de correción de etiquetas. 33
2.3. Algoritmo de fijación de etiquetas. 35
2.4. Algoritmo de arrastre. 36

4.1. Algoritmo de Asignación de Tareas en el Sistema Trolebús. 64

Resumen

En el presente proyecto se aborda el problema de asignación de conductores, en el contexto del sistema de transporte público masivo Trolebús. Se presenta un modelo de programación lineal entero, basado en un modelo de particionamiento de conjuntos. Este modelo tiene por objetivo reducir al mínimo el tiempo de inactividad total de los conductores en las terminales, respetando los acuerdos laborales y buscando ocupar el menor número de buses. Se propone el uso del método de generación de columnas para enfrentar el problema. Como subproblema resultante, se obtienen problemas de caminos más cortos con restricciones, uno para cada tipo de tarea. Debido al gran número de variables que se presentan en la asignación de conductores, se hace uso de la estrategia de fijación de variables. Por último, se formula un modelo de emparejamiento máximo para unir las jornadas laborales que se puedan realizar en una misma secuencia, en un día. Se reportan los resultados computacionales, para instancias reales, que cubren hasta 1400 viajes programados.

Palabras claves: Optimización del transporte público, problema de asignación de conductores, problema de asignación de vehículos, programación entera, problema de camino más corto con restricciones.

Abstract

In this project, the duty scheduling problem will be addressed in the context of the Trolebús mass transit system. An integer linear programming model based on a set partitioning formulation is presented. This model aims at minimizing the total idle time of drivers in the terminals, while complying to labor agreements and minimizing the number of buses used in the solution. A column generation method is proposed to address the problem. The solution scheme involves solving shortest paths problems with resource constraints, one different subproblem for each type of task path. Due to the large number of variables, we make use of a variable fixing strategy. Finally, a maximum matching model is formulated to join duties that can be done in one block, in a day. Computational results for real instances, covering up to 1400 scheduled trips, are reported.

Key words: Public transport optimization, duty scheduling problem, vehicle scheduling problem, integer programming, shortest path problem with resource constraints.

Capítulo 1

Introducción

En las últimas décadas el rostro urbano de la mayoría de ciudades del mundo ha experimentado grandes cambios. El constante crecimiento vehicular, la congestión en las redes viales y el crecimiento poblacional, han generado gran impacto en el medioambiente, la calidad de vida de las personas, la productividad, la calidad del transporte e incluso en la seguridad. El adecuado funcionamiento de la movilidad urbana es de gran importancia para el desarrollo de las actividades de la población, por lo cual los gobiernos siempre se han preocupado por mejorar la calidad del transporte urbano, enfocando muchas veces sus esfuerzos en el aumento de la capacidad de la red vial; sin embargo, esto conlleva una gran inversión y en ocasiones la topología de las ciudades no permite grandes avances en este sentido. Tomando en cuenta que los efectos de aumentar la capacidad de la red vial no serán sostenibles si el parque automotor sigue creciendo al ritmo actual y considerando que, especialmente en ciudades en vías de desarrollo, este crecimiento no va de la mano con el aumento del presupuesto para el desarrollo y mantenimiento de infraestructura, se hace necesario encontrar otras formas factibles de enfrentar este problema, como lo es el uso eficiente de los recursos de transportación disponibles, mediante el cubrimiento de la demanda usando menos energía, en el menor tiempo y al menor costo.

La ciudad de Quito es la capital de la República del Ecuador, se encuentra ubicada al sur de la línea equinoccial, en las faldas del cerro Atacazo, del volcán Guagua Pichincha y del Rucu Pichincha. La forma de Quito es alargada, tiene

aproximadamente 20 km de largo y 4 km de ancho, la cual la hace una ciudad con gran dificultad para el adecuado desarrollo de la infraestructura vial. Además enfrenta el problema de crecimiento vehicular, evidenciado en el aumento del número de vehículos en circulación a lo largo de los años. La Figura 1.1 muestra la evolución del parque vehicular en Quito, según estimaciones de la revisión técnica vehicular y la Gerencia de Planificación de la Movilidad, obtenida del Plan de Desarrollo del Municipio de Quito.

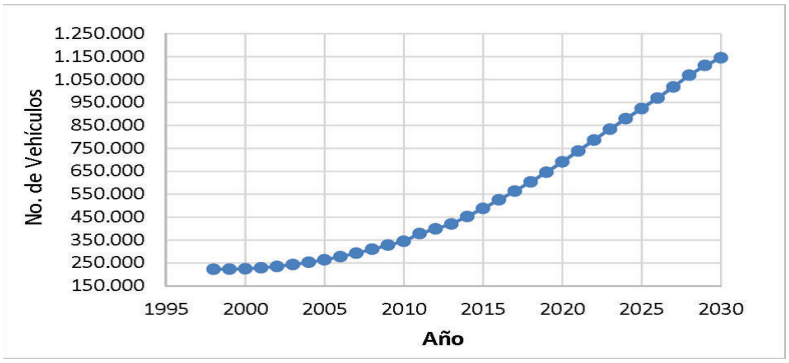

Figura 1.1: Evolución del parque vehicular en el DMQ.

1.1. Movilidad dentro del Distrito Metropolitano de Quito

La movilidad urbana de personas dentro del Distrito Metropolitano de Quito (DMQ) se desarrolla a través de diversos sistemas de transportes: mecanizados (público, comercial y privado) y no mecanizados (peatones y bicicletas). De acuerdo a la Empresa Metropolitana de Transporte de Pasajeros [2012a] el 84 % de esta movilidad se la realiza en transporte mecanizado y el 16 % en transporte no mecanizado. La movilidad se concentra especialmente en el hipercentro (zona comprendida entre el sector de la Y hasta la Villaflora) ya sea como lugar de origen o de destino con aproximadamente el 50 % del total de viajes del transporte público. En un día laborable normal se pueden evidenciar tres picos en la movilidad del DMQ, los cuales se producen entre 5 a 7 de la mañana pues es la hora donde las personas se movilizan para ir a sus lugares de trabajo o de estudio principalmente,

de 12 a 2 pues es la hora en la cual salen los estudiantes de los colegios y escuelas, además de ser la hora en la que las personas tienen su respectivo descanso para almorzar y el último se produce entre las 5 y 7 de la tarde, cuando la mayor parte de personas concluyen sus jornadas laborales y se trasladan a sus respectivos hogares.

En el Plan de Desarrollo 2012-2022 del Municipio del Distrito Metropolitano de Quito [2011] se presentan los principales motivos de los viajes que se realizan en el Distrito Metropolitano de Quito, en un día laboral. En la Figura 1.2 se muestra que el motivo principal es el estudio con un 32,5 %, seguido por el trabajo en un 31,1 % y en tercer lugar se encuentran los viajes por asuntos personales con un 24,3 %.

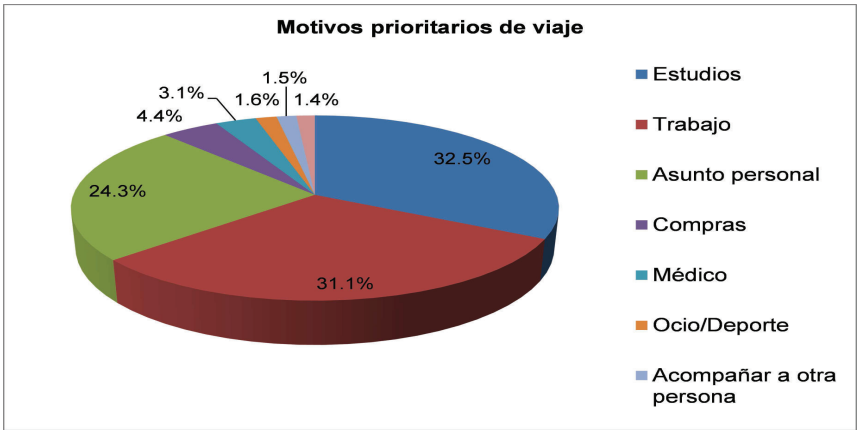

Figura 1.2: Principales motivos de los viajes en el DMQ. **Fuente:**EDM11.

De acuerdo a la Empresa Metropolitana de Transporte de Pasajeros [2012b], el Sistema Metropolitano de Transporte Público de Pasajeros – SMTP del DMQ se compone principalmente de tres subsistemas, que en el año 2012 satisficieron cerca del 73 % de viajes que se realizaron dentro del DMQ. Estos subsistemas son:

Subsistema de Transporte Metrobús-Q El Metrobús-Q se compone de corredores de transporte, urbanos y regionales, operando con buses bajo el concepto BRT (Autobuses de Tránsito Rápido). Estos buses permiten la movilidad masiva de personas, combinando el rendimiento y la calidad de un tren ligero con la flexibilidad, el costo y la simplicidad de un sistema de autobuses.

Subsistema de Transporte Convencional Está constituido por las líneas de

transporte urbanas, intra-parroquiales e inter-parroquiales, opera con buses convencionales o tipo (sin plataforma). Este subsistema brinda el servicio a los sectores localizados en el área urbana, periferia y valles aledaños a la ciudad de Quito.

Subsistema de Transporte Metro de Quito La Municipalidad del DMQ, inició en el año 2012 la construcción de la primera línea del Metro, con el objetivo de mejorar la ampliación de la cobertura y la calidad del servicio del transporte público. Este subsistema será el eje principal que integre los Subsistemas Metrobús-Q y Convencional.

1.1.1. Sistema de Transporte Metrobús-Q

El Sistema de Transporte Metrobús-Q fue concebido bajo el concepto BRT (Bus Rapid Transit), está conformado por corredores de transporte masivo y por líneas alimentadoras. Los corredores de transporte masivo de pasajeros se movilizan principalmente a través de líneas troncales longitudinales que van de Norte a Sur, estos corredores operan en carriles exclusivos y poseen un sistema de paradas y estaciones que permiten a los usuarios integrarse entre diferentes servicios de transporte. Este sistema busca el mejoramiento, la integración y la modernización del servicio de transporte público en el DMQ ofreciendo un servicio de calidad, con seguridad y al menor costo posible para los usuarios.

Figura 1.3: Sistema de Transporte Metrobús-Q.

El sistema Metrobús-Q opera bajo tres infraestructuras: la infraestructura fija que la conforman la red vial, las paradas, las estaciones, la señalización, la

semaforización, entre otras; la infraestructura rodante, constituida principalmente por los buses articulados; y la infraestructura tecnológica, que está conformada principalmente por el sistema de telecomunicación, el sistema de planificación operacional y le sistema de información al usuario.

Según Barrera [2013], actualmente, un poco más de 1 de cada 3 viajes que se realizan en bus es atendido en los corredores del sistema Metrobús-Q, constituido por el Trolebús, la Ecovía, el Corredor Suroriental, el Corredor Central Norte y el Corredor Suroccidental. En efecto, de los 3,6 millones de pasajeros diarios, el Metrobús-Q sirve a 1,2 millones, que se benefician de la integración tarifaria, los horarios extendidos y los ahorros de tiempo que caracterizan a este sistema.

El comportamiento de la demanda en el Sistema Metrobús-Q se evidencia en el siguiente cuadro:

Corredor	2010	2011	2012	junio de 2013
Trolebús	236.504	220.862	227.046	228.213
Ecovía	98.385	117.698	146.474	145.676
Central Norte	199.139	204.715	210.447	216.339
Sur Oriental	25.995	45.774	64.757	76.580
Sur Occidental			198.384	217.462
Total viajes día	560.004	589.049	847.708	884.270

Tabla 1.1: Comportamiento de la demanda de viajes en un día tipo en el subsistema Metrobús-Q (2010-junio 2013).

En la Tabla 1.1 se evidencia que dentro del Subsistema Metrobús-Q, el Corredor Central Trolebús es el medio de transporte en el que se moviliza la mayor parte de la población de Quito, cerca de 230.000 pasajeros en un día típico, por lo cual hemos elegido este medio de transporte para realizar el presente estudio.

1.1.2. Corredor Central Trolebús

El Corredor Central Trolebús es el primer corredor del Sistema Metrobús-Q, el cual inició sus operaciones en el año 1995. Está formado por autobuses de tránsito

rápido (BRT), que son corredores de transporte masivo de pasajeros que operan con energía eléctrica y diesel. La infraestructura fija del Trolebús es la Línea Troncal, la cual es una línea exclusiva que va desde la terminal Quitumbe hasta la terminal Norte, con una distancia de 18km. La línea troncal posee 53 puntos de parada considerando las estaciones de transferencia (puntos en los que los pasajeros pueden conectarse con otros corredores del Metrobús-Q), las terminales (puntos en los cuales los pasajeros pueden tomar alimentadores del Trolebús para dirigirse a las zonas periféricas de la ciudad) y las paradas tipo (puntos de embarque y desembarque de pasajeros que no son ni estaciones de transferencia ni terminales). Las paradas tipo tienen un único sentido, es decir, hay un punto de parada en el sentido norte a sur y otro en el sentido contrario.

De acuerdo a la Empresa Metropolitana de Transporte de Pasajeros [2012b], se dispone de una flota de 113 trolebuses con tres puertas derechas, con capacidad de 180 pasajeros. Estos buses son propiedad de la Municipalidad. Operan en varias líneas (circuitos) en diferentes horarios, interactuando entre sí y con otros corredores de transporte a través de las diferentes estaciones de transferencia y terminales. Los trolebuses circulan a una velocidad media operacional de 18 km/h.

En la página oficial del Trolebús se pueden observar los circuitos bajo los cuales opera el Trolebús, los mismos que son los siguientes:

Código	Origen	Destino
C1	El Recreo	La Y
C2	La Y	Morán Valverde
C2-Q	Quitumbe	La Y
C3	Santo Domingo	La Y
C4	Quitumbe	El Ejido
C5	Quitumbe	Colón
CQ-R	Quitumbe	El Recreo
Expreso Escolar	El Recreo	Colón

Tabla 1.2: Circuitos que operan en la línea troncal - Corredor Central Trolebús.

En la Figura 1.4 se presenta un esquema de la ubicación de los orígenes y destinos de los circuitos presentes en el Trolebús. El Trolebús posee tres terminales (en color verde), las mismas que sirven de depósitos de los autobuses, además que son los únicos puntos de relevo donde puede haber un intercambio de conductores. Estas terminales son: terminal Quitumbe (T-Q), terminal Recreo (T-S) y terminal la Y (T-N). Los puntos de color rosado representan a paradas tipo en las cuales los buses del Trolebús dan la vuelta para completar sus circuitos.

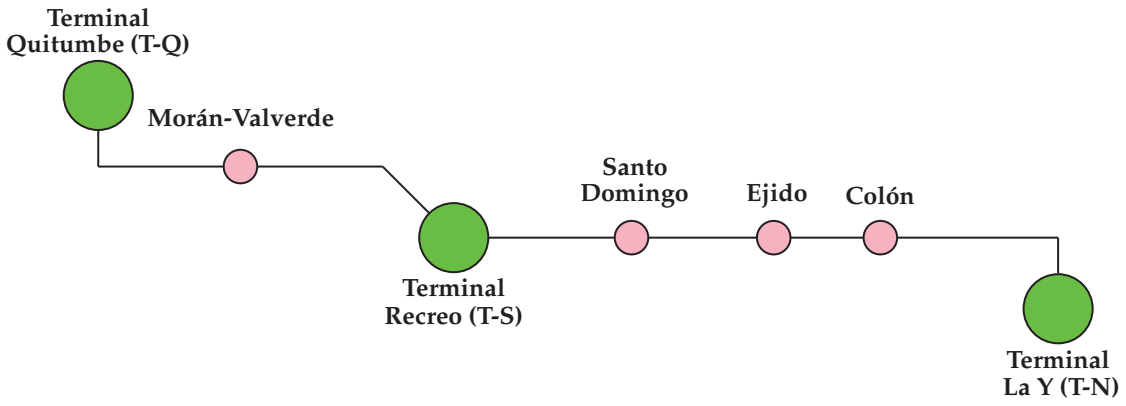

Figura 1.4: Esquema de las terminales del Trolebús.

El Trolebús cuenta con una flota heterogénea. Opera bajo dos tipos de flota: flota antigua y flota nueva. Las unidades de flota antigua debido a sus características técnicas pueden operar únicamente desde la terminal El Recreo hacia el norte, mientras las unidades de flota nueva pueden ser usadas a lo largo de toda la línea troncal. El número de unidades de buses disponibles de flota antigua es 54 y de flota nueva 59. El Trolebús trabaja actualmente con 240 conductores, los mismos que operan bajo distintas modalidades. Los conductores del grupo $4-2$ tienen 4 días de trabajo y 2 días de descanso, no tienen restricciones de fines de semana o feriados y trabajan jornadas de 8 horas en tres modalidades: mañana, tarde o rotativo. El grupo $5-2$ trabaja de lunes a viernes en jornadas de 8 horas, descansa en feriados y fines de semana. Los trabajadores del grupo $5-2$ *apertura cierre* son un caso especial del grupo $5-2$ en el que se encuentran los conductores asignados al inicio y cierre de operaciones. El grupo *expreso* está conformado por los conductores que trabajan 4 horas en la mañana y 4 en la tarde. En el contrato

laboral de cada conductor se garantizan al menos 20 minutos de descanso luego de 4 horas de conducir un vehículo. Se deben pagar horas suplementarias de acuerdo a la ley, a los conductores que excedan 8 horas diarias de trabajo. En horas suplementarias se paga el 50 % adicional por hora trabajada. En horas nocturnas (de 19:00 a 5:59), se paga adicionalmente el 25 % por hora nocturna de acuerdo a la ley. En horas extras (fines de semana y feriados) se paga el 100 % por hora adicional.

Típicamente todo el proceso de planificación en un sistema de transportación pública inicia con la recolección de los datos de la demanda de transportación, la que generalmente se encuentra expresada por medio de las conocidas *matrices origen-destino*. Cada entrada de la matriz origen-destino representa el número de pasajeros que desean viajar desde una zona inicial en la red a un destino dentro de un horizonte de tiempo. A partir de estos datos se debe realizar la planificación estratégica y planificación operacional.

El primer paso dentro de la planificación estratégica se centra en el establecimiento de la red y la infraestructura del sistema, donde se deben identificar las estaciones y las rutas de tránsito de transporte público. A este problema se lo conoce como *diseño de la red*. La siguiente fase es el problema de planificación de líneas (circuitos) y frecuencias que consiste en la selección de un conjunto de líneas y frecuencias de tal forma que la capacidad de transportación sea suficiente para satisfacer la demanda en la red. La tercera fase contempla el diseño del diagrama de marcha, es decir, las líneas y frecuencias se transforman en un plan de viajes. Cada viaje es caracterizado por un tiempo y estación de salida además de su tiempo y estación de llegada. Los tres pasos anteriores conforman el proceso de planificación estratégica.

A partir de estos resultados, se debe enfrentar el proceso asociado a la planificación operacional, el cual se encamina al uso eficiente de los recursos disponibles. Debido a este hecho se debe analizar la asignación adecuada de vehículos y de conductores a los viajes programados. La Gerencia Técnica del sistema de transporte público Trolebús es la encargada de supervisar y controlar las operaciones del Trolebús. Actualmente la asignación de tareas de conducción en el Corredor Central del Sistema Metrobús-Q (Trolebús) es realizada de manera

manual por el Jefe de Operaciones, considerando el problema de cubrir todos los viajes programados y respetando las restricciones laborales de los conductores. Debido a la complejidad del problema, realizar esta asignación de manera manual conlleva algunos inconvenientes como por ejemplo: terminar asignando a algunos conductores más de las ocho horas reglamentarias de trabajo; mientras que, a otros menos, o tener viajes programados no cubiertos por algún conductor. Para hacer frente a estos inconvenientes, proponemos implementar un modelo que encuentre de manera eficaz la asignación de tareas, minimizando costos y consumo de recursos.

El presente trabajo fue desarrollado en el marco del proyecto de investigación "Desarrollo e implementación de modelos matemáticos de optimización para el Sistema Metrobús-Q"[1], en el área Logística y Transporte, enfocado en el desarrollo de modelos matemáticos de optimización para dos problemas que aparecen en la planificación operacional del sistema de Trolebús en Quito: la asignación de vehículos y la asignación de conductores para cubrir el plan de marcha establecido. Este proyecto fue ejecutado conjuntamente por el Centro de Modelización Matemática (ModeMat) y la Secretaría de Movilidad del Distrito Metropolitano de Quito. Dicho proyecto contó con el financiamiento de la Secretaría Nacional de Educación Superior, Ciencia y Tecnología (SENESCYT).

[1]Proyecto Senescyt PIC-13-EPN-002 (Agosto 2013-Mayo 2015)

Capítulo 2

Marco Teórico

El proceso de planificación en el transporte público según Desaulniers and Hickman [2007] está divido en: planificación estratégica, táctica y operacional. Además de un control en tiempo real de las operaciones.

Planificación Estratégica Comprende las decisiones a largo plazo como el establecimiento de la red y la infraestructura del sistema, donde se deben identificar las estaciones y las rutas de tránsito de transporte público. A este problema se lo conoce como *diseño de la red*. Para enfrentar este problema se inicia con la recolección de los datos de la demanda de transportación, la que generalmente se encuentra expresada por medio de las conocidas *matrices origen-destino*. Cada entrada de la matriz origen-destino representa el número de pasajeros que desean viajar entre un conjunto de nodos o zonas, ya sea con una base diaria o para un período específico durante el día.

Planificación Táctica Aborda el problema de planificación de líneas (circuitos) y frecuencias, que consiste en la selección de un conjunto de líneas y frecuencias de tal forma que la capacidad de transportación sea suficiente para satisfacer la demanda en la red. Una vez que se establecen las líneas y las frecuencias del servicio, se debe determinar el tiempo exacto de llegada y de salida de los vehículos a las estaciones. Este paso se lo conoce como el diseño del diagrama de marcha.

Planificación Operativa Los problemas de esta área se encargan de determinar la manera en que las operaciones deben ser conducidas para ofrecer el servicio propuesto al mínimo costo. En contraste con las otras áreas la planificación operativa busca minimizar el costo total. Dado un conjunto de viajes programados, un conjunto de recursos disponibles (buses y conductores) y su distribución sobre los depósitos de la agencia de tránsito, la fase de planificación operacional debe realizar la asignación de vehículos y de conductores que minimicen el costo total mientras respeta todas las restricciones operacionales y las regulaciones laborales. Estos problemas son usualmente resueltos secuencialmente. Las frecuencias y el diagrama de marcha obtenido en la planificación táctica son los datos de entrada para la fase de la planificación operativa.

- El *problema de asignación de vehículos* consiste en asignar los viajes del diagrama de marcha a un conjunto de rutas de vehículos factibles, de manera que cada viaje sea cubierto exactamente por una ruta minimizando una cierta función objetivo. Cada viaje se caracteriza por una terminal de origen, una terminal de destino, una hora de salida, una hora de llegada y una duración. Las rutas son atendidas por vehículos, los cuales deben comenzar y terminar sus tareas en uno o más depósitos. El problema de asignación de vehículos se resuelve primero. Este paso es crucial para determinar si la tabla de marcha puede ser cubierta con la flota de vehículos disponible y cuanto costará. Cuando el problema de asignación de conductores no es factible o su costo operacional es excesivo, se realiza una revisión del diagrama de marcha y de las frecuencias de las rutas para facilitar la asignación de vehículos.
- La *asignación de tareas*, también conocida como *asignación de conductores*, es el segundo paso dentro de la planificación operativa. Este paso es importante desde el punto de vista económico pues considera el pago de rubros asociados a los conductores. Primero se proceden a dividir los bloques de viajes asignados a un mismo vehículo en el paso anterior, a segmentos de acuerdo a los llamados

puntos de alivio. Estos puntos son lugares en los cuales se puede realizar un intercambio de conductores. La asignación de conductores consiste en determinar jornadas laborales para los conductores, es decir asignarles segmentos de trabajo que sean factibles y que respeten los acuerdos laborales, cubriendo todos los bloques de viajes asignados a los vehículos.

- Una vez que se realiza la asignación de conductores, se deben determinar *turnos de trabajo* para los conductores en un horizonte de tiempo. Es decir asignar las jornadas laborales a los conductores para que formen su horario de trabajo, respetando ciertas condiciones como que un conductor no puede trabajar más que un cierto número determinado de días a la semana.

Control en tiempo real Maneja perturbaciones del plan usando varias estrategias de control. Estos problemas se resuelven en tiempo real durante las operaciones y deben minimizar los inconvenientes de los pasajeros.

En Weider [2007] se identifican únicamente dos fases: la estratégica y la operacional. Dentro de la planificación estratégica se incluyen los pasos de determinación de frecuencias y elaboración del diagrama de marcha expuestos en la planificación táctica. En [Borndörfer, 2008] se consideran tres pasos: el diseño del servicio, donde se encuentra el diseño de la red, la determinación de líneas, las frecuencias y el diagrama de marcha; la planificación operativa y las operaciones de control.

En las últimas décadas ha habido un gran desarrollo en las técnicas de optimización matemática en el transporte público, prueba de esto son las Conferencias sobre Sistemas Avanzados en Transporte Público (CASPT), que se realizan periódicamente a nivel mundial desde 1975, donde se tratan temas como: planificación de vehículos y de tripulación; enrutamientos y horarios; servicio de monitoreo, operaciones y despacho; diseño de redes, dimensionamiento de flota y planificación estratégica (ver, por ejemplo, [Voss and Daduna, 2001] y [Hickman et al., 2008]).

El problema que nos ocupa en este proyecto es la ***Asignación de Tareas***, una actividad dentro de la *Planificación Operativa* que busca la asignación de turnos diarios de trabajo para los conductores del sistema con la finalidad de cumplir con todos los viajes programados, minimizando los costos operativos y respetando las normas legales y acuerdos laborales.

2.1. Modelo lineal entero para el problema de asignación de tareas

El problema de asignación de tareas ha sido un tema ampliamente estudiado. En algunos trabajos se lo formula como un problema de particionamiento de conjuntos con una sola función objetivo [Borndörfer et al., 2001] o como un problema multiobjetivo en el que además de la función de costo se incluyen funciones relacionadas con la calidad del servicio [Lourenço et al., 2001]. Se han propuesto algunos métodos para resolver el problema de particionamiento de conjuntos basados en generación de columnas ([Desrochers and Soumis, 1989], [Potthoff et al., 2010] y [Borndörfer et al., 2001]), en heurísticas ([Caprara et al., 1999]), relajaciones lagrangeanas y de programación lineal, y relajaciones del conjunto de estados. Chu and Beasley [1995] han propuesto métodos basados en algoritmos genéticos. En el trabajo de Grossmana and Wool [1997] se hace un estudio comparativo de algunas técnicas que incluyen variantes de los algoritmos glotones, relajaciones fraccionarias y un algoritmo de red neuronal.

A continuación se va a describir un método para resolver el problema de asignación de tareas propuesto por Borndörfer et al. [2001] en el que se propone un método adaptativo basado en la generación de columnas. La generación de columnas es el núcleo principal de varios sistemas comerciales para la planificación de tareas como en el optimizador DS-OPT, el mismo que se describe en el trabajo de Borndörfer. Este optimizador se encuentra implementado en los sistemas de MICROBUS II y BERTA empleados en Berlín.

Se tomaron 3 escenarios europeos para realizar las pruebas computacionales, dos de ellos son problemas de planificación de tareas para la tripulación en buses

urbanos que surgen en las operaciones alemanas, y el otro escenario es un problema de planificación de tareas para conductores del metro, proveniente de una metrópolis europea. Todas las pruebas fueron desarrolladas dentro del sistema MICROBUS II. Las soluciones obtenidas con el sistema DS-OPT contienen un número significativamente menor de tareas, además la calidad de la planificación en términos del cumplimiento de las reglas es al menos comparable con las estadísticas de la mejor solución que han obtenido las compañías. Sistemas similares a DS-OPT son actualmente utilizados como herramientas de ayuda a la decisión en la planificación operativa en el transporte público.

2.1.1. Definición del problema y formulación matemática

Se parte de la situación donde los viajes ya han sido asignados a los vehículos. Los bloques de viajes asignados a un mismo vehículo son divididos en segmentos llamados *nodos de servicio*, de acuerdo a los lugares donde se puede realizar un intercambio de conductores. El *problema de asignación de tareas* (DSP) busca encontrar un conjunto de tareas factibles (secuencia de nodos de servicio que respeten las reglas laborales), de tal forma que cada nodo de servicio sea incluido en una única tarea al mínimo costo.

El *problema de asignación de tareas* es formulado en términos de un modelo de red. El modelo incluye un digrafo acíclico $D = (V, A)$. Los nodos se enumeran de la siguiente manera $V = \{0, 1, 2, ..., m, m+1\}$, donde los nodos $M = \{1, 2, ..., m-1, m\}$ son los *nodos de servicio o elementos de tarea*. Dos nodos u y v se unen por un arco, solo si el mismo conductor puede conducir v inmediatamente después de u. Los nodos 0 y $m+1$ sirven como nodo *fuente* y *sumidero* respectivamente. Estos nodos representan el inicio y el fin de cada tarea. Los nodos fuente y sumidero están conectados a todos por los otros nodos de la red por arcos de entrada y salida $\delta^+(0) = \{(0,1), (0,2), ..., (0, m+1)\}$ y $\delta^-(m+1) = \{(0, m+1), (1, m+1), ..., (m, m+1)\}$, respectivamente, donde $\delta^+(i)$ y $\delta^-(i)$ son los conjuntos de arcos salientes y entrantes a un nodo $i \in V$.

Las *tareas* (duties) corresponden a caminos dirigidos de 0 a $m+1$ en la red. Sin embargo, no todo camino es una tarea, pues la concatenación de nodos de

servicio que representa debe además satisfacer restricciones legales y laborales. La factibilidad y el costo de un camino dependen de los *tipos de tareas*.

Existen dos metodologías para clasificar a las tareas [Weider, 2007]. Las tareas se pueden clasificar por su hora de inicio en tempranas, de medio día y finales. Por otro lado, las tareas pueden ser catalogadas de acuerdo a la duración de la tarea y al número de subtareas. En este contexto, las tareas se clasifican en continuas, horario partido y cortas. Las tareas de *horario partido* son necesarias debido a que en un día típico la demanda del transporte público presenta picos (lapsos de tiempo en los cuales la demanda crece) y por tanto una tarea se puede dividir en dos o a veces tres partes (subtareas) para cubrir la demanda. En este tipo de tareas el tiempo de separación entre las subtareas no se lo contabiliza en la duración de la tarea. Cuando una tarea presenta dos subtareas pero con una separación mínima entre ambas subtareas se la llama *continua*. Las tareas continuas suelen tener un tiempo de desplazamiento (el tiempo contabilizado desde que inicia la tarea hasta que finaliza) de 9 horas mientras que las tareas de horario partido de aproximadamente 12 horas. Las tareas *cortas* presentan menos tiempo de duración que el resto, Estas tareas son usadas frecuentemente por la empresas que tienen trabajadores a tiempo parcial.

Un tipo especial de tarea es el *tripper*, que contiene un único nodo de servicio y que se incluye en el modelo únicamente con la finalidad de inicializar el esquema de generación de columnas.

2.1.2. Modelo de Particionamiento de Conjuntos

El problema de particionamiento de conjuntos (SPP) consiste en que dada una familia de subconjuntos $\mathcal{F} \subseteq 2^{\mathcal{X}}$ de un conjunto base $\mathcal{X}$ y una función de costo $c : \mathcal{F} \to \mathbb{R}_+$, encontrar un subsistema $\mathcal{F}' \subseteq \mathcal{F}$ cuyos conjuntos formen una partición de $\mathcal{X}$ y teniendo una suma de costos mínima. De manera formal, consideremos un conjunto finito y no vacío M y sea $\hat{J}$ el conjunto de todos los subconjuntos factibles de M. Asociado a cada elemento j de $\hat{J}$, se tiene un costo $\hat{w}_j$. El problema de particionamiento consiste en encontrar una colección de elementos de $\hat{J}$, la cual forme una partición de M, donde la suma de los costos de sus elementos sea

mínima. Su formulación se muestra a continuación.

Sean:

- $\hat{J} = \{1, 2, .., \hat{n}\}$ el conjunto de todas las tareas factibles.
- $\hat{\xi} \in \{0, 1\}^{\hat{n}}$ el vector de variables de decisión que indica la inclusión o exclusión de las tareas $j \in \hat{J}$ en el conjunto solución.
- $\hat{w} \in \mathbb{R}_{+}^{\hat{n}}$ el vector de costos no negativos de las tareas.
- $\hat{\Phi} \in \{0, 1\}^{m \times \hat{n}}$ la matriz de incidencias nodos de servicio-tareas.

El *problema de planificación de tareas* puede ser formulado como un problema de particionamiento de conjuntos, de la siguiente manera:

$$(\text{SPP}) \quad \text{mín} \quad \hat{w}^T \hat{\xi}, \qquad \hat{\Phi}\hat{\xi} = 1, \qquad \hat{\xi} \in \{0, 1\}^{\hat{n}}. \tag{2.1}$$

Las restricciones $\hat{\Phi}\hat{\xi} = 1$ estipulan que cada nodo de servicio debe ser cubierto exactamente por una tarea. SPP es un problema que contiene un gran número de variables, por lo cual será abordado por el método de generación de columnas descrito en detalle más adelante. Asociados con SPP, definimos los siguientes programas lineales:

$$(\text{MLP}) \quad \text{mín} \quad \hat{w}^T \hat{\xi}, \qquad \hat{\Phi}\hat{\xi} = 1, \qquad \hat{l} \leq \hat{\xi}, \tag{2.2}$$

$$(\text{RMLP}) \quad \text{mín} \quad w^T \xi, \qquad \Phi\xi = 1, \qquad l \leq \xi,$$

$$(\text{RDLP}) \quad \text{máx} \quad \pi^T \eta + w^T l, \qquad \pi^T \Phi \leq w^T. \tag{2.3}$$

El problema MLP es la relajación lineal de SPP obtenida al sustituir las restricciones $\hat{\xi} \in \{0, 1\}^{\hat{n}}$ (2.1) por $\hat{l} \leq \hat{\xi}$, se conoce también como problema lineal *maestro* en el esquema de generación de columnas. El vector $\hat{l} \in \{0, 1\}^{\hat{n}}$ establece cotas inferiores para las variables de decisión $\hat{\xi}$ y será utilizado en el algoritmo de solución para fijar iterativamente variables.

El problema primal maestro restringido (RMLP) se obtiene de MLP mediante la restricción de sus variables (es decir, de las posibles tareas a considerar en la solución) a un subconjunto

$$J := \{1, 2, ..., n\} \subseteq \{1, 2, ..., \hat{n}\}.$$

RDLP es el programa dual lineal de RMLP. Para su formulación, se ha definido el parámetro $\eta := 1 - \Phi l$.

Dado un vector $\pi \in \mathbb{R}^m$ de precios duales (una solución de RDLP), el *problema de costeo (pricing problem)* consiste en determinar si existe una tarea con costo reducido negativo respecto a π:

$$\text{(PRICE)} \qquad \exists \quad j \in \hat{J} : \qquad \overline{\hat{w}_j} := \hat{w}_j - \pi^T \hat{\Phi}_{.j} < 0. \tag{2.4}$$

El problema de costeo es la componente fundamental del método de generación de columnas descrito a continuación.

2.1.3. Método de Generación de Columnas aplicado al Modelo de Particionamiento de Conjuntos

La generación de columnas es un método para la solución de ciertos programas lineales que tienen un gran número de variables y presentan una estructura especial. En particular, este método ha sido aplicado con éxito al caso del modelo de particionamiento de conjuntos asociado al problema de planificación de tareas. El esquema general es el siguiente:

1. Construir (y resolver) una instancia inicial de RMLP, donde Φ representa un conjunto de tareas lo suficientemente grande para garantizar que el programa tenga al menos una solución factible.

2. Mientras no se satisfaga algún criterio de parada (por ejemplo, una brecha de optimalidad predeterminada), repetir las siguientes operaciones:

   a) Resolver el problema RDLP y obtener un vector de precios duales π.

   b) Resolver el problema PRICE para encontrar una o más nuevas tareas con costo reducido negativo respecto a π, las cuales potencialmente podrían mejorar la solución de RMLP.

   c) Incorporar las columnas asociadas a las nuevas tareas en la matriz Φ.

3. Resolver RMLP. Retornar la solución.

El esquema anterior permite resolver el problema master MLP, es decir, la relajación lineal del modelo de planificación de tareas SPP. Para encontrar la solución exacta a este último, la generación de columnas debe ser incorporada dentro de un esquema de ramificación-y-acotación (branch-and-bound), y debe ser resuelta en cada nodo del árbol del exploración. Este esquema se conoce como *branch-and-price*, pero por lo general no es aplicable computacionalmente a instancias reales de SPP.
En la práctica, la solución óptima a MLP (o una solución factible suficientemente buena) suele ser aproximada a una solución de SPP mediante un método de fijación de variables. Para mejorar este esquema, es común realizar la fijación de variables de manera paralela con la generación de nuevas columnas.

2.1.4. Heurística DS-OPT para la Planificación de Tareas

El Algoritmo 2.1 es la función principal del sistema de asignación de tareas DS-OPT. Como dato de entrada este algoritmo requiere de la estructura rdlp $\{$primal w, matriz Φ, primal l, dual $\pi\}$ que contiene un vector de costos w, una matriz de incidencias nodos de servicio-tareas Φ, un vector de cotas inferiores l y un vector de multiplicadores duales π de una instancia RDLP. También son necesarios los valores constantes no negativos ϵ, δ y τ, cuyo uso será explicado más adelante.
El algoritmo dsopt llama en un lazo a las subrutinas $cgen$, $chuzc$ y aug. La subrutina $cgen$ es la encargada de realizar la generación de columnas con un criterio de $\epsilon\delta$-optimalidad, $chuzc$ selecciona una columna j tal que $\Phi(l + e_j) \leq \mathbb{1}$ para ser fijada, es decir se establece su cota inferior a 1 ($l_j = 1$) y la subrutina aug busca una solución aproximada de RDLP.
El algoritmo parte de una instancia inicial de RDLP e itera la generación de columnas con la fijación de variables del problema RMLP. Este proceso se repite mientras las columnas prefijadas no cubran todos los nodos de servicio. En cada iteración k, la llamada al algoritmo *cgen* es controlada por un mecanismo de *branch-and-generate*, el cual consiste en definir una región de confianza para la función objetivo de RDLP $\pi_k^T \eta_k + w_k^T l_k$. Mientras el valor de la última solución encontrada para RDLP se mantenga por debajo de un cierto umbral Θ (que inicialmente tiene un valor de $-\infty$), se considera que este valor es una aproximación aceptable para el

valor óptimo del problema global MLP, y no se ejecuta la subrutina *cgen* (es decir, no se generan nuevas columnas). En cada iteración del algoritmo donde no se generan nuevas columnas, se fija una variable y se ajusta la solución de RDLP mediante la llamada a aug. La selección de las variables a ser fijadas se hace mediante el llamado a *chuzc*. Una vez que se selecciona una variable j se ajustan los valores de los parámetros l y η (el vector η indica los nodos de servicio que no son cubiertos por las variables fijadas) para forzar a que la solución de RMLP satisfaga $\xi_j = 1$. Cuando la función objetivo sobrepasa el valor de Θ, se considera que la fijación de variables ha alterado tanto al RDLP como para que su solución actual no se aproxime aceptablemente al valor del problema global (también alterado) MLP. En tal situación, *cgen* es llamado para actualizar el RDLP, luego de lo cual se establece una nueva región de confianza a τ porciento sobre el valor de la función objetivo de RDLP actualizada.

```
1  rdlp dsopt (rdlp {w, Φ, l, π}, pos ϵ, pos δ, pos τ);
2  int k ← 0;
3  int j_k ;
4  rdlp {w_k, Φ_k, l_k, π_k} ← {w, Φ, l, π};
5  dual η_k ← 𝟙 − Φ_k l_k;
6  double Θ ← −∞;
7  while (Φ_k^T l_k < 1) do
8      k ← k + 1;
9      if (π_k^T η_k + w_k^T l_k > Θ) then
10         {w_k, Φ_k, l_k, π_k} ← cgen({w_{k−1}, Φ_{k−1}, l_{k−1}, π_{k−1}}, ϵ, δ);
11         Θ ← (1 + τ)(π_k^T η_k + w_k^T l_k) ;
12     end
13     j_k ← chuzc({w_k, Φ_k, l_k, π_k});
14     l_k ← l_{k−1} + e_{j_k};
15     η_k ← η_{k−1} − (Φ_k)_{·jk};
16     π_k ← aug({w_k, Φ_k, l_k, π_k}) ;
17 end
18 return {w_k, Φ_k, l_k, π_k};
```

Algorithm 2.1: Heurística dsopt para la planificación de tareas.

2.1.4.1. Subrutina cgen

El algoritmo llamado ***cgen*** describe una generación de columnas para la solución de MPL (2.2). Este algoritmo llama en un lazo a tres subrutinas importantes:

aug Produce para una instancia arbitraria pero fija de RDLP, una sucesión iterativa de vectores (π_k) que tiene las siguientes propiedades:

$$\begin{aligned} \limsup_{k\to\infty} \quad \pi_k^T \Phi &\leq w^T, \\ \lim_{k\to\infty} \quad \pi_k^T \eta + w^T l &= \nu(w, \Phi, l). \end{aligned}$$

Donde $\nu(w, \Phi, l)$ es el valor óptimo de RMLP (o RDLP). De estas propiedades se sigue que (π_k) satisface las condiciones de factiblidad y optimalidad para RDLP en el límite.

price Esta subrutina resuelve una formulación débil del llamado *problema de costeo* .

$$(\delta - \text{PRICE}) \quad \exists \quad \begin{pmatrix} \hat{w}_j \\ \hat{\Phi}_{.j} \end{pmatrix} \in \begin{pmatrix} \hat{w}^T \\ \hat{\Phi} \end{pmatrix} : \bar{\bar{\gamma}} = (1+\delta)\hat{w}_j - \pi^T \hat{\Phi}_{.j} < 0. \tag{2.5}$$

En adelante, notaremos por $\tilde{x}$ a $(1+\delta)x$, donde δ es un factor positivo suficientemente pequeño.

Si (2.5) se cumple, price regresará como resultado esta columna, mas si no existe una columna de costo reducido δ-negativo, price puede regresar cualquier columna arbitraria.

test Esta subrutina verifica si se cumplen las siguientes condiciones de parada empleadas en el método de generación de columnas, conocidas como $\epsilon\delta$-optimalidad:

$$(\delta - \textit{factibilidad}) \qquad \pi^T \Phi \leq \tilde{w}, \tag{2.6}$$

$$(\epsilon - \textit{optimalidad}) \qquad (1+\epsilon)\pi^T \eta \geq \nu(w, \Phi, l) - w^T l. \tag{2.7}$$

El algoritmo *cgen* parte de una instancia de RDLP fija, sobre la cual itera recursivamente. En cada iteración k, llama a la subrutina *aug* para obtener un vector

solución π_k para RDLP. Luego llama a la subrutina *price* para encontrar nuevas columnas que son añadidas a la matriz Φ. Este proceso se repite mientras sea posible generar nuevas columnas y la condición de $\epsilon\delta$-optimalidad no haya sido alcanzada, lo que se verifica con una llamada a la subrutina *test*. Al terminar, el algoritmo retorna la nueva matriz Φ con las columnas que han sido añadidas, el vector de costos w de las columnas, las cotas inferiores l de ξ (que para las nuevas columnas siempre serán 0) y el vector solución del problema dual π.

2.1.4.1.1. Subrutina price El problema de *costeo*-δ (2.5) puede ser formulado como un problema de camino más corto acíclico con restricciones (ACSP). Las restricciones nos permiten fijar límites en el consumo de los recursos como por ejemplo en el tiempo de conducción, el tiempo de trabajo, el número de piezas de trabajo o en el tiempo de descanso. Las diferentes restricciones consideradas en cada tipo de tarea, se manejan usando problemas de caminos más cortos diferentes.

El ACSP es un problema NP-difícil incluso para una sola restricción de recurso. Para resolver este problema se propone abordarlo mediante técnicas de relajación Lagrangeana. Se usa la relajación Lagrangeana del ACSP para obtener una clase de cotas inferiores que son usadas para establecer un criterio eficiente de bactracking en un algoritmo de enumeración del ACSP.

El algoritmo (***price***) que se usa para resolver el problema de camino más corto con restricciones es un algoritmo enumerativo de búsqueda profunda que usa las etiquetas de distancias Lagrangeanas como criterio para realizar el backtracking. Este algoritmo inicia en el sumidero, y recursivamente va buscando todos los posibles caminos que tengan “costo reducido negativo” hasta llegar a la fuente.

2.1.4.2. Subrutina aug

aug busca la solución $\varepsilon\delta$-óptima de RDLP. Debido al gran número de filas que puede presentar una instancia de RDLP se busca la solución mediante métodos de ascenso dual basado en relajaciones Lagrangeanas.

El algoritmo ***aug*** combina un algoritmo de búsqueda de coordenadas junto con una

técnica del control del tamaño del paso llamada *boxstep*. El algoritmo aug a partir de una instancia del RDLP fija, empieza a repetir el proceso de buscar coordenada por coordenada del vector de precios duales π, si es posible ascender en ella, es decir si el nodo de servicio i aún no ha sido cubierto por las variables fijadas (si $\eta_i = 1$) se añade un valor α a la coordenada π_i en función de los costos reducidos de las columnas que contienen al nodo de servicio i, para obtener un valor aproximado de la solución óptima de RDLP. Este proceso se termina al cumplirse algún criterio de convergencia.

Un problema que se presenta en los cálculos del ascenso en coordenadas es que algunos multiplicadores tienden a tomar valores extremadamente elevados en las primeras iteraciones, los cuales son difíciles de reajustar después. Para resolver este problema se usa la técnica del *boxstep* , el cual trata de mitigar este efecto limitando el tamaño del espacio de búsqueda. El paso de ascenso α no puede ir más allá del límite dado por el boxstep.

2.2. Método de Generación de Columnas

El método de generación de columnas es muy útil en programas lineales donde se trabaja con un gran número de variables pero con un número relativamente pequeño de restricciones. Esta técnica fue implementada por primera vez por Gilmore and Gomory [1961] y Gilmore and Gomory [1963] para resolver el problema del corte de material ("cutting stock").

El método de generación de columnas consiste en resolver problemas de programación lineal donde es computacionalmente imposible escribir todas las columnas (variables del problema) explícitamente, generalmente en problemas que tienen un número exponencial de variables. Llamaremos en adelante *problema maestro* al programa original, que contiene todas las columnas. Consideremos el siguiente problema maestro para explicar el método:

$$
\begin{aligned}
(\mathbf{M}) \quad \text{mín} \quad & c^T x, \\
Ax &= a, \\
x &\geq 0,
\end{aligned}
$$

donde $A \in \mathbb{R}^{m \times n}$, c, $x \in \mathbb{R}^n$ y $a \in \mathbb{R}^m$. El programa dual lineal de (M) es:

$$
\begin{aligned}
(\mathbf{D}) \quad \text{máx} \quad & a^T \lambda, \\
\lambda^T A &\leq c^T, \\
\lambda &\in \mathbb{R}^m.
\end{aligned}
$$

En general la generación de columnas se usa cuando n es mucho mayor que m. Ahora tomamos un subconjunto $J \subseteq \{1, 2, ..., n\}$ de columnas de A y formulamos el llamado *problema restringido*:

$$
\begin{aligned}
(\mathbf{R}) \quad \text{mín} \quad & c_J^T x_J, \\
A_{.J} x_J &= a, \\
x_J &\geq 0.
\end{aligned}
$$

El programa dual de (R) se lo formula a continuación:

$$
\begin{aligned}
(\textbf{DR}) \quad \text{máx} \quad & a^T\lambda, \\
\lambda^T A_{.J} &\leq c_{.J}^T, \\
\lambda &\in \mathbb{R}^m.
\end{aligned}
$$

Será posible resolver (M) si somos capaces de resolver (R) y si adicionalmente existe una forma de resolver el llamado *problema del costeo* $(PRICE)$ para cualquier $\lambda \in \mathbb{R}^m$:

$$(\textbf{PRICE}) \qquad \min_{j=1,2,3,\ldots,n} \quad c_j - \lambda^T A_{.j}.$$

Teorema 1. *Sean x^* y λ^* las soluciones óptimas de los programas restringidos (R) y (DR). Si $(\lambda^*)^T A_{.j} \leq c_j^T \ \forall j \in \{1, 2, \ldots, n\}$, entonces la solución $\bar{x} \in \mathbb{R}^n$ definida por:*

$$\bar{x}_j = \begin{cases} x_j^* & si\ j \in J \\ 0 & caso\ contrario \end{cases}$$

es óptima de (M).

Demostración 2.1. *Basta notar que (DR) es una relajación de (D), pues (D) tiene restricciones adicionales de la forma $(\lambda^*)^T A_{.j} \leq c_j, \forall j \notin J$. En efecto, sea λ^* una solución óptima de (DR) que es factible para (D). Si λ^* no es óptima para (D), entonces existe $\hat{\lambda} \in \mathbb{R}^m$ tal que $(\hat{\lambda})^T A_{.j} \leq c_j, \forall j \in \{1, 2, \ldots, n\}$ y $a^T\hat{\lambda} > a^T\lambda^*$. Pero entonces, $\hat{\lambda}$ es factible también para (DR) y contradice la optimalidad de λ^*.*

Tenemos entonces una solución λ^* óptima de (DR) que también es óptima de (D), por el teorema de la dualidad en la programación lineal, se tiene que $a^T\lambda^* = c_J^T x^*$. Notemos que la solución primal óptima de (R) extendida por:

$$\bar{x}_j = \begin{cases} x_j^* & si\ j \in J \\ 0 & caso\ contrario \end{cases}$$

es una solución factible para (M) y que $c^T\bar{x} = c_J^T x^*$. Hemos encontrado dos soluciones factibles $\bar{x}$ y λ^* para los problemas primal y dual (M) y (D) respectivamente, tales que los valores correspondientes de las funciones objetivo coinciden, entonces por el teorema de la dualidad dichas soluciones factibles son óptimas para sus respectivos problemas. Por tanto, $\bar{x}$ es solución óptima de (M).

Por otro lado supongamos que λ^* es una solución óptima de (DR) y que $\exists j \in n : c_j \leq (\lambda^*)^T A_j$, entonces λ^* no es factible para (D). Al añadir la columna j a (R) aseguramos que se verifique una de las restricciones que hacían falta para que λ^* sea factible para (D), por lo tanto hemos encontrado una columna que potencialmente mejora la función objetivo, entonces su índice se añade a J y se resuelve el problema restringido otra vez. EL proceso continúa hasta que se cumplan las condiciones del Teorema 3.2.
A continuación se analizará un modelo basado en generación de columnas para el problema del Cutting Stock.

2.2.1. Problema de Corte de Material

Una empresa vende a sus clientes rollos de papel de m distintos anchos $b_1, b_2, \ldots, b_m$. Para fabricar los rollos demandados, la empresa compra rollos de un ancho fijo L y los corta de acuerdo a los distintos anchos requeridos por los clientes (se asume que todos los rollos poseen el mismo ancho). Cada rollo de ancho b_i posee una demanda d_i. El objetivo es encontrar el número de rollos de ancho L que se deben usar así como la manera en que deben ser cortados para satisfacer las demandas, minimizando el desperdicio (lo que sobra de los rollos de ancho L una vez cortados los rollos finales).

2.2.2. Un modelo de generación de columnas

Anteriormente se explicó que la técnica de generación de columnas se utiliza para resolver problemas de programación lineal donde las columnas del modelo no son conocidas con toda anterioridad. Para modelar el ejemplo se debe plantear:

El problema maestro (M) Esta es la formulación lineal del problema del corte de material. En este problema cada columna representa un patrón de corte, el cual debe ser factible. Cada columna tiene asociada una variable que indica el número de rollos que van a ser cortados bajo el patrón de corte que represente la columna.

El subproblema del costeo (PRICE) Este problema consiste en determinar si existe alguna columna que aún no haya sido considerada en el problema restringido que pueda mejorar la solución actual. Para ello, se verifica si existe una columna de costo reducido negativo.

El algoritmo que se usa para resolver este problema es:

- Inicialización: Se genera un conjunto de columnas con las cuales el problema maestro sea factible.
- Iteraciones:
  - Resolver el problema lineal restringido.
  - Resolver el problema de costeo y verificar si existe al menos una columna con costo reducido negativo, si es el caso, añadir la columna al problema primal restringido y continuar la siguiente iteración; caso contrario, se ha llegado a la solución óptima y se debe parar el algoritmo.

2.2.2.1. El Problema Maestro

Se busca minimizar el desperdicio producido al cortar los rollos de ancho L en rollos de ancho más pequeños, para esto se define el costo c_j asociado al patrón de corte j de la forma $c_j = L - \sum_{i=1}^{m} b_i a_{ij}$, donde a_{ij} representa el número de rollos de ancho i que se obtienen con el patrón de corte j. Se define también las variables de decisión x_j que indican cuantos rollos deben ser cortados bajo el patrón de corte j. Con esto se plantea el problema maestro (M):

$$
\begin{aligned}
(\mathbf{M}) \quad \text{mín} \quad & \sum_{j=1}^{n} c_j x_j, \\
& \sum_{j=1}^{n} a_{ij} x_j = d_i, \quad \forall \quad i = 1, 2, \ldots, m, \\
& x_j \in \mathbb{R}_+, \quad \forall \quad j = 1, 2, \ldots, n.
\end{aligned}
$$

Tomemos un subconjunto $J \subseteq \{1, 2, 3, \ldots, n\}$ y formulemos el problema primal restringido (R):

$$
\begin{array}{llrcl}
(\mathbf{R}) & \text{mín} & \displaystyle\sum_{j\in J} c_j x_j, & & \\
& & \displaystyle\sum_{j\in J} a_{ij} x_j & = & d_i, \quad \forall \quad i = 1,2,\ldots,m, \\
& & x_j & \in & \mathbb{R}_+, \quad \forall \quad j \in J.
\end{array}
$$

El problema dual (DR) de (R) es:

$$
\begin{array}{llrcl}
(\mathbf{DR}) & \text{máx} & \displaystyle\sum_{i=1}^{m} d_i \pi_i, & & \\
& & \displaystyle\sum_{i=1}^{m} a_{ij} \pi_i & \leq & c_j, \quad \forall \quad j \in J, \\
& & \pi_i & \in & \mathbb{R}, \quad \forall \quad i = 1,2,3,\ldots,m.
\end{array}
$$

2.2.2.2. Subproblema

Se busca la columna con el menor costo reducido: si este costo reducido es negativo, la columna obtenida es incluida en el problema primal restringido. Por otro lado, si el costo reducido mínimo es no negativo, quiere decir que no hay columnas con costo reducido negativo y por lo tanto la solución que se tiene es la óptima.
Sea π_i el valor de la variable dual asociada a la restricción i (en el ejemplo, π_i es la variable dual asociada a la condición de satisfacción de la demanda de rollos de ancho b_i). De esta manera, el costo reducido de la variable j puede ser calculado como:

$$
\begin{aligned}
\bar{c}_j &= c_j - \sum_{i=1}^{m} \pi_i a_{ij}, \\
&= L - \sum_{i=1}^{m} b_i a_{ij} - \sum_{i=1}^{m} \pi_i a_{ij}, \\
&= L + \sum_{i=1}^{m} (-b_i - \pi_i) a_{ij}.
\end{aligned}
$$

Además se debe satisfacer que la columna j debe ser un patrón de corte, por lo tanto todos los rollos obtenidos de esta columna deben caber en el rollo de ancho L, es decir, $L \geq \sum_{i=1}^{m} b_i a_{ij}$. j está fijo, pues es la columna que vamos a generar.

$$
\begin{aligned}
(\textbf{PRICE}) \quad \text{mín} \quad L + \sum_{i=1}^{m} (-b_i - \pi_i) a_{ij}, & \\
\sum_{i=1}^{m} b_i a_{ij} &\leq L, \\
a_{ij} &\in \mathbb{Z}_+, \quad \forall \quad i = 1, 2, \ldots, m.
\end{aligned}
$$

Puesto que L es un valor constante podemos quitarlo de la función objetivo sin alterar el problema, además dado que j es fijo, se puede tomar a las variables a_{ij} como variables x_i . Por otro lado podemos transformar el problema de minimización a uno de maximización de la siguiente manera:

$$
\begin{aligned}
(\textbf{PRICE}) \quad \text{máx} \quad \sum_{i=1}^{m} (b_i + \pi_i) x_i, & \\
\sum_{i=1}^{m} b_i x_i &\leq L, \\
x_i &\in \mathbb{Z}_+, \quad \forall \quad i = 1, 2, \ldots, m.
\end{aligned}
$$

De esta manera hemos encontrado que el *problema del costeo* resultante tiene la forma del *problema de la mochila*. El problema de la mochila consiste en tener m objetos no fraccionables de pesos b_i y beneficios dados en nuestro caso por $b_i + \pi_i$. El peso máximo que puede llevar la mochila es L. Queremos llenar la mochila con objetos, de forma que se maximice el beneficio total, respetando la capacidad de la mochila. El problema de la mochila ha sido ampliamente estudiado, existen algoritmos de programación dinámica (ver [Martello and Toth, 1990]) para su resolución.

2.3. El Problema de Camino más Corto con Restricciones de Recursos(SPPRC)

El problema de camino más corto con restricciones de recursos (SPPRC) es una generalización del problema de camino más corto clásico. El SPPRC aparece como un subproblema en algunas aplicaciones prácticas (principalmente en problemas de enrutamiento de vehículos), debido a que además de encontrar el camino más corto entre dos nodos, es necesario respetar ciertas restricciones como ventanas de tiempo, capacidades de vehículos, restricciones laborales, etc. Un caso particular es el problema de camino más corto con ventanas de tiempo (SPPTW), en el cual el único recurso utilizado es el tiempo. Este problema fue formulado en G. Desaulniers [2000]. En el presente estudio aparece como un subproblema del problema de asignación de tareas y va a servir para modelar el problema de costeo (2.4). Sea $D = (V, A)$ un grafo dirigido y $R = \{1, \ldots s\}$ un conjunto de $recursos$, asociadas con D y R se definen las siguientes funciones:

- La función de *costo* $c : A \longrightarrow \mathbb{Q}$ sobre los arcos de D.
- La función *consumo de recurso* $t^r : A \longrightarrow \mathbb{Q}$ para todo recurso $r \in R$ y para todo arco $a \in A$.
- Las funciones *límite inferior* $a^r : V \longrightarrow \mathbb{Q}$ y límite superior $b^r : V \longrightarrow \mathbb{Q}$ para todo recurso $r \in R$. Para cada nodo $i \in V$, el intervalo $[a_i^r, b_i^r]$ será llamado *la ventana del r-ésimo recurso* de i.

Consideremos un camino no elemental dirigido p en D como una secuencia de nodos $i_0, i_1, \ldots, i_H$. Para cualquier recurso $r \in R$ y para cualquier nodo i_h sobre p, el $nivel$ $T_{i_h}^r$ de r al alcanzar i_h es definido recursivamente como:

$$T_{i_h}^r = \text{máx}\{T_{i_{h-1}}^r + t_{i_{h-1},i_h}^r, a_{i_h}^r\}$$

para $i \leq h \leq H$. El nivel inicial $T_{i_0}^r$ es arbitrariamente escogido de la ventana del primer nodo $[a_{i_0}^r, b_{i_0}^r]$. El camino p se llama $factible$ si, para cada recurso $r \in R$, es posible escoger el nivel inicial de manera que $a_{i_h}^r \leq T_{i_h}^r \leq b_{i_h}^r$ se cumpla para cada nodo i_h en la secuencia. El SPPRC se establece como la tarea de encontrar

un camino dirigido factible de mínimo costo entre dos nodos dados o y d. Este problema es formulado de la siguiente manera:

Sea $D = (V, A)$ la red, donde V y A representan el conjunto de nodos y de arcos respectivamente. Sea o el nodo de origen, d el nodo destino y N el conjunto de todos los otros nodos $N = V \backslash \{o, d\}$.

Una variable binaria X_{ij} es definida para cada arco $(i, j) \in A$ que toma el valor de 1 si el arco (i, j) es usado en la solución y 0 caso contrario. Una variable entera T_i^r es definida para cada nodo $i \in V$ y para cada recurso $r \in R$, para indicar el nivel del recurso r en el nodo i después de visitarlo.

$$\text{mín} \sum_{(i,j)\in A} c_{ij} X_{ij}, \tag{2.8}$$

$$\sum_{j:(o,j)\in A} X_{oj} = \sum_{i:(i,d)\in A} X_{id} = 1, \tag{2.9}$$

$$\sum_{j:(i,j)\in A} X_{ij} - \sum_{j:(j,i)\in A} X_{ji} = 0, \forall i \in N, \tag{2.10}$$

$$X_{ij}(T_i^r + t_{ij}^r - T_j^r) \leq 0, \quad \forall (i, j) \in A, \forall r \in R, \tag{2.11}$$

$$a_i^r \leq T_i^r \leq b_i^r, \qquad \forall i \in V, \forall r \in R, \tag{2.12}$$

$$X_{ij} \in \{0, 1\}, \qquad \forall (i, j) \in A. \tag{2.13}$$

La función objetivo (2.8) consiste en minimizar el costo del camino. La estructura de camino está definida en las restricciones (2.9) y (2.10). Las restricciones (2.11) impiden la formación de subcircuitos en la solución si al menos una función de recurso es estrictamente positiva. Finalmente las restricciones (2.12) garantizan que se respeten las ventanas de los recursos en cada nodo y las restricciones (2.13) son requerimientos sobre las variables de flujo. Este modelo no es lineal debido a las restricciones (2.11). Usando el hecho de que las variables de flujo X_{ij} son binarias, es posible reemplazar las restricciones (2.11) por las siguientes desigualdades lineales:

$$T_i^r + t_{ij}^r - T_j^r \leq (1 - X_{ij}) M_{ij}^r, \quad \forall (i, j) \in A, \forall r \in R,$$

siendo M_{ij}^r constantes suficientemente grandes. Cuando X_{ij} es igual a 0, estas restricciones se vuelven superfluas. Por otro lado, si $X_{ij} = 1$, las restricciones toman

exactamente la misma forma que sus contrapartes no lineales. Es suficiente por (2.12) tomar $M_{ij}^r = \text{máx}\{b_i^r + t_{ij}^r - a_j^r, 0\}$, $\forall (i,j) \in A$, $\forall r \in R$.

2.3.1. Algoritmos para resolver SPPRC

Todos los algoritmos presentados en esta subsección han sido tomados de Torres [2003] y de las referencias allí señaladas. Si se escala por un factor apropiado, t^r,a^r y b^r, pueden ser considerados como valores enteros para todo $r \in R$. Entonces es posible reducir el SPPRC a un problema de camino más corto clásico (SCPP) en un nuevo grafo $\tilde{D} = (\tilde{V}, \tilde{A})$ definido como sigue:

- Se crean varias copias $\hat{\imath}_1, \ldots, \hat{\imath}_{M(i)}$ de cada nodo $i \in V$. Estas copias corresponden a todas las combinaciones factibles de los valores permitidos de los niveles de recursos en i. Estas combinaciones serán llamadas $estados$. Debido a que a^r y b^r son valores enteros hay "sólo" un número finito de estados. De hecho,

$$M(i) = \prod_{r=1}^{s} (b_i^r - a_i^r + 1), \forall i \in V.$$

- Para cada arco $ij \in A$, se introducen arcos con el mismo costo c_{ij} entre todos los pares $(\hat{\imath}_k, \hat{\jmath}_n)$ de estados asociados a i y a j que satisfagan la siguiente condición: Si $T_{\hat{\imath}_k}^r$ y $T_{\hat{\jmath}_n}^r$ son los niveles del recurso $r \in R$ asociados a los estados $\hat{\imath}_k$ y $\hat{\jmath}_n$, entonces $T_{\hat{\imath}_k}^r + t_{ij}^r = T_{\hat{\jmath}_n}^r, \quad \forall r \in R$.

Dados dos nodos o, $d \in V$, es directo ver que para cada camino factible $(o,d) \in D$, existe un camino dirigido con el mismo costo entre ciertos estados $\tilde{o}_k$ y $\tilde{d}_n \in \tilde{D}$, y viceversa. Los algoritmos usuales de caminos más cortos pueden ser aplicados en $\tilde{D}$ para resolver SPPRC. Sin embargo, hay dos dificultades que deben ser consideradas.

La primera es concerniente a la función de costo. Cuando el SPPRC aparece como un subproblema en el cálculo de un camino factible de mínimo costo reducido para problemas de enrutamiento de vehículos, el grafo D contiene usualmente ciclos de costo negativo. En estos casos se conoce que el clásico SCPP se convierte en *NP-difícil*. Afortunadamente, si al menos una función de consumo de recurso toma

valores siempre positivos (por ejemplo, cuando se trabaja con ventanas de tiempo), el grafo $\tilde{D}$ permanece acíclico, y algoritmo de Ford-Bellman puede ser utilizado.
La segunda dificultad se enfoca en el tamaño del grafo auxiliar $\tilde{D}$. La transformación propuesta en la parte anterior no es polinomial, pues el número de nodos en $\tilde{D}$ depende de los valores de las funciones a^r y b^r. Los algoritmos presentados aquí tienen una carácter pseudo-polinomial.
Un algoritmo de solución para el SPPRC no creará el grafo $\tilde{D}$ de manera explícita, los estados de un nodo son generados en la medida que son requeridos bajo la forma de *etiquetas*. Considere un camino factible $p_k \in D$, que comenzando en el nodo de origen o, alcanza el nodo $i \in V$ consumiendo $T^r_{i,k}$ unidades de recurso $r \in R$ y a un costo de $c_{i,k}$. Se asocia a p_k una etiqueta en el nodo i de la forma $f(i,k) = (c_{i,k}, T^1_{i,k}, \ldots, T^s_{i,k})$.
Una simple pero importante observación hecha primero por Aneja et al. [1983] ayuda a reducir drásticamente el número de etiquetas que serán tomadas en cuenta.

Definición 2.1. *Dadas dos etiquetas $f(i,k)$ y $f(i,n)$, se dice que $f(i,k)$* domina *a $f(i,n)$ si las siguientes condiciones se cumplen:*

$$\begin{aligned} c_{i,k} &\leq c_{i,n}, \\ T^r_{i,k} &\leq T^r_{i,n}, \forall r \in R. \end{aligned}$$

Definición 2.2. *Una etiqueta se dice* eficiente *si no existe otra etiqueta que la domine. Un camino que use sólo etiquetas eficientes se denomina* camino eficiente.

Es inmediato que para todo camino factible p_n entre el origen o y otro nodo $i \in V$, existe un camino eficiente p_k de máximo el mismo costo. Por lo tanto al resolver una instancia de SPPRC, es suficiente considerar sólo las etiquetas eficientes.
Denotaremos por Q_i el conjunto de todas las etiquetas en el nodo i y sea $EFF(Q_i)$ el subconjunto de todas las etiquetas eficientes en Q_i.

2.3.1.1. Algoritmo de corrección de etiquetas

En el contexto del problema de camino más corto con ventanas de tiempo, Desrosiers and Soumis [1983] propusieron el algoritmo de *corrección de etiquetas*,

el cual es una adaptación del algoritmo clásico Ford-Bellman-Moore (ver en [Bellman, 1958]). La versión descrita por Torres [2003] ha sido modificada levemente para tomar en cuenta recursos adicionales.

Definición 2.3. *Dada una etiqueta* $f(i,k) = (c_{i,k}, T^1_{i,k}, \ldots, T^s_{i,k})$ *en el nodo* i*, y un nodo* $j \in \Gamma^+(i) \quad (\Gamma^+(i) = \{j \in V : (i,j) \in A\})$*, definimos por* $NEXT(f(i,k), j)$ *la función que retorna la etiqueta* $f(j,n)$ *en* j *definida por:*

$$\begin{aligned} c_{j,n} &= c_{i,k} + c_{ij}, \\ T^r_{j,n} &= \text{máx}\{T^r_{i,k} + t^r_{ij}, a^r\}, \forall r \in R. \end{aligned}$$

Si $f(j,n)$ es infactible debido a la violación de alguna ventana de recurso en j, entonces el valor que retorna la función $NEXT(f(i,k), j)$ es el conjunto vacío.

```
1  Initialization: Q_o ← {(0, T^1_{o,α}, ..., T^s_{o,α})};
2  Q_i ← {(∞, a^1, ..., a^s)} ∀i ∈ V \ {o};
3  L ← {o};
4  while (L ≠ ∅) do
5      { Selection of next label} ;
6      Choose a node i ∈ L ;
7      for (j ∈ Γ^+(i)) do
8          Q'_j ← EFF(∪_k {NEXT(f(i,k), j)} ∪ Q_j) ;
9          if Q'_j ≠ Q_j then
10             Q_j ← Q'_j ;
11             L ← L ∪ {j} ;
12         end
13     end
14     {Reduction of L} ;
15     L ← L \ {i} ;
16 end
```

Algorithm 2.2: Algoritmo de correción de etiquetas.

El Algoritmo 2.2 muestra un esquema del método de corrección de etiquetas. Observe que el conjunto Q_o es inicializado usando alguna etiqueta válida $f(o,\alpha)$. Normalmente esta etiqueta es fijada por la instancia del problema, caso contrario el

algoritmo tiene que ser ejecutado una vez para cada posible estado inicial.

El algoritmo mantiene una lista L de nodos que aún deben ser procesados. Cada vez que un nodo i es procesado, todas las posibles nuevas etiquetas para cada uno de sus sucesores $j \in \Gamma^+(i)$ son generadas. Antes de añadir nuevas etiquetas a un conjunto Q_j, se realiza un chequeo para verificar que éstas no sean dominadas por algunas etiquetas ya existentes. Para instancias que involucren muchos recursos el chequeo requerirá demasiado tiempo computacional. Si son encontradas nuevas etiquetas eficientes, ellas son añadidas a Q_j y j es insertado en la lista L. La ejecución termina cuando la lista esté vacía.

Para el resto de esta sección, nos restringiremos al caso en que al menos una función de consumo de recurso es estrictamente positiva. Sea, por ejemplo t^1 esta función. Puesto que el valor de $T^1_{i,k}$ aumenta para cada nueva etiqueta $f(i,k)$ que es creada, la finalización del algoritmo está garantizada por los límites superiores b^1_i en los nodos.

2.3.1.2. Algoritmo de fijación de etiquetas

En el caso del problema clásico sin restricciones SCPP, si se conoce a priori que la función de costo es no-negativa, entonces este hecho puede ser explotado para mejorar la eficiencia del método de corrección de etiquetas, especificando un cierto orden en el cual los nodos deben ser procesados, de tal manera que ningún nodo sea considerado más de una vez. El conocido algoritmo de Dijkstra se basa en esta idea. Dado que t^1 es estrictamente positiva, es natural preguntarse si una mejora similar se puede realizar aquí. Sin embargo, esto requerirá procesar a las etiquetas independientemente de los nodos a los que pertenecen.

El algoritmo de fijación de etiquetas de Desrochers and Soumis [1988] provee una generalización del algoritmo de Dijkstra para el caso de SPPTW. Otra vez, la versión presentada en el Algoritmo 2.3 ha sido levemente modificado para admitir más de una restricción de recurso. En lugar de mantener una lista L de nodos a ser procesada, este algoritmo define para cada nodo un conjunto P_i de etiquetas permanentes, es decir etiquetas cuyo valor no cambiará más durante la ejecución del algoritmo.

```
1  Initialization:
2  Q_o ← {(0, T^1_{o,α}, ..., T^s_{o,α})};
3  Q_i ← ∅, ∀i ∈ V \ {o};
4  P_i ← ∅, ∀i ∈ V;
5  while (∪_{i∈V(Q_i\P_i)} ≠ ∅) do
6      { Selection of next label} ;
7      Choose a label f(i,k) from ∪_{i∈V(Q_i\P_i)} with T^1_{i,k} minimal ;
8      {Treatment of label f(i,k)} ;
9      for (j ∈ Γ^+(i)) do
10         Q_j ← EFF(Q_j ∪ NEXT(f(i,k), j)) ;
11         P_i ← P_i ∪ f(i,k) ;
12     end
13 end
```

Algorithm 2.3: Algoritmo de fijación de etiquetas.

El paso clave en este método es la selección de la siguiente etiqueta a ser procesada, puesto que esto incluye comparar el valor de $T^1_{i,k}$ entre todas las etiquetas $f(i,k)$ que aún no son marcadas como permanentes. Para acelerar la búsqueda, la idea de crear *buckets*, propuesta por Denardo and Fox [1979] para el SCPP, puede ser aplicada.

Un bucket es una lista no ordenada que contiene todas las etiquetas cuyos valores del primer recurso caen dentro de un intervalo específico. El *h-ésimo* bucket, por ejemplo contendrá las etiquetas $f(i,k)$ para los cuales $hw \leq T^1_{i,k} < (h+1)w$, donde w es una constante llamada el *ancho* de los buckets y es definida por:

$$w = \min_{ij \in A} t^1_{ij}.$$

Entonces, mientras se procesa una etiqueta dentro de un bucket B, todas las nuevas etiquetas creadas pertenecerán a buckets que vienen después de B. En lugar de buscar el mínimo entre todas las etiquetas aún no procesadas, se procesan todas las etiquetas del primer bucket no necesariamente en orden, luego se va al segundo bucket, y así sucesivamente.

Por supuesto, un costo computacional extra tiene que ser pagado para insertar las nuevas etiquetas creadas en el bucket correcto, pero continúa siendo más eficiente que guardar el conjunto entero de etiquetas ordenado.

2.3.1.3. Algoritmo de arrastre

Los dos algoritmos anteriores se basan en extender caminos factibles mediante la examinación de todos los sucesores de un cierto nodo i en cada iteración, una tarea que implica actualizar el conjunto de etiquetas Q_j, $\forall j \in \Gamma^+(i)$. Mientras el número de recursos s aumenta, el costo computacional de estas operaciones crece rápidamente, dado que determinar $EFF(Q_j)$ implica comparar los valores de los niveles $T^r_{j,k}$ entre todas las etiquetas $f(j,k) \in Q_j$ para cada recurso $r \in R$.

```
1  Initialization:
2  Q_o ← {(0, T^1_{o,α}, ..., T^s_{o,α})};
3  π_o ← b^1_o ;
4  Q_i ← ∅, ∀i ∈ V \ {o};
5  π_i ← a^1_i, ∀i ∈ V \ {o};
6  while (∃j ∈ V con π_j < b^1_j) do
7      { Selection of next node} ;
8      Choose j ∈ V with π_j minimal ;
9      {Treatment of node j} ;
10     for (i ∈ Γ⁻(j)) do
11         Q_j ← Q_j ∪_k NEXT(f(i,k), j)) ;
12     end
13     Q_j ← EFF(Q_j);
14     π_j ← mín{b^1_j, π_j + w};
15 end
```

Algorithm 2.4: Algoritmo de arrastre.

Para reducir la frecuencia con que estos conjuntos de etiquetas son actualizados, Desrochers [1986] propuso un enfoque de arrastre para el SPPRC. Su método es descrito en el Algoritmo 2.4. La idea principal es procesar un nodo j examinando

las etiquetas de todos sus predecesores $i \in \Gamma^-(j) \quad (\Gamma^-(j) = \{i \in V : (i,j) \in A\})$ y extendiéndolas como antes. Como todas las nuevas etiquetas son creadas en j, sólo el conjunto Q_j tiene que ser actualizado durante la iteración.

Como en el caso anterior, w es el mínimo valor de la función t^1 sobre A. En cada nodo i, una variable π_i se mantiene para marcar un subconjunto de etiquetas permanentes. Éstas son definidas como las etiquetas $f(i,k)$ de Q_i para las cuales el valor concerniente al primer recurso es menor que π_i.

En cada iteración, el nodo j con el menor valor de π_j es seleccionada. Examinando sus predecesores, nuevas etiquetas son creadas en j y Q_j es actualizada. La minimalidad de π_j garantiza que no serán generadas más etiquetas $f(j,k)$ para las cuales $T^1_{j,k} < \pi_j + w$ se cumpla. El valor de π_j es entonces incrementado por w y una nueva iteración comienza. Después de un número finito de pasos los valores π_i alcanzarán las cotas superiores b^1_i para todos los nodos $i \in V$, y el algoritmo terminará.

Capítulo 3

Asignación de Tareas en el Sistema Metrobús-Q

En el Capítulo 1 se discutió el problema de asignación de tareas de conducción en el Corredor Central del Metrobús; el Trolebús. En este capítulo se propone una solución para este problema mediante la formulación de un programa lineal entero, que respete las condiciones laborales.

3.1. Planteamiento del Problema

El problema de asignación de tareas en el Corredor Central del sistema Metrobús-Q (Trolebús) busca asignar turnos diarios de trabajo para los conductores del sistema, con la finalidad de cumplir con todos los viajes programados, minimizando los costos operativos y respetando los acuerdos laborales. La asignación de tareas involucra en primer lugar restricciones físicas para la concatenación de los viajes programados que podrían ser cubiertos por un mismo conductor; tales como: la duración de los viajes, su origen y su destino. En segundo lugar, existen reglas de índole legal que surgen de restricciones laborales y norman la validez de las tareas, por ejemplo, las reglas que obligan a un conductor a tomar un descanso después de un determinado número de horas de manejo.

La flota del Trolebús es heterogénea y comprende dos tipos de buses, los mismos que de acuerdo a sus condiciones mecánicas operan sólo en determinados

circuitos: la flota tipo 1 es más antigua y opera sólo los circuitos C1, C5 Y C3 y la flota tipo 2 que es flota nueva puede circular a lo largo de toda la línea del Trolebús hasta Quitumbe, es decir, puede realizar todos los circuitos. En base a la tabla de marcha brindada por el Municipio de Quito, se identificaron los circuitos de la Tabla 3.1.

Código	Estación de Salida	Estación de Llegada
C3	Terminal Norte	Terminal Norte
C2-N	Terminal Norte	Terminal Norte
C2-I	Terminal Sur	Terminal Norte
C1-N	Terminal Norte	Terminal Sur
C1-S	Terminal Sur	Terminal Norte
C2-Q	Terminal Norte	Terminal Quitumbe
C2	Terminal Quitumbe	Terminal Norte
CQ-R-I	Terminal Quitumbe	Terminal Sur
CQ-R	Terminal Quitumbe	Terminal Quitumbe
C4	Terminal Quitumbe	Terminal Quitumbe
CQ-MV	Terminal Quitumbe	Terminal Quitumbe
C5	Terminal Sur	Terminal Sur

Tabla 3.1: Circuitos del Trolebús modificados.

En el Sistema Trolebús, una restricción importante recae en el hecho de que cada conductor tiene la custodia legal (compartida o no) de un autobús, a lo largo de su jornada laboral en un día; con esto, a un conductor no se le pueden asignar viajes a cubrir en diferentes autobuses el mismo día. Por esta razón, cada jornada laboral asignada a cada conductor, sólo contendrá viajes que puedan ser realizados bajo el mismo tipo de flota. Por lo tanto, la asignación de tareas a conductores debe realizarse de manera simultánea a la asignación de vehículos a los viajes programados.

Llamaremos en adelante ***tarea***, a la jornada laboral programada para un conductor. Una tarea consiste en dos subtareas, cada una contiene viajes consecutivos que no deben exceder una duración total de 4 horas. Las tareas que poseen una duración

mayor a ocho horas involucran el pago de horas extras para los conductores.
El Trolebús cuenta con ciertos tipos de jornadas específicas descritas en el Capítulo 1, que de acuerdo a su hora de inicio y estructura se las clasificó de la siguiente manera:

Jornadas continuas En este conjunto se encuentran las jornadas: mañana, tarde, apertura y cierre. Estas jornadas deben tener un descanso de aproximadamente 20 minutos entre las subtareas. La duración de una jornada continua es la longitud del intervalo desde el inicio del primer viaje de la primera subtarea, hasta el fin del último viaje de la segunda subtarea.

Jornadas de horario partido En este conjunto se encuentran las jornadas de tipo expreso. Por lo general en este tipo de jornadas la primera subtarea toma lugar en la mañana y la segunda en la tarde. La duración de una jornada de horario partido se calcula como la suma de las longitudes de las dos subtareas.

Cada tarea debe contener un número entero de viajes. Es decir, no es posible terminar o iniciar una tarea en la mitad de un viaje. Se deben respetar además las siguientes restricciones:

- Cada tarea consiste de dos subtareas, las cuales nos deben exceder 4 horas de conducción cada una.
- Cada viaje es cubierto exactamente dentro de una subtarea.
- Las tareas son estructuradas en: continuas y de horario partido.
- Entre dos subtareas de una tarea continua existe un descanso de aproximadamente 20 minutos.
- La duración de una tarea no debe exceder las ocho horas.
- Se cuenta con un número limitado, de buses de cada tipo de flota y de trabajadores, que debe ser respetado.

La solución manual actual del Trolebús no satisface todos los requerimientos expuestos en el planteamiento del problema, además tratar de cumplirlos con

exactitud es una tarea difícil. Por esta razón, se relajaron los requerimientos de la siguiente manera:

- Las subtareas que tengan longitudes de máximo 5 horas son aceptadas.
- El tiempo de descanso en las tareas continuas debe ser como mínimo de 15 minutos y máximo de 2 horas.
- La duración total de una tarea no debe exceder las 9 horas y media.

Sin embargo, la desviación de los requerimientos preestablecidos es penalizado en la función objetivo. Esto se logra mediante un score definido para cada tipo de tarea. Para las tareas continuas el score viene dado por:

$$\hat{w}_c = T_1 + T_2 + 1,5 \mid D_d - 20 \mid +1,5 \mid D_1 - 240 \mid +1,5 \mid D_2 - 240 \mid +2[D_T - 540]^+,$$

donde T_i es el tiempo total en andén dentro de la subtarea $i \in \{1,2\}$, es decir el total que resulta de sumar todos los tiempos en el que el bus debe esperar en una estación para poder realizar el próximo viaje. D_d es la duración del descanso entre ambas subtareas, D_i es la duración de la subtarea i y D_T es la duración total de la tarea, donde $[x]^+$ se define como $\text{máx}\{0, x\}$. Todos los tiempos están representados en minutos. Cabe notar que el último término representa el pago de horas extras.

El score de una tarea de horario partido es calculado de manera similar, sin considerar el término del tiempo de descanso entre ambas subtareas:

$$\hat{w}_{hp} = T_1 + T_2 + 1,5 \mid D_1 - 240 \mid +1,5 \mid D_2 - 240 \mid +2[D_T - 540]^+.$$

Una ruta factible para un bus debe contener sólo un número entero de tareas a su cargo, pues cada conductor mantiene la custodia legal de un único bus. Además el número de tareas que pueden ser ejecutadas en un mismo bus es una o dos, debido a que el tiempo de operación diario del Trolebús es de menos de 20 horas (desde las 5:00 a.m hasta media noche). El score de una ruta es la suma de los scores de las tareas que forman la ruta añadiéndole un valor constante para reflejar el costo fijo del uso de la unidad. El score de la solución es la suma de los scores

de todas las rutas de los buses. Nuestro objetivo es encontrar una solución factible para la versión relajada del problema de asignación de conductores que minimice el score general.

3.2. Formulación del Modelo

Proponemos a continuación un modelo de programación lineal entero para el problema de asignación de tareas de conducción, el mismo que toma como base el trabajo de Borndörfer et al. [2001], descrito en el Capítulo 2.
Sean:

- $\hat{J} = \{1, 2, .., \hat{n}\}$ el conjunto de todas las tareas factibles.
- $\hat{\xi} \in \{0, 1\}^{\hat{n}}$ el vector de variables de decisión que indica la inclusión o exclusión de las tareas $j \in \hat{J}$ en el conjunto solución.
- $\hat{w} \in \mathbb{R}_{+}^{\hat{n}}$ el vector de costos no negativos de las tareas. El costo $\hat{w}_j$ de cada variable se calcula mediante el score definido en la sección anterior dependiendo del tipo de tarea que represente la variable j.
- $\hat{\Phi} \in \{0, 1\}^{m \times \hat{n}}$ la matriz de incidencias nodos de servicio (viajes)-tareas, donde m representa el número de viajes a ser cubiertos.
- $\hat{\Psi} \in \{0, 1\}^{m_f \times \hat{n}}$ la matriz de incidencias tipos de flota-tareas, donde m_f representa el número de tipos de flota.
- $f \in \mathbb{Z}_{m_f \times 1}^{+}$ el vector que indica el número de buses disponibles por cada tipo de flota.
- $\hat{\Omega} \in \{0, 1\}^{m_u \times \hat{n}}$ la matriz de incidencias tipos de jornada-tareas, donde m_u representa el número de tipos de jornada.
- $u \in \mathbb{Z}_{m_u \times 1}^{+}$ el vector que indica el número de conductores disponibles por cada tipo de jornada.

El *problema de planificación de tareas* puede ser formulado como un problema de particionamiento de conjuntos con restricciones adicionales como se muestra en el modelo AT-MBQ (Asignación de Tareas en el Sistema Metrobús-Q), a continuación:

$$
\begin{aligned}
(AT-MBQ) \quad \text{mín} \quad & \hat{w}^T \hat{\xi}, & & \\
\hat{\Phi}\hat{\xi} &= \mathbb{1}, & & (3.1)\\
\hat{\Psi}\hat{\xi} &\leq f, & & (3.2)\\
\hat{\Omega}\hat{\xi} &\leq u, & & (3.3)\\
\hat{\xi} &\in \{0,1\}^{\hat{n}}. & &
\end{aligned}
$$

Las restricciones (3.1) $\hat{\Phi}\hat{\xi} = \mathbb{1}$ estipulan que cada viaje debe ser cubierto exactamente por una tarea. El problema AT-MBQ además de contar con las restricciones de particionamiento contará con otros dos grupos de restricciones: restricciones de trabajadores y restricciones de flota. Las restricciones de flota (3.2) se deben a que en el Trolebús se trabaja con dos tipos de flota: flota1 y flota2, como se mencionó anteriormente. Cada tipo de flota cuenta con un número fijo de unidades disponibles f_i con $i \in \{1, 2, 3, \ldots, m_f\}$, que deben ser respetados. Las restricciones (3.3) se deben a restricciones de tipo laboral, puesto que existen diferentes tipos de jornadas laborales y el número de jornadas planificadas en la solución no debe sobrepasar el número de trabajadores u_i con $i \in \{1, 2, 3, \ldots, m_u\}$, con los que cuenta el Trolebús actualmente, bajo un tipo de contrato específico.

3.3. Modelo de Generación de Columnas

Dado que existe una gran cantidad de posibilidades para combinar viajes dentro de una jornada laboral, es prácticamente imposible escribir todas las columnas (variables del problema) explícitamente en el programa lineal entero AT-MBQ. En este caso donde el número de variables es relativamente grande en comparación con el número de restricciones, se utiliza el método de generación de columnas descrito en la Sección 2.2. Usaremos el método de generación de columnas para resolver la relajación lineal de AT-MBQ. Con esta finalidad, definimos los siguientes

programas lineales asociados con AT-MBQ:

$$
\begin{array}{llrcl}
(MLP-MBQ) & \text{mín} & \hat{w}^T\hat{\xi}, & & \\
& & \hat{\Phi}\hat{\xi} & = & \mathbb{1}, \\
& & \hat{\Psi}\hat{\xi} & \leq & f, \\
& & \hat{\Omega}\hat{\xi} & \leq & u, \\
& & \hat{\ell} & \leq & \hat{\xi}.
\end{array}
$$

El problema MLP-MBQ es la relajación lineal de AT-MBQ cuando $\hat{\ell} = 0$, y es el llamado problema lineal *maestro* en el esquema de generación de columnas. El vector $\hat{\ell} \in \{0,1\}^{\hat{n}}$ establece cotas inferiores para las variables de decisión $\hat{\xi}$ y será utilizado en el algoritmo de solución para fijar iterativamente variables.

$$
\begin{array}{llrcl}
(RMLP-MBQ) & \text{mín} & w^T\xi, & & \\
& & \Phi\xi & = & \mathbb{1}, \\
& & \Psi\xi & \leq & f, \\
& & \Omega\xi & \leq & u, \\
& & \ell & \leq & \xi.
\end{array}
$$

El problema primal maestro restringido (RMLP-MBQ) se obtiene de MLP-MBQ mediante la restricción de sus variables (es decir, de las posibles tareas a considerar en la solución) a un subconjunto

$$J := \{1, 2, ..., n\} \subseteq \{1, 2, ..., \hat{n}\}.$$

Lema 3.1. *RDLP-MBQ es el programa dual lineal de RMLP-MBQ, con los parámetros* $\eta = \mathbb{1} - \Phi\ell$, $\theta = f - \Psi\ell$ *y* $\vartheta = u - \Omega\ell$.

$$
\begin{array}{llr}
(RDLP-MBQ) & \text{máx} & \pi^T\eta + \varpi^T\theta + \varrho^T\vartheta + w^T\ell, \\
& & \pi^T\Phi + \varpi^T\Psi + \varrho^T\Omega \leq w^T, \\
& & \pi \in \mathbb{R}, \quad \varpi \leq 0, \quad \varrho \leq 0.
\end{array}
$$

Donde $\pi \in \mathbb{R}^m$, $\varpi \in \mathbb{R}_-^{m_f}$ *y* $\varrho \in \mathbb{R}_-^{m_u}$

Demostración 3.1. *Para obtener el programa dual de RMLP-MBQ, primero recordemos el problema primal maestro restringido (RMLP-MBQ):*

$$
\begin{aligned}
(RMLP-MBQ) \quad \text{mín} \quad & w^T\xi, \\
\Phi\xi &= \mathbb{1}, \\
\Psi\xi &\leq f, \\
\Omega\xi &\leq u, \\
\ell &\leq \xi.
\end{aligned} \tag{3.4}
$$

Sustituyendo $\zeta = \xi - \ell$ *, se tiene que* $\xi = \zeta + \ell$ *y por tanto el programa* (3.4) *queda expresado de la siguiente manera:*

$$
\begin{aligned}
(RMLP-MBQ) \quad \text{mín} \quad & w^T\zeta \quad + w^T\ell, \\
\Phi(\zeta+\ell) &= \mathbb{1}, \\
\Psi(\zeta+\ell) &\leq f, \\
\Omega(\zeta+\ell) &\leq u, \\
0 &\leq \zeta.
\end{aligned}
$$

De donde,

$$
\begin{aligned}
(RMLP-MBQ) \quad \text{mín} \quad & w^T\zeta \quad + w^T\ell, \\
\Phi\zeta &= \mathbb{1} - \Phi\ell, \\
\Psi\zeta &\leq f - \Psi\ell, \\
\Omega\zeta &\leq u - \Omega\ell, \\
0 &\leq \zeta.
\end{aligned}
$$

Reemplazando los parámetros $\eta = \mathbb{1} - \Phi\ell$ *,* $\theta = f - \Psi\ell$ *y* $\vartheta = u - \Omega\ell$*, RMLP-MBQ toma la forma siguiente:*

$$
\begin{aligned}
(RMLP - MBQ) \qquad & \\
\text{mín} \quad & w^T\zeta \quad + w^T\ell, \\
\Phi\zeta \quad & = \quad \eta, \\
\Psi\zeta \quad & \leq \quad \theta, \\
\Omega\zeta \quad & \leq \quad \vartheta, \\
0 \quad & \leq \quad \zeta.
\end{aligned} \tag{3.5}
$$

Y el programa dual de este último programa lineal está dado por:

$$
\begin{aligned}
(RDLP - MBQ) \quad \text{máx} \quad & \pi^T\eta + \varpi^T\theta + \varrho^T\vartheta + w^T\ell, \\
& \pi^T\Phi + \varpi^T\Psi + \varrho^T\Omega \leq w^T, \\
& \pi \in \mathbb{R}, \quad \varpi \leq 0, \quad \varrho \leq 0.
\end{aligned}
$$

Se usa la siguiente notación y convenciones:

- $\eta = \mathbb{1} - \Phi\ell$ denota el vector de incidencia del residuo de las filas no cubiertas por las variables fijadas.
- $\theta = f - \Psi\ell$ denota el vector que contiene el número de buses de cada tipo que aún no son ocupados por las variables fijadas.
- $\vartheta = u - \Omega\ell$ denota el vector que contiene el número de trabajadores de cada tipo que aún no son ocupados por las variables fijadas.
- Un MLP-MBQ nunca contendrá una cero-columna $\left(\Phi_{.j}; \Psi_{.j}; \Omega_{.j}\right)^T = (O_{m\times 1}; O_{m_f\times 1}; O_{m_u\times 1})^T$.
- $\overline{\hat{w}_j} := \hat{w}_j - \pi^T\hat{\Phi}_{.j} - \varpi^T\hat{\Psi}_{.j} - \varrho^T\hat{\Omega}_{.j}$ es el *costo reducido* asociado a la tarea j.

Dados unos vectores $\pi \in \mathbb{R}^m$, $\varpi \in \mathbb{R}^{m_f}$ y $\varrho \in \mathbb{R}^{m_u}$ de precios duales (una solución de RDLP-MBQ), el *problema de costeo (pricing problem)* consiste en determinar si existe una tarea con costo reducido negativo respecto a π, ϖ y ϱ:

$$
(\text{PRICE} - \text{MBQ}) \qquad \exists \quad j \in \hat{J} : \qquad \overline{\hat{w}_j} := \hat{w}_j - \pi^T\hat{\Phi}_{.j} - \varpi^T\hat{\Psi}_{.j} - \varrho^T\hat{\Omega}_{.j} < 0.
$$

3.4. Generación de nuevas jornadas

El problema de costeo consiste en encontrar una columna j, asociada a una jornada laboral factible para un conductor, la cual al incluirla en el problema RMLP-MBQ, mejore potencialmente la solución actual obtenida. Para esto, se busca la columna que posea el menor costo reducido negativo y se la incluye en RMLP-MBQ.
Cuando no se encuentren más columnas de costo reducido negativo se ha encontrado el óptimo del problema MLP-MBQ con $\hat{\ell} = 0$.

Lema 3.2. *Tomemos los siguientes programas lineales:*

$$
\begin{aligned}
(M) \quad \text{mín} \quad & \hat{w}^T \hat{\xi}, \\
\hat{\Phi}\hat{\xi} &= \mathbb{1}, \\
\hat{\Psi}\hat{\xi} + \hat{s}_1^{'} &= f, \\
\hat{\Omega}\hat{\xi} + \hat{s}_2^{'} &= u, \\
0 \leq \hat{\xi}, \quad 0 \leq \hat{s}_1^{'}, & \quad 0 \leq \hat{s}_1^{'}.
\end{aligned}
\qquad y \qquad
\begin{aligned}
(D) \quad \text{máx} \quad & \pi^T \mathbb{1} + \varpi^T f + \varrho^T u, \\
\pi^T \hat{\Phi} + \varpi^T \hat{\Psi} + \varrho^T \hat{\Omega} &\leq \hat{w}^T, \\
\varpi &\leq 0, \\
\varrho &\leq 0, \\
\pi &\in \mathbb{R}.
\end{aligned}
$$

Donde $\hat{s}_1^{'}$ y $\hat{s}_2^{'}$ son variables de holgura, $\pi \in \mathbb{R}^m$, $\varpi \in \mathbb{R}_-^{m_f}$ y $\varrho \in \mathbb{R}_-^{m_u}$.

$$
\begin{aligned}
(RM) \quad \text{mín} \quad & w^T \xi, \\
\Phi\xi &= \mathbb{1}, \\
\Psi\xi + s_1^{'} &= f, \\
\Omega\hat{\xi} + s_2^{'} &= u, \\
0 \leq \xi, \quad 0 \leq s_1^{'}, & \quad 0 \leq s_1^{'}.
\end{aligned}
\qquad y \qquad
\begin{aligned}
(RD) \quad \text{máx} \quad & \pi^T \mathbb{1} + \varpi^T f + \varrho^T u, \\
\pi^T \Phi + \varpi^T \Psi + \varrho^T \Omega &\leq w^T, \\
\varpi &\leq 0, \\
\varrho &\leq 0, \\
\pi &\in \mathbb{R}.
\end{aligned}
$$

El programa lineal (M) es la relajación lineal de AT-MBQ conocido como programa maestro en el esquema de generación de columnas y (D) su programa dual. (RM) es el programa maestro restringido y (RD) su respectivo programa dual.

Sean $\begin{pmatrix} \xi^* \\ s_1'^* \\ s_2'^* \end{pmatrix}$ *y* $\begin{pmatrix} \pi^* \\ \varpi^* \\ \varrho^* \end{pmatrix}$ *las soluciones óptimas de los programas restringidos (R) y (DR). Si* $\pi^T\hat{\Phi}_{.j} + \varpi^T\hat{\Psi}_{.j} + \varrho^T\hat{\Omega}_{.j} \leq \hat{w}_j, \forall j \in \{1, 2, \ldots, \hat{n}\}$, *entonces la solución* $\bar{x} = \begin{pmatrix} \bar{\xi}^* \\ s_1'^* \\ s_2'^* \end{pmatrix}$ *donde* $\bar{\xi} \in \mathbb{R}^{\hat{n}}$ *está definida por:*

$$\bar{\xi}_j = \begin{cases} \xi_j^* & si\ j \in J \\ 0 & caso\ contrario \end{cases}$$

es óptima de (M).

Demostración 3.2. *Basta aplicar el Teorema 3.2 con* $c^T = (w^T \quad 0_{1\times mf} \quad 0_{1\times mu})$,

$x = \begin{pmatrix} \xi \\ s_1' \\ s_2' \end{pmatrix}$, $a = \begin{pmatrix} \mathbb{1} \\ f \\ u \end{pmatrix}$, $\lambda = \begin{pmatrix} \pi \\ \varpi \\ \varrho \end{pmatrix}$ *y la matriz* $A = \begin{pmatrix} \Phi & O_{m\times m_f} & O_{m\times m_u} \\ \Psi & I_{m_f\times m_f} & O_{m_f\times m_u} \\ \Omega & O_{m_u\times m_f} & I_{m_u\times m_u} \end{pmatrix}$

Se obtiene directamente que $\hat{x}$ *es óptima de (M).*

Notar que (M) es una cota inferior de (MLP-MBQ), pues (MLP-MBQ) posee más restricciones $(\hat{\xi} \geq \hat{\ell})$.

Sean: π el vector dual asociado a las restricciones de particionamiento, ϖ el vector dual asociado a las restricciones de tipo de flota y ϱ el vector dual asociado a las restricciones de tipo de jornada del problema (RMLP-MBQ), el costo reducido de la variable j puede ser calculado como:

$$\overline{\hat{w}_j} := \hat{w}_j - \pi^T\hat{\Phi}_{.j} - \varpi^T\hat{\Psi}_{.j} - \varrho^T\hat{\Omega}_{.j}. \tag{3.6}$$

Además se debe satisfacer que la columna j debe ser una tarea factible y por tanto debe respetar las restricciones listadas en la Sección 3.1. El problema de costeo puede ser formulado como un ***problema de camino más corto con restricciones de recursos***. Se emplean dos recursos para su formulación: tiempo y descanso. El recurso tiempo indica el tiempo transcurrido desde el inicio de la jornada. El recurso

descanso se emplea para modelar la restricción de que cada conductor debe tener un tiempo de descanso después de cierto tiempo de conducir; este recurso indica si el camino ha pasado o no por un descanso al alcanzar el nodo i . Se formula un problema de camino más corto con restricciones de recurso distinto, para cada tarea y para cada tipo de flota, de acuerdo a sus particularidades. La Figura 3.1 muestra un ejemplo de una jornada continua.

Sea $D = (N, A)$ un grafo dirigido y $R = \{1, 2\}$ un conjunto de $recursos$. El conjunto N posee seis tipos de nodos: nodos de viaje, nodos de descanso, nodos de terminales de inicio de jornada, nodos de terminales de fin de jornada, nodo fuente y nodo sumidero. En la Figura 3.1 los nodos o y e representan al nodo fuente y al nodo sumidero, respectivamente. Los nodos de descanso están representados por 5 y 6; los nodos de inicio de jornada son TN, TS y TQ (indican las terminales del Trolebús); los nodos de fin de jornada son Tn, Ts y Tq; y los nodos de viajes son los restantes. Cada nodo de viaje posee una hora de salida del viaje, una hora de llegada, una terminal de salida, una terminal de llegada y una duración, la cual se calcula como la diferencia entre la hora de fin y la hora de inicio del viaje. Los nodos fuente o y sumidero e representan el inicio y el fin de una jornada laboral, respectivamente.

El arco (i, j) existe entre un par de nodos de viajes i y j, si estos nodos son compatibles, es decir si la terminal de salida del nodo j coincide con la terminal de llegada del nodo i y si la hora de salida del viaje j es mayor que la hora de llegada del viaje i con un máximo de diferencia de dos horas. Los arcos $(o, TN), (o, TS), (o, TQ), (Tn, e), (Ts, e)$ y (Tq, e) indican que toda jornada debe empezar y terminar en una de las tres terminales del Trolebús. Los arcos $(TN, 1), (TN, 2), (TS, 3)$ y $(TQ, 4)$ indican que los nodos $1, 2, 3$ y 4 son los viajes de inicio de jornada cuya terminal de salida coincide con su nodo predecesor. Los viajes de inicio de jornada son tomados de acuerdo a su hora de inicio para generar un tipo de jornada específico. Entre cada par de nodos compatibles i y j existe un nodo de descanso d si y sólo si, la diferencia entre la hora de salida del viaje j y la hora de llegada del viaje i es mayor a 15 minutos (por simplicidad no se graficaron todos los arcos entre nodos de viaje y nodos de descanso en la Figura 3.1). Las terminales de salida y de llegada del nodo de descanso d coinciden y son iguales a

la terminal de llegada de i y a la terminal de salida de j. La hora de salida del nodo de descanso d es la hora de llegada del viaje i, y su hora de llegada es la hora de salida del viaje j. Finalmente, existe un arco entre todo nodo de viaje i y el nodo de terminal de fin de jornada que represente la terminal de llegada del viaje i.

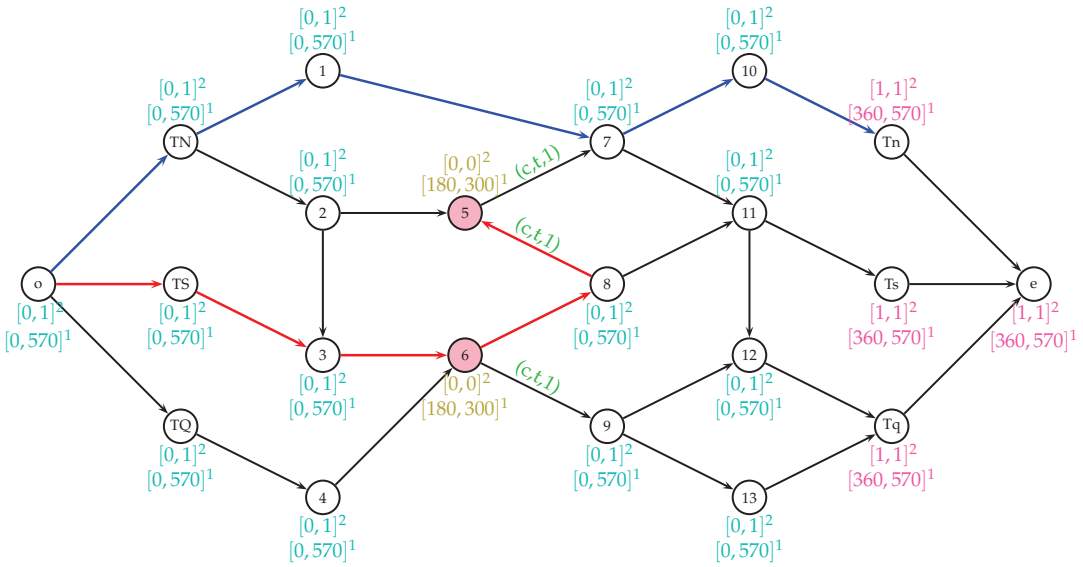

Figura 3.1: Formulación del problema de costeo como un problema de caminos más cortos con dos recursos.

La función de *costo* $c : A \longrightarrow \mathbb{R}$ sobre cada arco (i, j) entre dos nodos de viaje se obtiene como la diferencia entre la hora de salida del nodo j y la hora de llegada del nodo i (tiempo de espera entre viajes) menos el precio dual (respecto a las restricciones de particionamiento) correspondiente al viaje de inicio del arco. La resta de los precios duales correspondientes al tipo de jornada y al tipo de flota, se realiza en los arcos que llegan al nodo sumidero, pues todos los caminos que se generan son del mismo tipo de jornada y del mismo tipo de flota. El costo sobre los arcos (i, j) donde i no sea un nodo de viaje (sin considerar a los nodos de terminales de fin de jornada) es cero. Por otro lado, el costo sobre los arcos (i, j) donde i es un nodo de viaje y j no, es simplemente el precio dual correspondiente al viaje i precedido del signo menos.

La función *consumo del recurso tiempo* $t^1 : A \longrightarrow \mathbb{Z}^+$ sobre cada arco (i,j) indica el tiempo total transcurrido desde el inicio del viaje (o nodo de descanso) i hasta

el inicio del viaje (o nodo de descanso) j. El consumo del recurso tiempo, sobre los arcos cuya nodo inicial es un viaje y cuyo nodo final es una terminal de fin de jornada, indica la duración del viaje. La función consumo del recurso tiempo en el conjunto de arcos restantes es cero, pues ninguno de los dos nodos extremos de estos arcos son nodos de viajes.

La función *consumo del recurso descanso* $t^2 : A \longrightarrow \{0,1\}$, toma el valor de 0 sobre todos los arcos, excepto sobre los arcos que salen de un nodo de descanso donde toma el valor de 1.

Para cada nodo $i \in N$, el intervalo $[a_i, b_i]^r$ será llamado *la ventana del r-ésimo recurso* de i. Las ventanas del recurso tiempo (segunda etiqueta sobre cada nodo) quedan expresados de la manera siguiente:

- Los nodos: fuente o, de terminales de inicio de jornada y de viajes tienen una ventana de tiempo de $[0, 570]^1$. 570 representa el tiempo máximo de una tarea (9 horas y media) expresado en minutos. Esta ventana de tiempo permite que todos los viajes tengan la posibilidad de ser incluidos en una tarea.

- La ventana de tiempo de los nodos de descanso es $[180, 300]^1$. Esta ventana de tiempo fuerza a que el descanso sea tomado entre 3 y 5 horas después de iniciada la jornada laboral.

- Los nodos de terminales de fin de jornada y sumidero, poseen una ventana de tiempo de $[360, 570]^1$. Esta ventana de tiempo se emplea para modelar la restricción de que una tarea tenga una duración total de entre 6 horas hasta 9 horas y media.

Cada nodo, además de poseer una ventana para el recurso tiempo, posee una ventana para el recurso descanso (primera etiqueta sobre cada nodo):

- Los nodos fuente, terminales de inicio y viajes, tienen por ventana de descanso a $[0, 1]^2$. Esto permite que se pueda tomar un viaje, ya sea que se haya pasado por un descanso o no.

- La ventana de descanso de los nodos de descanso es $[0, 0]^2$. Esta ventana establece que se pueda incluir un descanso, solamente si aún no se ha incluido uno previamente.

- Los nodos de terminales de fin de jornada y el nodo sumidero poseen una ventana de descanso de $[1,1]^2$. Con esta ventana de descanso, se asegura que la tarea obtenida incluya obligatoriamente un descanso.

Consideremos un camino no elemental dirigido p en D como una secuencia de nodos $i_0, i_1, \ldots, i_H$. Para cualquier recurso $r \in R$ y para cualquier nodo i_h sobre p, el $nivel\ T^r_{i_h}$ de r al alcanzar el nodo i_h es definido recursivamente como:

$$T^r_{i_h} = T^r_{i_{h-1}} + t^r_{i_{h-1},i_h},$$

para $i \leq h \leq H$. El nivel inicial $T^r_{i_0}$ es tomado como 0 para todo $r \in R$. El camino p se llama $factible$ si, para cada recurso $r \in R$ y para cada nodo i_h en la secuencia, se cumple $a^r_{i_h} \leq T^r_{i_h} \leq b^r_{i_h}$. El problema de camino más corto con dos restricciones de recursos puede ser establecido como la tarea de encontrar un camino dirigido factible de mínimo costo entre los nodos o y e.

La manera en que se definió la función consumo del recurso descanso conjuntamente con las ventanas de descanso, obligan a que todos los caminos entre los nodos o y e incluyan exactamente un descanso. Miremos un ejemplo de un camino que intenta pasar por dos nodos de descanso (5 y 6): el camino rojo $o - TS - 3 - 6 - 8 - 5$ de la Figura 3.1. Una vez que el camino ha pasado por el nodo de descanso 6 la función consumo del recurso descanso del arco $(6,8)$ hace que el nivel T_8 del recurso descanso en el nodo 8 sea 1. Por tanto, el camino no podrá incluir al otro nodo de descanso 6, pues se viola la condición de camino factible $0 \leq T_5 \leq 0$ al ser T_5 igual a 1. Ahora, tomemos otro camino que no incluya en su recorrido un nodo de descanso. Por ejemplo, el camino azul $o - TN - 1 - 7 - 10 - Tn$. Al no incluir el camino un nodo de descanso, todos los arcos que forman este camino tienen por función de consumo del recurso descanso el valor de 0. Así, el el nivel T_{T_n} del recurso descanso en el nodo T_n será 0, por tanto se viola la condición de camino factible $1 \leq T_{T_n} \leq 1$.

Notemos que en esta formulación no se incluyen en los costos considerados todas las componentes de los scores dados por 3.1 y 3.1. Por tal motivo, no trabajamos directamente sobre el grafo sino bajo un esquema de etiquetas que nos permita considerar todas las componentes de los scores.

3.4.1. Algoritmo de Fijación de Etiquetas

El problema de costeo (es decir, la generación de nuevas jornadas laborales) puede ser formulado como un problema de camino más corto con dos recursos. Los algoritmos presentados en la Sección 2.3.1 emplean el concepto de etiquetas eficientes, el cual no es aplicable a nuestro problema particular, como se demuestra en el siguiente ejemplo. Por este motivo, se propone una modificación leve del algoritmo de fijación de etiquetas, para adaptarlo a nuestro problema.

Demostración 3.3. *Considerar un conjunto de 5 viajes programados dentro del Sistema Trolebús, con las características indicadas en la Tabla 3.2. El objetivo es encontrar una jornada laboral factible de mínimo costo. Para simplificar el ejemplo no se considerará ni el descanso ni los precios duales. El costo de cada jornada representa el tiempo en andén total (espera entre viaje y viaje dentro de la jornada laboral).*

Nodo	Hora de salida	Hora de llegada	Terminal de salida	Terminal de llegada
1	06:00	08:59	T.Norte	T.Sur
2	07:00	09:00	T.Quitumbe	T.Sur
3	09:00	11:00	T.Sur	T.Norte
4	11:00	14:00	T.Norte	T.Quitumbe
5	14:30	15:00	T.Quitumbe	T.Norte

Tabla 3.2: Características de los viajes.

En la Figura 3.2 los nodos s y t representan el inicio y el fin de la jornada laboral respectivamente, el conjunto de nodos $\{1,2,3,4,5\}$ *representa al conjunto de viajes que se han tomado en cuenta para este ejemplo. Cada viaje posee una hora de salida, una hora de llegada, una terminal de salida y una terminal de llegada, las mismas que se encuentran sobre su respectivo nodo. Las ventanas de tiempo de todos los nodos excepto t son* $[0,480]$*, la ventana de tiempo del nodo t es [460,500], que corresponde a la duración establecida para una jornada laboral (todos los tiempos están dados en minutos). El arco* (i,j) *existe entre dos nodos que representan viajes si la terminal de llegada del nodo i es la misma que la terminal de salida del nodo j y si la hora de salida del viaje j es mayor o igual que la hora de llegada del*

viaje i, es decir, si se puede realizar el viaje j inmediatamente después del viaje i. Los arcos $(s,1)$, $(s,2)$, $(4,t)$ *y* $(5,t)$ *indican que toda jornada debe empezar con uno de los viajes 1 o 2 (viajes de inicio) y terminar con uno de los viajes 4 o 5 (viajes de finalización). Finalmente, sobre cada arco* (i,j) *se indican dos valores* (c,t)*. El costo c se obtiene como la diferencia entre la hora de inicio del nodo j y la hora de fin del nodo i (tiempo de espera entre viajes) y el consumo del recurso tiempo t indica el tiempo total transcurrido desde el inicio del viaje i hasta el inicio del viaje j.*

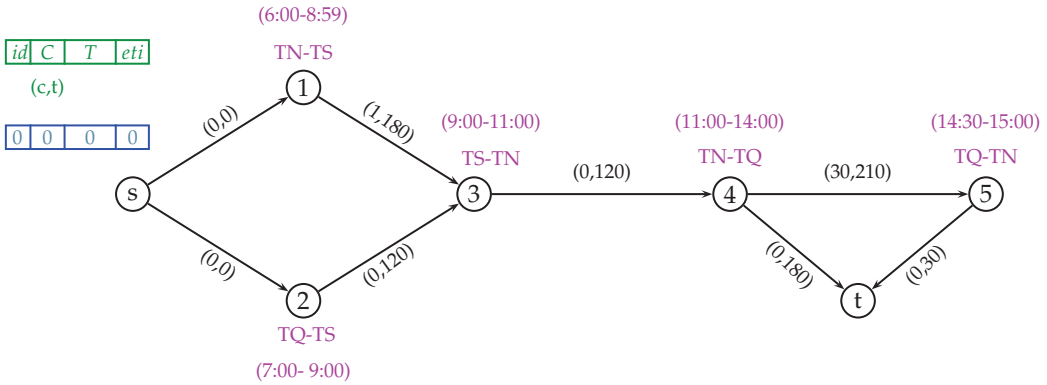

Figura 3.2: Instancia del problema de caminos más cortos con restricciones de recursos para encontrar jornadas laborales.

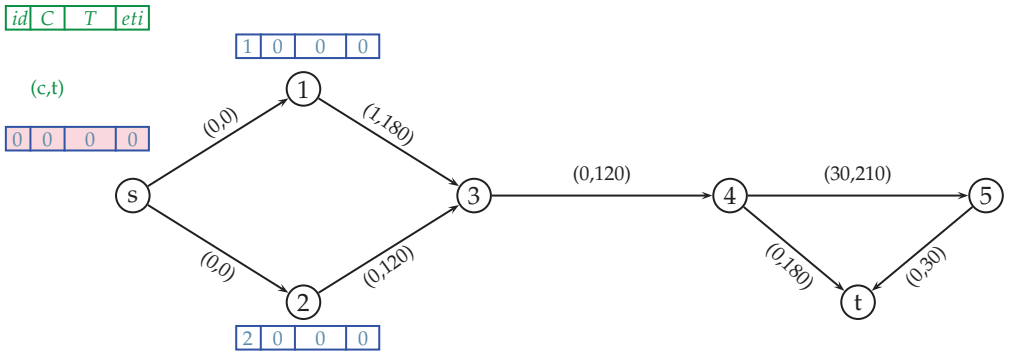

Figura 3.3: Primera iteración del algoritmo de fijación de etiquetas.

Si aplicamos el algoritmo de fijación de etiquetas *en la primera iteración obtenemos las etiquetas indicadas en la Figura 3.3. Cada etiqueta presenta los campos : id, C, T y eti. El campo id representa la identificación de cada etiqueta. C indica el costo acumulado del camino al que representa la etiqueta, tomado desde el primer viaje hasta el viaje al cual pertenece la etiqueta. T indica el tiempo total transcurrido del camino que ha alcanzado*

la etiqueta. eti indica la etiqueta predecesora de la etiqueta actual, este campo nos permite identificar el camino que hemos encontrado. En lo posterior las etiquetas sombreadas serán las etiquetas que ya fueron procesadas por el algoritmo. Siguiendo el procedimiento en la iteración número dos se obtiene lo indicado en el grafo de la Figura 3.4.

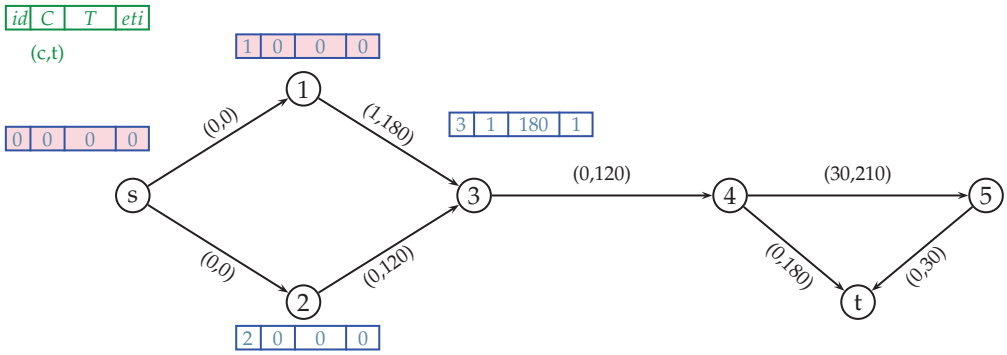

Figura 3.4: Segunda iteración del algoritmo de fijación de etiquetas.

En la Figura 3.5 se muestra la tercera iteración. Al aplicar el concepto de etiquetas eficientes, la nueva etiqueta domina a la etiqueta generada en la iteración anterior al poseer menor costo y tiempo acumulados ($0 < 1$ *y* $120 < 180$)*. Con esto, la etiqueta generada en la iteración anterior (que se encuentra pintada de rojo) es eliminada de la lista de etiquetas por procesar.*

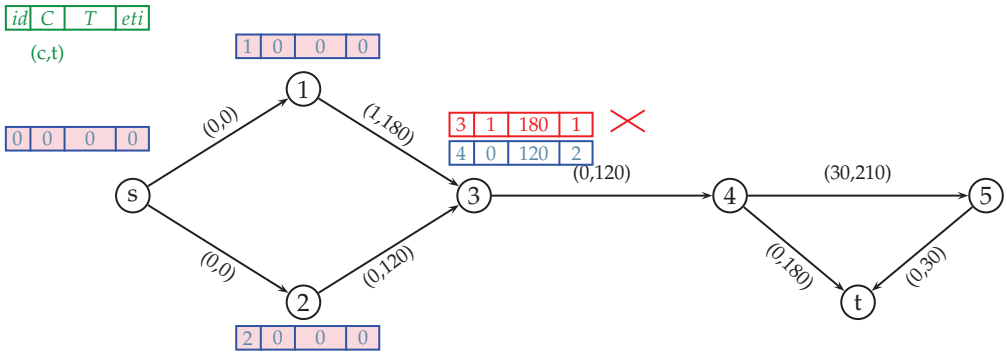

Figura 3.5: Tercera iteración del algoritmo de fijación de etiquetas.

Continuando con las iteraciones, el resultado final se obtiene en la Figura 3.6 , donde el camino más corto viene dado por: $s - 2 - 3 - 4 - 5 - t$ *con un costo de* 30.

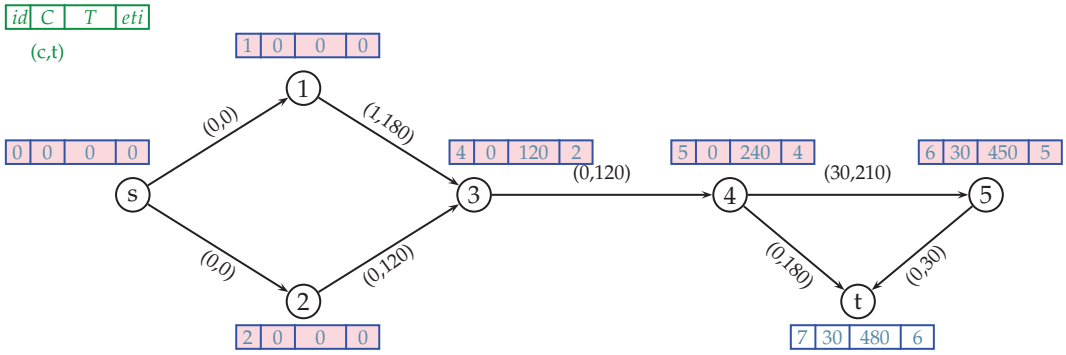

Figura 3.6: Iteración final del algoritmo de fijación de etiquetas sin considerar la etiqueta dominada.

Ahora observemos que pasaría si conservamos la etiqueta de la Figura 3.5 que fue eliminada por el concepto de etiquetas eficientes.

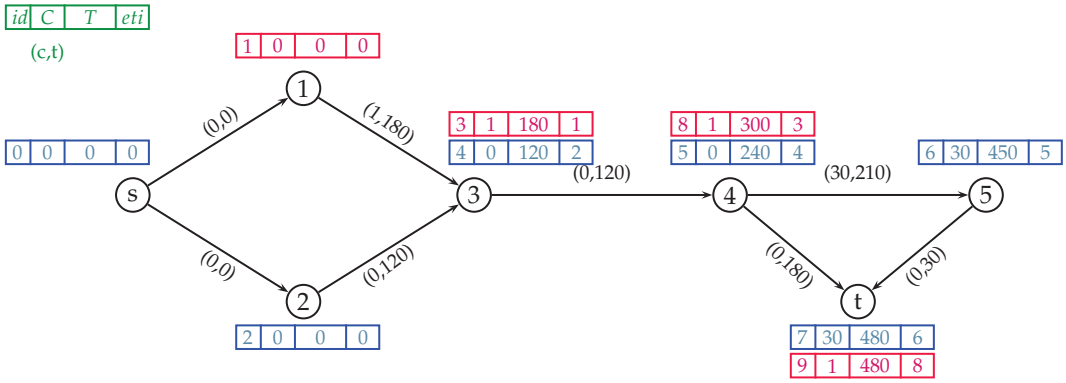

Figura 3.7: Iteración final del algoritmo de fijación de etiquetas considerando la etiqueta dominada.

En la Figura 3.7 observamos en color azul el camino obtenido al eliminar la etiqueta dominada y en color rojo el camino obtenido al conservarla. Comparemos los dos caminos cuyos costos y tiempos se observan en las etiquetas del nodo t. Al retirar la etiqueta dominada se obtuvo el camino $s-2-3-4-5-t$ con un costo de 30. Al conservar la etiqueta dominada se obtuvo el camino $s-1-3-4-t$ con un costo de 1. Es decir, el camino de menor costo requiere el uso de una etiqueta no eficiente. Esto se debe a que el arco "barato"

$(4, t)$ no está habilitado en el camino de etiquetas eficientes (azules), debido a que viola el límite inferior de la ventana de tiempo en t y a que no son posibles las esperas.

Notar que el consumo del recurso tiempo marcado en una etiqueta asociada a un nodo de viaje i, indica el tiempo transcurrido desde el inicio del primer viaje de la jornada, hasta el inicio del viaje i. De esta manera, si se fija el primer viaje para todas las jornadas, este recurso tiene el mismo valor en todas las etiquetas asociadas a un mismo nodo. Empleamos esta idea para modificar el algoritmo de solución. Se encuentra el camino más corto con restricción de recursos mediante el algoritmo de fijación de etiquetas, para cada uno de los posibles nodos iniciales; estos nodos iniciales se fijan de acuerdo a un rango de tiempo, pues al ser los viajes con los que inicia una jornada laboral se puede tener una estimación del rango apropiado, por ejemplo las jornadas de la mañana pueden tener como nodos iniciales todos los viajes que inicien entre las 5 y las 7 de la mañana. Cada nodo poseerá a lo sumo dos etiquetas, una que contenga en el campo del recurso descanso el valor 0 y otra el valor de 1, al ser el recurso tiempo igual para todas las etiquetas del mismo nodo, sólo se comparará el costo para determinar si una etiqueta domina o no a otra. Una vez que se han encontrado todos los caminos más cortos para cada uno de los posibles viajes iniciales, se elige el camino de mínimo costo de entre todos los caminos obtenidos en el paso anterior.

Apliquemos este algoritmo sobre el grafo de la Figura 3.2 para cada uno de los dos posibles nodos iniciales 1 y 2. En este caso, se obtienen las redes de las Figuras 3.8 y 3.9 respectivamente.

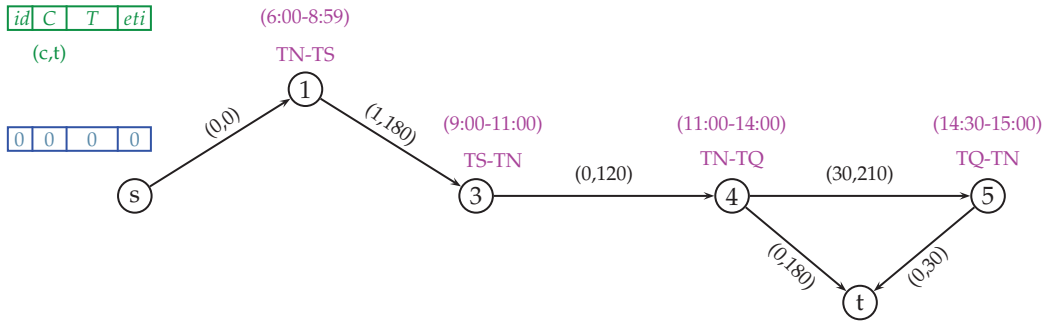

Figura 3.8: Red tomando al nodo 1 como inicio de la jornada laboral.

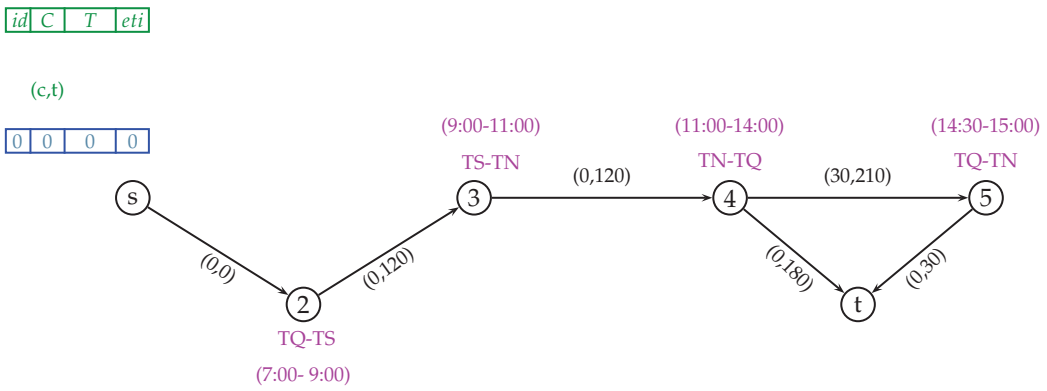

Figura 3.9: Red tomando al nodo 2 como inicio de la jornada laboral.

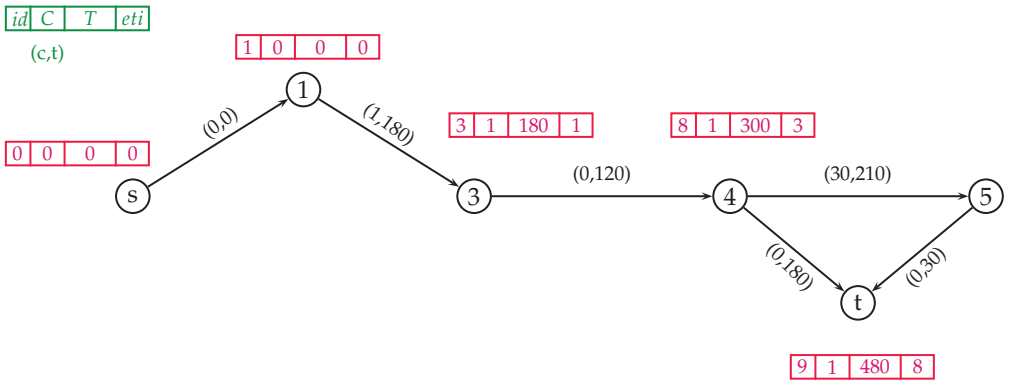

Figura 3.10: Iteración final del algoritmo de fijación de etiquetas tomando como viaje inicial el nodo 1.

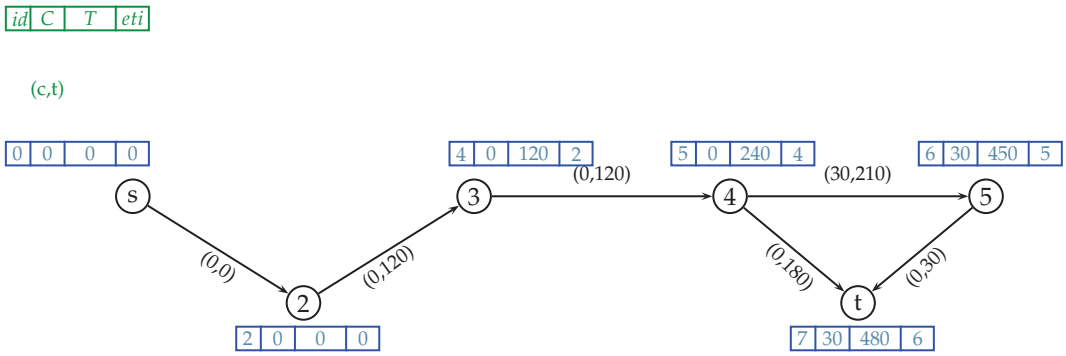

Figura 3.11: Iteración final del algoritmo de fijación de etiquetas tomando como viaje inicial el nodo 2.

Las iteraciones finales del algoritmo de fijación de etiquetas para cada uno de los

dos nodos iniciales se encuentran en las Figuras 3.10 y 3.11 respectivamente. Para finalizar, se toma de todos los caminos encontrados, que en este ejemplo son dos, el camino de mínimo costo que como se vio anteriormente es el que inicia con el nodo 1 a un costo de 1. De esta manera se ha encontrado el camino óptimo. En la implementación del problema de camino más corto que aparece en el problema del costeo se hará uso de esta observación.

3.5. Asignando jornadas a rutas

Cada conductor tiene la custodia legal (compartida o no) de un bus, es decir, a un conductor no se le pueden asignar viajes en diferentes buses y tampoco en buses de diferente tipo de flota. Por este motivo, una ruta factible para un vehículo se compone de jornadas laborales (una o dos) asignadas a los conductores. Las jornadas de horario partido son asignadas a un sólo vehículo, pues esta jornada opera en la mañana y en la tarde. Una manera fácil de conformar las rutas en base a las jornadas laborales, es generar las rutas en la generación de columnas. Así, se generan no solamente jornadas laborales (continuas y de horario partido), sino también rutas que pueden ser asignadas a buses. La generación de rutas se lo formula de igual manera como un problema de camino más corto con restricciones de dos recursos. Primero, se obtiene la jornada laboral de costo mínimo entre todas las jornadas factibles de un mismo tipo que pueden realizarse en la mañana bajo un tipo de flota específico. A continuación, se generan todas las jornadas laborales factibles que pueden realizarse en la tarde bajo el mismo tipo de flota anterior y del mismo tipo de tarea. De entre todas las jornadas de la tarde se eligen a las que sean compatibles con la jornada de costo mínimo de la mañana.

Definición 3.1. *Se dice que la jornada j es compatible con la jornada i, si y sólo si, la jornada i termina en la misma terminal en la que comienza la jornada j y si además la jornada j empieza después de que termina la jornada i. Es decir, si la terminal de llegada del viaje final de la jornada i coincide con la terminal de salida del primer viaje de la jornada j y si la diferencia de la hora de finalización del último viaje de la jornada i, de la hora de inicio del primer viaje de la jornada j es mayor que cero.*

Para finalizar, se selecciona de todas las jornadas de la tarde compatibles con la jornada de la mañana, a la que posea el menor costo. Así, al unir dos jornadas continuas hemos generado una ruta factible para un vehículo.

En la solución de AT-MBQ se encontrarán jornadas laborales permitidas respecto a los requerimientos legales. Sin embargo, se busca también ocupar el menor número de buses para realizar la asignación de tareas. En vista de que no sólo se generan rutas de 16 horas que puedan ser asignadas a un sólo bus, se debe encontrar una manera en que las jornadas continuas se junten en rutas factibles para minimizar el número de buses. El objetivo es maximizar el número de jornadas continuas que queden unidas (rutas). De esta forma, el problema que aparece de manera natural es un problema de emparejamiento máximo (maximum matching).

Sean:

- $I = \{1, 2, 3, \ldots, n_s\}$ el conjunto que contiene a todas las jornadas continuas, donde n_s representa al número de jornadas continuas obtenidas en la solución de AT-MBQ.
- $A_{n_s \times n_s}$ cuyas entradas a_{ij} son iguales a 1 si la jornada j es compatible con la jornada i, y 0 caso contrario.
- x_{ij} las variables de decisión binarias que indican si la jornada j es unida a la jornada i en la solución..

Con la notación introducida en el párrafo anterior, el problema puede ser formulado de la manera siguiente:

$$
\begin{aligned}
(M - MBQ) \quad \text{máx} \quad & \sum_{i=1}^{n_s} \sum_{j=1}^{n_s} x_{ij}, && (3.7)\\
x_{ij} \quad \leq \quad & a_{ij} \quad \forall (i,j) \in I \times I, && (3.8)\\
\sum_{j=1}^{n_s} x_{ij} \quad \leq \quad & 1 \quad \forall i \in I, && (3.9)\\
\sum_{i=1}^{n_s} x_{ij} \quad \leq \quad & 1 \quad \forall j \in I, && (3.10)\\
x_{ij} \quad \in \quad & \{0, 1\}.
\end{aligned}
$$

Esta formulación fue utilizada por facilidad en la implementación y porque es suficiente para el tamaño de los problemas tratados. La función objetivo (3.7) indica que se busca maximizar el mayor número de parejas de jornadas continuas de 8 horas que se puedan unir para formar una ruta de 16 horas. Las restricciones (3.8) prohíben que se seleccione una pareja de jornadas que no pueda unirse de acuerdo a la definición 3.1. Las restricciones (3.9) y (3.10) especifican que una jornada puede ser emparejada como máximo con otra, prohibiendo que una jornada este emparejada al mismo tiempo con varias jornadas.

Capítulo 4

Implementación Computacional del Problema de Asignación de Tareas en el Trolebús

La implementación del presente problema de asignación de conductores en el Corredor Central Trolebús, fue realizada en la interfaz del optimizador Gurobi con C++, en un computador con procesador dual Xeon E5504 con 13,6gb de RAM y sistema operativo Centos 6,5. Gurobi es un solver comercial de optimización ampliamente utilizado, su nombre se debe a sus fundadores Zonghao **Gu**, Edward **Ro**thberg y Robert **Bi**xby. Gurobi es utilizado por alrededor de 700 compañías y cerca de dos docenas de industrias (marketing, publicidad, energía, petróleo, farmacéuticas, minerales, alimentación, transportación, etc) para resolver los más difíciles problemas de negocios y mejorar inmesurablemente funciones que aparecen en esta rama, incluyendo producción, distribución, inversiones de capital, finanzas, compras y recursos humanos. Este solver transforma datos en decisiones inteligentes, es decir, sirve como un asistente de toma de decisiones pues permite a los usuarios expresar sus problemas como modelos matemáticos y automáticamente considera incluso trillones de posibles soluciones para encontrar la mejor solución posible.

El optimizador Gurobi es robusto y capaz de resolver problemas que incluyen millones de variables de decisión. Los solvers en el optimizador Gurobi fueron

diseñados de manera que aproveche las arquitecturas modernas y procesadores multi-núcleo, usando las implementaciones más avanzadas de los últimos algoritmos. Incluye los siguientes solvers: solver de programación lineal (LP), solver de programacipon linear entera-mixta (MILP), solver de programación cuadrática entera-mixta (MIQP), solver de programación cuadrática (QP), solver de programación cuadrática restringida (QCP) y solver de programación cuadrática entera-mixta restringida (MIQCP).

La Figura 4.1 muestra el desempeño de diferentes solvers tanto comerciales como no comerciales respecto a los tiempos de ejecución. El conjunto de datos de prueba de referencia, los cuales son la base de los resultados en la Figura, consiste en un total de 87 problemas que pueden ser resueltos hasta llegar al óptimo por al menos un solver dentro de dos horas respecto al tiempo de ejecución. Este gráfico muestra la media geométrica de los tiempos de ejecución de los resultados tomados de la página principal de Hans Mittelmann (16/Enero/2015) y el número de instancias que fueron resueltas por el solver respectivo.

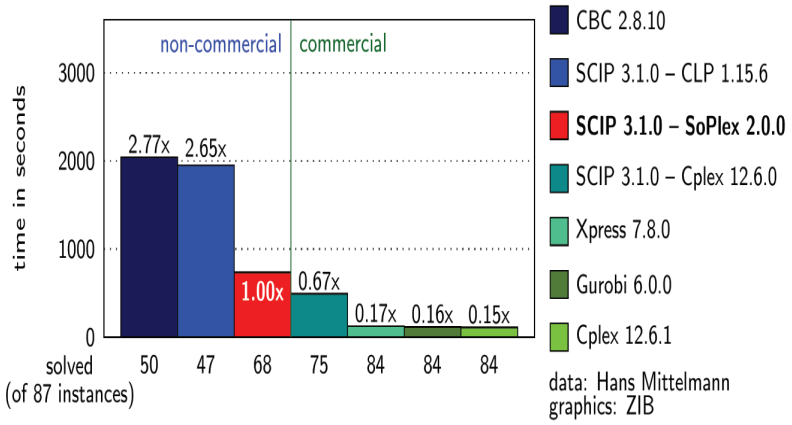

Figura 4.1: Tiempo de ejecución de diferentes solvers aplicados a instancias de problemas de referencia

En la Figura 4.1 los solvers que se encuentran a la derecha son los mejores, pues no solo resolvieron el mayor número de instancias sino que también lo hicieron en el menor tiempo, con lo que se evidencia que el optimizador Gurobi es uno de los mejores solvers comerciales y por este motivo fue utilizado en el presenta trabajo. En Gurobi Optimization [2014] se puede encontrar una guía práctica para el uso de

Gurobi.

4.1. Pseudocódigo del Problema de Asignación de Tareas

A continuación se describe el pseudocódigo del algoritmo de asignación de tareas AT-MBQ; problema presentado en el capítulo anterior.

Data: trips
Result: duties
Initialization: Solve $RMLP - MBQ$ with $\Phi = I_m$, $\Psi = O_{m_f \times m}$ and $\Omega = O_{m_u \times m}$;
Solve $RDLP - MBQ$ and get the dual prices π, ϖ y ϱ ;
Solve $PRICE - MBQ$;
while $(\exists j \in \hat{J} : \overline{\hat{w}_j} < 0)$ **do**
  Add j to $\begin{pmatrix} \Phi \\ \Psi \\ \Omega \end{pmatrix}$;
  Solve $RMLP - MBQ$;
  if $(\exists\, \xi_j$ *with* $j \in J$: *meets the conditions of variable fixing)* **then**
    Fix the variable ξ_j, do: $\ell_j = 1$ y $\ell_j \leq \xi_j$;
    Solve $RMLP - MBQ$;
  end
  Solve $RDLP - MBQ$ and get the dual prices π, ϖ y ϱ;
  Solve $PRICE - MBQ$;
end
Solve $AT - MBQ$;
if $(\quad \exists\, \xi_j^* = e_j$, *where* e_j *the vector with all entries* 0*s except la entrie* j *which value is 1)* **then**
  Join the remaining trippers to duties in the solution $AT - MBQ$.
end
Solve $M - MBQ$;

Algorithm 4.1: Algoritmo de Asignación de Tareas en el Sistema Trolebús.

La entrada del algoritmo es el conjunto de todos los viajes m que realiza el Trolebús en una instancia real, y la salida será el conjunto de jornadas laborales que debe realizar cada conductor, así como el número de buses que serán utilizados.
El esquema general es el siguiente:

1. Construir (y resolver) una instancia inicial de $RMLP - MBQ$, donde $\begin{pmatrix} \Phi \\ \Psi \\ \Omega \end{pmatrix}$ representa un conjunto de tareas lo suficientemente grande para garantizar que el programa tenga al menos una solución factible.

2. Mientras no se satisfaga algún criterio de parada (mientras existan columnas de costo reducido negativo), hacer:

   a) Resolver el problema $RDLP - MBQ$ y obtener los vectores de precios duales π, ϖ y ϱ.

   b) Resolver el problema $PRICE - MBQ$ para encontrar una o más nuevas tareas con costo reducido negativo respecto a π, ϖ y ϱ, las cuales potencialmente podrían mejorar la solución de $RMLP - MBQ$.

   c) Incorporar las columnas asociadas a las nuevas tareas en la matriz $\begin{pmatrix} \Phi \\ \Psi \\ \Omega \end{pmatrix}$ de $RMLP - MBQ$.

   d) Resolver $RMLP - MBQ$.

   e) Verficar si se cumple el criterio de fijación de variables, si ese es el caso, fijar las variables que cumplan el criterio y resolver $RMLP - MBQ$.

3. Resolver $AT - MBQ$ manteniendo las variables fijadas y retornar la solución.

4. Si en la solución quedan trippers

   a) Se unen los trippers a jornadas laborales de la solución con las cuales sean compatibles. Se toma cada viaje al que representa cada tripper y se lo une a la jornada compatible con el viaje, cuyo score al incluir este viaje ya sea a su inicio o a su final sea el menor.

5. Resolver $M - MBQ$ y obtener la solución final.

4.1.1. Instancia inicial de RMLP-MBQ

Con el objeto de que el problema tenga al menos una solución factible, las columnas (tareas) con las que se va a inicializar son los *trippers*, es decir aquellas tareas que cubren exactamente un viaje. Cada tripper poseerá un costo alto $W \in \mathbb{R}+$ y existirá un tripper por cada nodo de servicio $i \in V$, donde V representa el conjunto de todos los viajes. Se resolverá la instancia :

$$
\begin{aligned}
(RMLP - MBQ) \quad \text{mín} \quad & w^T\xi, \\
\Phi\xi &= \mathbb{1}, \\
\Psi\xi &\leq f, \\
\Omega\xi &\leq u, \\
\ell &\leq \xi.
\end{aligned}
$$

Las matrices Φ, Ψ y Ω serán $I_{m\times m}$, $O_{m_f\times m}$ y $O_{m_u\times m}$ respectivamente, es decir:
$\forall i \in V: \quad \exists j \in J: \quad w_j = W \quad and \quad \Phi_{.j} = e_i, \Psi_{.j} = 0_{m_f\times 1} \quad y \quad \Omega_{.j} = 0_{m_u\times 1}$.
Adicionalmente, en la instancia inicial al no haber fijado aún ninguna variable se consideran todas las entradas del vector de cotas inferiores ℓ de las variables como cero, es decir, $\ell = 0_{m\times 1}$.

4.1.2. El problema de costeo

Cada vez que el algoritmo llama a PRICE se crean todos los tipos de jornadas laborales mencionadas antes para cada tipo de flota, y se añaden a las columnas de RMLP-MBQ si su costo reducido es negativo ,es decir si para cada nueva columna $j \in \hat{J}$ tenemos que:

$$
\overline{\hat{w}_j} = \hat{w}_j - \sum_{i=1}^{m} \pi_i \Phi_{ij} - \sum_{i=1}^{m_f} \varpi_i \Psi_{ij} - \sum_{i=1}^{m_u} \varrho_i \Omega_{ij} < 0, \tag{4.1}
$$

donde m representa el número de viajes que deben ser cubiertos, m_f representa el número de tipos de flota que existen y m_u representa el número de tipos de jornadas laborales.

4.1.2.1. Implementación del problema de camino de más corto con dos recursos

A continuación se muestra la manera en que fue diseñada la red aplicada a una jornada laboral específica:

- Sea N el conjunto de nodos, el cual contiene seis tipos de nodos: los nodos de viaje V, los nodos de descanso D que serán generados dinámicamente, los nodos de terminales de inicio de jornada F_1, los nodos de terminales de fin de jornada F_2, el nodo fuente o y el nodo sumidero e. Cada nodo $i \in V$ representa un viaje programado del Trolebús, el mismo que posee una hora de inicio del viaje hs_i, una hora de fin hl_i, una terminal de salida ps_i, una terminal de llegada pl_i y una duración k_i la cual se calcula como hl_i-hs_i. Los nodos fuente o y sumidero e representan el inicio y el fin de una jornada laboral, respectivamente.

- De acuerdo con las leyes laborales una jornada de conducción en el Trolebús es de 8 horas y cada conductor debe tener un descanso de 20 minutos después de aproximadamente 4 horas de conducción, para dar flexibilidad a esta norma permitiremos que un trabajador pueda conducir tres o cinco horas antes de tomar su descanso como mínimo de 15 minutos, y luego continuar conduciendo otras tres o cinco horas, tomando como tiempo máximo de conducción nueve horas y media. A continuación se explican las ventanas de tiempo de cada nodo, $[a_i, b_i]$, con $i \in N$, las cuales están expresadas en minutos:

  - Los nodos: fuente o, sumidero e, de terminales de inicio de jornada y de viajes tienen una ventana de tiempo de $[0, 570]$.
  - La ventana de tiempo de los nodos de descanso es $[180, 300]$.
  - Los nodos de terminales de fin de jornada poseen una ventana de tiempo de $[360, 570]$.

  Cada nodo, además de poseer una ventana para el recurso tiempo, posee una ventana para el recurso descanso.

- Los nodos: fuente o, de terminales de inicio y de viajes $i \in V$, tienen por ventana de descanso a $[0, 1]$.
- La ventana de descanso de los nodos de descanso es $[0, 0]$.
- Los nodos de terminales de fin de jornada y le nodo sumidero e poseen una ventana de descanso de $[1, 1]$.

- La red se genera de manera dinámica, inicialmente posee las siguientes conexiones:

  - El nodo fuente o se enlaza con los tres nodos que representan las terminales de inicio de la jornada (Terminal Norte, Terminal Sur y Terminal Quitumbe), mediante arcos que tienen costo y consumo de recursos 0, es decir $(c_{of}, t^1_{of}, t^2_{of}) = (0, 0, 0)\ \forall f \in F_1$. Al nodo sumidero e se unen los tres nodos que representan las terminales de finalización de la jornada laboral (Terminal Norte, Terminal Sur y Terminal Quitumbe), mediante arcos que tienen consumo de recursos 0 y un costo de $-\varpi_i - \varrho_i$, es decir $(c_{ge}, t^1_{ge}, t^2_{ge}) = (-\varpi_i - \varrho_j, 0, 0)\ \forall g \in F_2$, ϖ_i y ϱ_j son los precios duales asociados al tipo de flota j y al tipo de jornada i, respectivamente.
  - Se dice que el nodo $i \in N$ es compatible con el nodo $j \in N$ cuando existe el arco $(j, i) \in A$.
  - El nodo de viaje $i_0 \in V$, el cual es el nodo con el que se va a iniciar la jornada, es compatible con el nodo de terminal de inicio de jornada $f \in F_1$, si y sólo si, la terminal de salida de i_0 es la misma que la terminal que representa el nodo f. El costo c_{fi_0} y los consumos de los recursos tiempo y descanso $t^1_{fi_0}$ y $t^2_{fi_0}$, asociados al arco (f, i_0) son 0.
  - El nodo de viaje $j \in V$ es compatible con el nodo de viaje $i \in V$, si y sólo si, $pl_i = ps_j$ y si además la espera entre j e i es menor a dos horas, es decir si $0 \leq hs_j - hl_i \leq 120$. El costo c_{ij} asociado al arco (i, j), es el tiempo en andén menos el precio dual del nodo i; el consumo del recurso tiempo t^1_{ij} asociado, se obtiene sumando al tiempo en andén la duración del viaje i; y el consumo del recurso descanso t^2_{ij}, es cero. El tiempo en andén se calcula mediante la diferencia entre la hora de inicio del viaje j (hs_j) y la

hora de fin del viaje i (hl_i), mientras el precio dual se lo representa como π_i. Con esto, $c_{ij} = (hs_j - hl_i) - \pi_i$, $t^1_{ij} = (hs_j - hl_i) + k_i$ y $t^2_{ij} = 0$.

- El conjunto de todos los nodos compatibles con un nodo $i \in N$ se representa por $\Gamma^+(i)$.

Las conexiones que se realizan dinámicamente son:

- Una vez que el camino ρ_ς alcance el nodo $i \in V$; consumiendo $T^1_{i,\varsigma}$ unidades del recurso tiempo, las mismas que añadidas a k_i entren en la ventana de tiempo $[360, 570]$ de los nodos de finalización de jornadas; y consumiendo $T^2_{i,\varsigma}$ unidades del recurso descanso, que sean factibles para la ventana del recurso descanso $[1, 1]$ de los nodos de finalización de jornadas, se crea el arco (i, g) para algún $g \in F_2$, si y sólo si, la terminal de llegada del viaje i es la misma que la terminal que representa el nodo g. El costo c_{ig} asociado al arco (i, g) es $-\pi_i + 1,5 \mid D_2 - 240 \mid +2[D_T - 540]^+$, el tiempo t^1_{ig} es la duración del viaje i, k_i y el consumo del recurso descanso t^2_{ig} es cero.

- Una vez que el camino ρ_ς alcance el nodo $i \in V$; consumiendo $T^1_{i,\varsigma}$ unidades del recurso tiempo, las mismas que añadidas a k_i entren en la ventana de tiempo $[180, 300]$ de los nodos de descanso; y consumiendo $T^2_{i,\varsigma}$ unidades del recurso descanso, que sean factibles para la ventana del recurso descanso $[0, 0]$ de los nodos de descanso, se revisan todos los nodo $j \in \Gamma^+(i)$ y se crea un nodo de descanso cada vez que se verifique la condición:
$(hs_j - hl_i) \geq 20$. Los nodos de descanso creados poseen las siguientes características:

  - $hs_d = hl_i$.
  - $hl_d = hs_j$.
  - $k_d = hl_d - hs_d$.

  Con las características anteriores se crean los arcos (i, d) y (d, j) de la siguiente manera:

  - $(c_{id}, t^1_{id}, t^2_{id}) = (-\pi_i + 1,5 \mid D_1 - 240 \mid, k_i, 0)$.
  - $(c_{dj}, t^1_{dj}, t^2_{dj}) = (hs_j - hl_d + 1,5 \mid D_d - 20 \mid, k_d, 1)$.

4.1.3. Fijación de Variables

La fijación de variables se hizo necesaria pues, al realizar pruebas computacionales sin esta técnica para la primera instancia, quedaban alrededor del 25 % de viajes como trippers al resolver el programa entero debido a que en la solución del programa lineal, los valores de las variables en la función objetivo eran demasiado pequeños (incluso menor a 0.05), además de que el tiempo computacional era de alrededor de 13 días.

En el esquema empleado, se fijan nuevas variables cuando el descenso relativo de la función objetivo z después de mínimo l iteraciones del algoritmo de generación de columnas es menor que un cierto ϵ, es decir cuando:

$$\frac{z_{a-l} - z_a}{z_{a-l}} < \epsilon \quad \text{y} \quad a - a_0 >= l.$$

Donde a representa a la iteración actual del algoritmo y a_0 es la iteración donde se realizó la última fijación de variables. z_a indica el valor de la función objetivo en la iteración actual, mientras z_{a-l} nos indica el valor obtenido hace l iteraciones. La segunda condición asegura que hayan pasado al menos l iteraciones desde la última vez que se fijaron variables para volver a fijar variables. Para los experimentos se toma $l = 50$.

Los criterios para fijar variables son:

- se fijan las variables que posean el mayor valor en la solución actual ($\text{máx}\ \xi_j^\star$),
- se fijan las variables cuyo aumento en el costo dividido entre el número de viajes que cubren, sea el menor posible $\left(\text{mín}\ \frac{w_j(1-\xi_j^\star)}{n_j}\right)$.

Donde $\xi_j^\star$ representa el valor de la variable ξ_j en la solución actual de RMLP-MBQ y n_j el número de viajes que son cubiertos por la variable j.

4.2. Estructura de Datos

La implementación del algoritmo se realizó en la iterfaz de Gurobi con C++. Se definieron clases específicas dentro de C++, para implementar los modelos mencionados en la sección anterior.

4.2.1. Clase Viajes

La clase viajes fue implementada para guardar la información de los viajes programados por el Trolebús.

```
class viajes
{
    public:
        viajes();
       ~viajes();

       unsigned int id;
           int horas_lle;
           int horas_sa;
           int duracion;
           int hora_lle;
           int hora_sa;
           int minuto_lle;
           int minuto_sa;
           int flota;
           string terminales_lle;
           string terminales_sa;
         string circuito;
};
```

Cada viaje presenta:

- Una identificación (*id*).
- Una hora de salida y una hora de llegada. Estas horas se encuentran presentadas bajo: *horas_lle* y *horas_sa* (en minutos); mientras en: *hora_lle, hora_sa, minuto_lle y minuto_sa* , se presenta la hora y los minutos de llegada y salida, respectivamente.
- Una *duración*, la cual se calcula como la resta de la hora de salida del viaje de la hora de llegada.
- Una terminal de llegada (*terminales_lle*) y una terminal de salida (*terminales_sa*).

- El tipo de *flota* que puede cubrir este viaje.
- Un *circuito*, determinado por la Tabla 3.1.

Para definir la red de la Subsección 4.1.2 sobre la cual resolver el problema de camino más corto, se crearon las clases: arco y nodo.

4.2.2. Clase Arco

Esta clase posee los arcos que formarán la red para la implementación del problema de camino más corto con dos recursos. Esta clase posee los siguientes campos:

- La identificación de los nodos que une dicho arco, el nodo de inicio (*inicio*) y el nodo de fin (*fin*).
- Un costo (c), que representa el tiempo inactivo del conductor entre viajes.
- Un tiempo (t), el cual indica la distancia (en minutos) entre el inicio de un viaje y el siguiente.

```
class arco
{
    public:
        arco();
       ~arco();

       int t;
       double c;
       unsigned int inicio,fin;
};
```

4.2.3. Clase Nodo

Esta clase contiene los nodos para generar la red de caminos más cortos. Cada nodo posee las siguientes características:

- Una identificación (*id*), que permite diferenciar si el nodo es: de descanso, de terminal o un viaje.

- Una ventana de tiempo, cuyo límite inferior viene dado por *a* y cuyo límite superior por *b*.

```
class nodo
{
    public:
        nodo();
        ~nodo();

       int  b;
       int a;
       unsigned id;
};
```

4.2.4. Clase Etiqueta

Esta clase nos permite almacenar en cada nodo, las etiquetas que se utilizan al usar el algoritmo de fijación de etiquetas para resolver el problema de camino más corto con restricciones. Las etiquetas poseen los siguientes campos:

- Una identificación *id*.
- El costo que se va almacenando a lo largo de un camino. Todo camino posee dos tipos de costos: el *costo* que representa el costo reducido dado por (4.1), y el *costo_real* , mismo que será ubicado en la función objetivo de RMLP-MBQ a minimizar.
- El tiempo *T* acumulado hasta alcanzar el nodo, al cual pertenece dicha etiqueta.
- Los tiempos: *tiempo_t1, tiempo_t3, tiempo_t4 y tiempo_scor*. Estos tiempos nos permiten calcular el score dado en (3.1) y (3.1).
- Una ubicación *ubication*, que nos indica si el viaje al que pertenece dicha etiqueta, es realizado antes o después de tomar el descanso.

- Los campos *et_procedente y eti_procedente*, indican el nodo y la etiqueta de la cual procede la etiqueta actual. Estos campos nos permiten recuperar el camino de mínimo costo, una vez que ha finalizado el algoritmo de fijación de etiquetas.

- Una identificación *descanso_unico*, que nos permite conocer si el camino hasta ahora encontrado ha pasado o no por un descanso. Si aún no se ha pasado por un descanso, este campo posee el valor de 0; caso contrario, posee el valor de 1.

```
class etiqueta
{
    public:
        etiqueta();
       ~etiqueta();

       unsigned int id;
       double costo;
       double costo_real;
       double T;
       double tiempo_t1;
       double tiempo_t3;
       double tiempo_t4;
       double tiempo_scor;
       int ubication;

       unsigned int et_procedente;
       unsigned int eti_procedente;
       unsigned int descanso_unico;
};
```

4.2.5. Clase Duty

Esta clase permite almacenar la información de cada jornada laboral encontrada. Los campos de esta clase son:

- Una identificación *id*.

- Los mismos costos (*costo_reducido* y *costo_real*) definidos en la clase *etiqueta*, con la diferencia de que estos costos, no pertenecen al camino alcanzado hasta llegar a un cierto nodo; sino, pertencen a los costos finales de una jornada laboral completa.

- Al igual que los costos, cada jornada posee el tiempo total que dura una jornada laboral (*tiempo_to*).

Todos los operadores declarados en esta clase nos permiten comparar las jornadas laborales, en función del costo reducido.

```
class duty
{
    public:
        duty();
       ~duty();

        double costo_reducido;
        double costo_real;
        unsigned int id;
        double tiempo_to;
        bool operator ==( const duty &d2) ;
        bool operator <( const duty &d2) ;
        bool operator >( const duty &d2) ;
};
```

4.2.6. Clase Score

La clase Score permite guardar la información de los scores de cada tipo de jornada laboral.

Esta clase posee los campos:

- El tiempo *tiempo_t1*, que representa el tiempo transcurrido desde el inicio de la jornada hasta tomar el descanso; el tiempo *tiempo_t2*, que representa el tiempo desde que se toma el descanso hasta terminar la jornada laboral; el tiempo *tiempo_t3*, el cual representa el tiempo total inactivo dentro de la jornada laboral; el tiempo *tiempo_t4*, que es el tiempo total desde el inicio de

la jornada hasta su fin; y, el *tiempo tiempo_d1*, el cual representa el tiempo tomado de descanso dentro de la jornada. Todos estos campos se definen para una jornada de tipo separada y partida.

- Cuando la jornada es de tipo junta, además de los campos anteriores se presentan los campos: *tiempo_t5, tiempo_t6, tiempo_t7, tiempo_t8 y tiempo_d2*. Estos campos representan lo mismo que los anteriores respectivamente, la única diferencia es que se definen para la segunda jornada laboral.

- Se define el campo *tipo_sco*, para diferenciar el tipo de jornada de la cual se va a calcular el score.

```
class score
{
public:
        score();
        virtual ~score();

        double tiempo_t1;
        double tiempo_t2;
        double tiempo_t3;
        double tiempo_t4;
        double tiempo_t5;
        double tiempo_t6;
        double tiempo_t7;
        double tiempo_t8;
        double tiempo_d1;
        double tiempo_d2;
        int tipo_sco;

        void calcular_score( const score &sco, double &scor_final);
};
```

Se define además, la funcion *calcular_score*; esta función permite calcular el score de una jornada laboral dada.

4.2.7. Clase Jornadas

Esta clase es la encargada de encontrar las jornadas laborales (tareas) por medio del algoritmo de fijación de etiquetas; la clase jornadas llama a las clases: *viajes, arco, nodo, etiqueta, duty* y *score*. Las funciones que maneja esta clase son:

leer_final y leer_final_flota1 Estas funciones son las encargadas de leer la información de entrada (tablas de viajes programados de cada terminal de salida). Cada una llama a las funciones: *leer_viajes* y *leer_viajes_flota1*, respectivamente, para guardar la información de los viajes (de acuerdo al tipo de flota), en las estructuras que llevan el mismo nombre (*viajes*).

leer_grafo_separado, camino_separado y camino_partido La primera función genera la red del problema de camino más corto con restricciones; para ello, llama a las clases *nodo* y *arco*. La red generada tiene como viaje de inicio un solo viaje seleccionado dentro de un rango de horas adecuado. Las funciones: *camino_separado y camino_partido*, son específicamente la aplicación del algoritmo de fijación de etiquetas (para una jornada tipo continua y de horario partido, respectivamente), a la red generada con la función anterior.

jornada_separada Esta función encuentra para cada uno de los posibles viajes iniciales (dentro del tipo de jornada continua que se esté creando), la jornada laboral de menor costo. Al final de entre todos estos caminos, elije el de menor costo entre todos; obteniendo como solución final el camino (factible) de mínimo costo.

jornada_junta_partes y jornada_junta Estas funciones son creadas para generar rutas, es decir presenta dos jornadas continuas unidas. La primera función *jornada_junta_partes* se encarga de guardar en una matriz, todos los caminos más cortos que se encuentren para los viajes iniciales. La función *jornada_junta*, primero genera todos los caminos más cortos que se puedan encontrar en la mañana y en la tarde mediante la función anterior. Luego, elige el camino de mínimo valor de todos los de la mañana, y a este camino, lo une al camino de la tarde de menor costo de todos los caminos que sean

compatibles (en sentido de terminal y de hora). Finalmente, se obtiene una ruta de 16 horas.

jornada_partido_partes y jornada_partida La primera función permite crear las subtareas de una tarea de horario partido. Se encuentra primero la subtarea de mínimo costo de entre todas las posibles subtareas que se puedan realizar en la mañana. A continuación, se calculan todas las subtareas de la tarde que sean compatibles con la subtarea de menor costo de la mañana. Finalmente se une la subtarea de la mañana a la subtarea de la tarde de mínimo costo. Este proceso lo realiza la segunda función definida.

```
class jornada
{
public:

void leer_viajes(const string & file_name, vector<viajes> & viaje, double
    & j);
void leer_final(vector<viajes>& viaje, vector<double>& costos_iniciales);

void leer_viajes_flota1(const string& file_name, vector<viajes> & viaje,
   double &j);
void leer_final_flota1( vector<viajes> & viaje, vector<double> &
   costos_iniciales );

void leer_grafo_separado(const vector<bool> &viajes_fijos, const int&
   jor_a, ... , const unsigned int& viaje_tomado_inicial);

void camino_separado(score& scr, const int ubicaci1, ... , double &
   costo_duty, double & costo_duty_gen_column);

void jornada_separada(score &sco_final, const vector<viajes> &viaje1, ...
    , double & costo_duty_gen_column);

void jornada_junta_partes(vector<score> & todos_score, const vector<bool>
    &viajes_fijos, ... ,const vector<double>&Precio_Sombrarestr);
```

```
void jornada_junta(vector<score> &scor_final,const int a, , ... , const
    vector<double>&Precio_Sombrarestr);

void camino_partido(score &scr, double &tiempo_total, ... , double &
    costo_duty_gen_column);

void jornada_partido_partes(vector<score> & todos_score, const vector<
    bool> &viajes_fijos, ... , const vector<double>&Precio_Sombrarestr);

 void jornada_partida(vector<score> &scor_final, const vector<bool> &
     viajes_fijos, ... , const vector<double>&Precio_Sombrarestr);

#endif // PROBLEMAMIP_HPP
```

4.2.8. Clase Problema MIP

Esta clase es la eje vertebral para la solución del problema AT-MBQ, pues aquí se crea el modelo. Esta clase contiene a las clase jornada, que nos permite encontrar una jornada laboral para cada tipo de tarea. Esta clase presenta las siguientes funciones:

add_vars y crear_restricciones Estas funciones nos permiten crear la instancia inicial del algoritmo, explicada en la Subsección 4.1.1.

anadir_variables Esta función permite crear todos los tipos de jornadas aceptables para buses de flota tipo 2. Para esto llama a las funciones de la clase *jornadas*. Si las jornadas laborales generadas tienen costo reducido negativo se llama a la función *anadir_columnas*, la cual se encarga de añadir la nueva columna en el programa RMLP-MBQ.

anadir_variables_flota1 Esta función permite crear todos los tipos de jornadas aceptables para buses de flota tipo 1. Para esto llama a las funciones de la clase *jornadas*. Si las jornadas laborales generadas tienen costo reducido negativo se llama a la función *anadir_columnas*, la cual se encarga de añadir la nueva columna en el programa RMLP-MBQ.

obtener_precios_duales Esta función nos permite obtener los precios duales de la actual solución de RDLP-MBQ.

solve Resuelve el problema RMLP; mientras, *solve_integer* es llamado para resolver el problema AT-MBQ.

fijando_variables Función que permite fijar variables de acuerdo al criterio de la Subsección 4.1.3.

modelo_entero Permite cambiar el tipo de las variables de continuo a binario.

uniendo_viajes_sueltos Permite unir los viajes que en la solución final de AT-MBQ quedan sueltos, a jornadas laborales de la solución. Esto se realiza mediante el criterio del menor score; se prueba que efecto tendría sobre el score de una jornada laboral si unimos el viaje a esta, siempre y cuando sea factible (en términos de terminal y de hora), y se une el viaje a la jornada donde se haya obtenido el menor score.

```
class MIP_Gurobi
{
private:

 GRBEnv* env_modelo;
 vector<GRBVar > _vars;
 GRBModel*  modelo_ip ;
 unsigned int n_restricc;
 unsigned int n_restricc_tipo_duty;
 unsigned int n_restricc_tipo_flota;
 vector<GRBConstr> _restrics;
 vector<GRBConstr> _restric_tipo_duty;
 vector<GRBConstr> _restric_tipo_flota;

public:
      MIP_Gurobi();
     ~MIP_Gurobi();

void add_vars(vector<double> &costos_iniciales);
void crear_restricciones();
```

```
void anadir_variables(vector <score> &Scr,const vector<bool> &
    viajes_fijos ,const vector<double> &Precio_Sombrarestr,int &contador,
    vector<viajes> & viaje, vector<vector<identificador> >&A, vector<int>&
    flota, vector<int>&conductor, const int & a);

void anadir_variables_flota1(vector <score> &Scr,const vector<bool> &
    viajes_fijos ,const vector<viajes> & viaje1 ,const vector<double> &
    Precio_Sombrarestr,int &contador, vector<viajes> & viaje, vector<
    vector<identificador> >&A,vector<int>&flota, vector<int>&conductor,
    const int & a);

void anadir_columnas(vector <score> &Scr, const score sco_final ,const
    vector <int>& ubi,int & contador, const vector <unsigned> ruta4, const
     double ruta4_costo,const double ruta4_costo_gen_column,vector<vector<
    identificador> >&A);

void solve(void);
void solve_integer(void);
void write_MIP(string nombre);
void obtener_precios_duales(vector<double> &Precio_Sombrarestr);

void fijando_variables( const vector<vector<identificador> >A,vector<bool
    > &viajes_fijos ,vector <unsigned int> & variables_fijadas);

void modelo_entero();

void uniendo_viajes_sueltos( vector <score> &Scr, vector<vector<
    identificador> > &A,const vector<viajes> &viaje, vector<int> &flota);
ostream& show_sol_integer(std::ostream & os,const vector <score> Scr,
    vector <score> &Scr_B, vector<vector<identificador> >&A,const vector<
    int>flota ,const vector<int>conductor, const vector<viajes> viaje ,
    vector<vector<identificador> > &B, vector<viajes> & viaje_mat ,
    unsigned int & numeros_viajes);
};
#endif
```

Capítulo 5

Resultados Computacionales

Para las pruebas computacionales se implementó el algoritmo de generación-fijación de columnas en el lenguaje de programación C++ usando la versión 5.5 de Gurobi como solver, en un computador con procesador dual Xeon E5504 con 13.6 GB de RAM y sistema operativo Centos 6.5. El Trolebús posee tres terminales: terminal Norte, terminal Sur y terminal Quitumbe. Cada viaje programado inicia y termina en una de estas terminales. Se conoce la hora de salida, la hora de llegada, la terminal de salida, la terminal de llegada y la duración de cada uno de los viajes. El sistema opera bajo una flota heterogénea, donde los vehículos de tipo 1 pueden cubrir los viajes entre las terminales Norte y Sur de la red, mientras los vehículos de tipo 2 pueden cubrir cualquier viaje.

Se usan cuatro tipos diferentes de programaciones diarias para las operaciones durante la semana: una de lunes a jueves, y una programación diferente para cada uno de los días restantes. Se considera cada programación como una instancia de prueba separada.

En las pruebas computacionales, todos los conductores están sujetos a las mismas reglas, el tiempo máximo de conducción es de 570 minutos (9 horas y media) divididas en dos piezas de trabajo de máximo 300 minutos (5 horas). Estas dos piezas de trabajo deben estar separadas por un descanso de al menos 15 minutos. Los cambios de conductor ocurren sólo en las terminales y no están permitidos durante una tarea. Por esto, si un conductor toma un descanso, entonces el vehículo debe estar estacionado. Finalmente, cada vehículo debe ser asignado a uno o dos

conductores, lo cual determina que un día de operación para un vehículo está compuesto por una tarea (continua o de horario partido) o por la secuencia de dos tareas continuas.

En la Tabla 5.1 se indican los viajes programados para cada instancia particionándolos de acuerdo al tipo de flota que los puede cubrir, también se muestran algunas estadísticas de la solución manual actualmente implementada por el operador del Trolebús, en particular en lo que respecta a los valores de score definidos en la Sección 3.1, al número de buses que se usa y al porcentaje de las jornadas que cumplen con las restricciones mencionadas en el párrafo anterior. Notar que en la solución actualmente implementada, sólo el 91 % de las jornadas laborales programadas en los días lunes a jueves cumplen con las restricciones especificadas. En los días viernes, donde la cantidad de viajes programados es la más alta, la cantidad de jornadas laborales que cumplen todos los requerimientos baja al 82 %.

	Viajes Programados				**Estadísticas de Operación**		
	día	**total**	**tipo 1**	**tipo 2**	**Score**	**Buses**	**Porcentaje**
Instancia 1	lunes-jueves	1.165	770	395	35.228,5	81	91 %
Instancia 2	viernes	1.464	984	480	54.430,5	103	82 %
Instancia 3	sábado	827	600	227	22.619,0	49	100 %
Instancia 4	domingo	767	540	227	20.219,5	46	100 %

Tabla 5.1: Características de las instancias de prueba y de la solución actualmente implementada por el operador del Sistema Trolebús.

Se realizaron 12 experimentos de prueba para cada instancia, modificando algunos parámetros que controlan la fijación de variables. En el esquema empleado, se fijan nuevas variables cuando el descenso relativo de la función objetivo z después de mínimo l iteraciones del algoritmo de generación de columnas es menor que un cierto ϵ, es decir cuando:

$$\frac{z_{a-l} - z_a}{z_{a-l}} < \epsilon \quad \text{y} \quad a - a_0 >= l.$$

Donde a representa a la iteración actual del algoritmo y a_0 es la iteración donde se realizó la última fijación de variables. z_a indica el valor de la función objetivo

en la iteración actual, mientras z_{a-l} nos indica el valor obtenido hace l iteraciones. La segunda condición asegura que hayan pasado al menos l iteraciones desde la última vez que se fijaron variables para volver a fijar variables. Para los experimentos se toma $l = 50$. Con esto, los parámetros que se modificaron durante las pruebas son:

- épsilon: se probaron valores de 0,05 o de 0,1,
- el número de variables que a fijar en cada ronda: se tomaron 1, 3 o 10 variables, y
- el criterio para seleccionar las variables a fijar; se probaron dos criterios: en el primero, se fijan las variables que posean el mayor valor en la solución actual ($\text{máx } \xi_j^\star$); en el segundo, se fijan las variables cuyo aumento en el costo dividido entre el número de viajes que cubren, sea el menor posible $\left(\text{mín } \frac{w_j(1-\xi_j^\star)}{n_j}\right)$.

Parámetros de Fijación de Variables			**Nombre del experimento**
Criterio	**N. de variables**	**Épsilon**	
$\text{máx } \xi_j^\star$	1	0,05	1a1b1c
$\text{máx } \xi_j^\star$	1	0,1	1a1b2c
$\text{máx } \xi_j^\star$	3	0,05	1a2b1c
$\text{máx } \xi_j^\star$	3	0,1	1a2b2c
$\text{máx } \xi_j^\star$	10	0,05	1a3b1c
$\text{máx } \xi_j^\star$	10	0,1	1a3b2c
$\text{mín } \frac{w_j(1-\xi_j^\star)}{n_j}$	1	0,05	2a1b1c
$\text{mín } \frac{w_j(1-\xi_j^\star)}{n_j}$	1	0,1	2a1b2c
$\text{mín } \frac{w_j(1-\xi_j^\star)}{n_j}$	3	0,05	2a2b1c
$\text{mín } \frac{w_j(1-\xi_j^\star)}{n_j}$	3	0,1	2a2b2c
$\text{mín } \frac{w_j(1-\xi_j^\star)}{n_j}$	10	0,05	2a3b1c
$\text{mín } \frac{w_j(1-\xi_j^\star)}{n_j}$	10	0,1	2a3b2c

Tabla 5.2: Parámetros de las pruebas computacionales.

Realizando todas las combinaciones de esos parámetros, resultan las 12 pruebas computacionales para cada instancia. En la Tabla 5.2 se muestran todas las combinaciones y el nombre con que será identificada cada prueba.

En las tablas de resultados: Tabla 5.3, Tabla 5.4, Tabla 5.5 y Tabla 5.6 se presentan los resultados de las pruebas computacionales para cada una de las instancias respectivamente. Para la primera iteración del algoritmo se necesitan definir ciertas variables iniciales que permitan que el problema $RMLP - MBQ$ sea factible. Estas variables iniciales son los trippers, variables que cubren exactamente un viaje. Los costos de las variables iniciales (trippers) se los determinó en 8.000, tomando un costo tan alto para evitar que en la solución final queden viajes sin ser cubiertos. El número de buses máximo que se puede usar fue relajado, en vista de que no se encontraron soluciones factibles que respeten el número de buses de cada tipo de flota a ser utilizado. Para reducir el número de trippers que quedaban en la solución, se tomó cada viaje al que representa cada tripper y se lo unió a la ruta compatible con el viaje cuyo score al incluir este viaje ya sea a su inicio o a su final sea el menor. Al unir los viajes que quedan sueltos en la solución a rutas obtenidas en la misma solución, las tareas que conforman las rutas no necesariamente respetarán las reglas establecidas (9 horas y media máximo de trabajo, dos supernodos de entre 3 y 5 horas). Sin embargo, en cada instancia se mostrará que la mejor solución obtenida al usar nuestro algoritmo cumple las reglas establecidas y a un menor score que la solución actual del Trolebús.

En la primera columna se encuentra el nombre de la prueba que ha sido efectuada, seguida del valor de la función objetivo (Score) obtenido, el número de buses y el número de conductores requeridos, el tiempo de CPU requerido, el número de trippers que han quedado antes de emplear la estrategia para unirlos a rutas (Trippers A) y cuántos quedan después de ser empleada la estrategia (Trippers D). Se debe tomar en cuenta que los valores de las columnas score, buses y conductores no consideran los trippers que quedan en la solución final (última columna).

Prueba	Score	Buses	Conductores	CPU	Trippers A	Trippers D
1a1b1c	80.452	94	119	04d 05h 47m	168	60
1a1b2c	71.296	94	118	03d 21h 18m	191	82
1a2b1c	15.156	96	140	02d 22h 44m	0	0
1a2b2c	18.904,5	92	141	02d 17h 02m	7	0
1a3b1c	24.984,5	100	147	02d 04h 05m	6	0
1a3b2c	27.219,5	94	150	01d 20h 58m	5	0
2a1b1c	58.947	97	125	04d 11h 12m	104	12
2a1b2c	64.173,5	95	122	04d 03h 08m	122	31
2a2b1c	21.011,5	93	138	03d 02h 12m	11	0
2a2b2c	20.486	96	140	02d 17h 39m	7	0
2a3b1c	28.132	101	144	02d 10h 43m	10	0
2a3b2c	23.384	96	143	02d 06h 29m	9	0

Tabla 5.3: Resultados del modelo para la instancia Lunes-Jueves.

En la Tabla 5.3 se presentan los resultados de las pruebas computacionales para la instancia lunes-jueves. Se observa que al fijar una sola variable (1b) se presentan los valores más altos en el score así como una gran cantidad de trippers. Por otro lado, se puede notar que seleccionar tres variables (2b) a fijar es la estrategia que presenta los mejores resultados, pues en estas pruebas se presentan los menores valores en la función objetivo. En algunas pruebas se obtiene menor valor en el score tomando épsilon como 0,05 (1c) y en otras con 0,10 (2c). El resultado con el menor score y sin dejar trippers se logra con el criterio de fijar las 3 variables (2b) con mayor valor en la función objetivo (1a) y tomando épsilon como 0,05 (1c). La estrategia para unir los trippers a rutas da buenos resultados, pues en la mayoría de pruebas al usar esta estrategia logramos cubrir todos los viajes con rutas.

En la Tabla 5.4 no se presentan los resultados de todas las pruebas computacionales del día viernes, debido a que el tiempo de ejecución de estas pruebas sobrepasó el tiempo límite (13 días). Al observar los resultados obtenidos, notamos que en algunas pruebas computacionales el score es menor al usar el

criterio para fijar las variables con el mayor valor en la función objetivo (1a) y en otras el criterio de menor costo ponderado (2a). El fijar 3 variables (2b) conduce a los menores scores de las pruebas computacionales. La estrategia 1a3b2c es la que muestra el mayor valor del score de toda la tabla, además de requerir el mayor número de buses y de conductores. El menor score de toda la tabla se logra fijando las 3 variables (2b) que tengan el mayor valor en la función objetivo (1a) y con épsilon de 0,05 (1c). La estrategia para unir los trippers a rutas da buenos resultados, pues en todas las pruebas al usar esta estrategia logramos cubrir todos los viajes con rutas.

Prueba	Score	Buses	Conductores	CPU	Trippers A	Trippers D
1a1b1c	-	-	-	00d 00h 00m	-	-
1a1b2c	-	-	-	00d 00h 00m	-	-
1a2b1c	22.047,50	127	187	07d 15h 58m	0	0
1a2b2c	22.680	128	186	07d 01h 51m	2	0
1a3b1c	43.654,50	130	192	05d 16h 30m	25	0
1a3b2c	49.455,50	131	193	05d 00h 09m	34	0
2a1b1c	-	-	-	00d 00h 00m	-	-
2a1b2c	-	-	-	00d 00h 00m	-	-
2a2b1c	24.361,50	128	186	06d 22h 36m	10	0
2a2b2c	30.984	128	186	06d 04h 11m	11	0
2a3b1c	30.664	123	191	05d 11h 04m	8	0
2a3b2c	39.575,50	127	191	04d 21h 18m	19	0

Tabla 5.4: Resultados del modelo para la instancia Viernes.

La Tabla 5.5 presenta los resultados de las pruebas computacionales para la instancia del día sábado. El menor score y el menor número de trippers se obtienen con la estrategia 1a1b2c. Los valores más altos en el score se dan cuando se fija 1 sola variable. En la mayoría de pruebas los scores de menor valor se presentan al utilizar un épsilon de 0,05. El criterio para fijar las variables que posean el mayor valor en la solución (1a) presenta en algunas pruebas un score mayor y en otras

pruebas menor score respecto al criterio de fijar las variables de menor costo ponderado (2a). La estrategia para unir los trippers a rutas da buenos resultados, pues en la mayoría de pruebas al usar esta estrategia logramos cubrir todos los viajes con rutas.

Prueba	Score	Buses	Conductores	CPU	Trippers A	Trippers D
1a1b1c	41.551	66	75	02d 05h 04m	143	93
1a1b2c	11.486,50	70	89	02d 18h 48m	0	0
1a2b1c	15.885,50	70	93	01d 11h 58m	1	0
1a2b2c	18.458	72	94	01d 09h 05m	8	0
1a3b1c	20.878	76	96	00d 23h 28m	5	0
1a3b2c	22.826,50	79	95	00d 19h 47m	12	0
2a1b1c	20.060	69	88	02d 01h 08m	23	1
2a1b2c	41.507,50	76	84	02d 00h 13m	65	8
2a2b1c	17.898,50	71	91	01d 13h 38m	13	0
2a2b2c	18.297,50	73	92	01d 07h 24m	10	0
2a3b1c	15.824,50	72	92	01d 05h 49m	7	0
2a3b2c	18.702	72	95	01d 01h 40m	9	0

Tabla 5.5: Resultados del modelo para la instancia Sábado.

La Tabla 5.6 presenta los resultados de las pruebas computacionales para la instancia del día domingo. Al usar un épsilon de 0,05 para fijar las variables se obtiene en todas las pruebas computacionales un menor valor en el score que con un épsilon de 0,10. El menor valor de score de toda la tabla se presenta al fijar las 3 variables (2b) con el mayor valor en la solución (1a) y con un épsilon de 0,05 (1c). El mayor valor de score de todas las pruebas computacionales se obtiene con la estratega 1a1b2c. Al observar los resultados obtenidos, notamos que en algunas pruebas computacionales el score es menor al usar el criterio para fijar las variables con el mayor valor en la función objetivo (1a) y en otras el criterio de menor costo ponderado (2a). La estrategia para unir los trippers a rutas da buenos resultados en la mayoría de pruebas computacionales.

Prueba	Score	Buses	Conductores	CPU	Trippers A	Trippers D
1a1b1c	17.456	61	82	02d 15h 15m	11	0
1a1b2c	34.336,50	62	71	01d 20h 32m	124	82
1a2b1c	12.379	63	86	01d 05h 06m	0	0
1a2b2c	14.983	64	86	01d 03h 01m	4	0
1a3b1c	19.708,50	69	88	00d 19h 29m	8	0
1a3b2c	20.270,50	72	90	00d 15h 31m	7	0
2a1b1c	30.156	62	78	01d 09h 31m	44	3
2a1b2c	21.810,50	61	81	01d 10h 01m	24	0
2a2b1c	21.174	66	82	01d 02h 29m	24	0
2a2b2c	26.877	68	84	00d 21h 56m	21	0
2a3b1c	15428,5	67	87	00d 20h 02m	6	0
2a3b2c	19.343	67	89	00d 16h 25m	10	0

Tabla 5.6: Resultados del modelo para la instancia Domingo.

En la Figura 5.1 se muestran los tiempos de CPU empleados (en segundos) en cada una de las doce pruebas computacionales respecto al número de viajes de cada instancia. Se puede notar que a medida que aumenta el número de viajes también se incrementa el tiempo computacional requerido para resolver el problema AT-MBQ. Se envidencia que hay una marcada diferencia en el tiempo computacional empleado de acuerdo al número de variables fijadas, ocupando el menor tiempo la estrategia de fijar 10 variables (triángulo), seguida por la de 3 variables (cuadrado) y requiriendo el mayor tiempo computacional la estrategia de fijar 1 variable (rombo). Para el criterio de fijar las variables con el mayor valor en la solución, se muestra que fijarlas con un épsilon de 0,05 (morado) requiere más tiempo computacional que fijarlas con un épsilon de 0,1 (verde), independientemente del número de variables fijadas. Para el criterio de fijar las variables con el menor costo ponderado en la solución se muestra que fijarlas con un épsilon de 0,05 (azul) requiere más tiempo computacional que fijarlas con un épsilon de 0,1 (tomate), independientemente del número de variables fijadas.

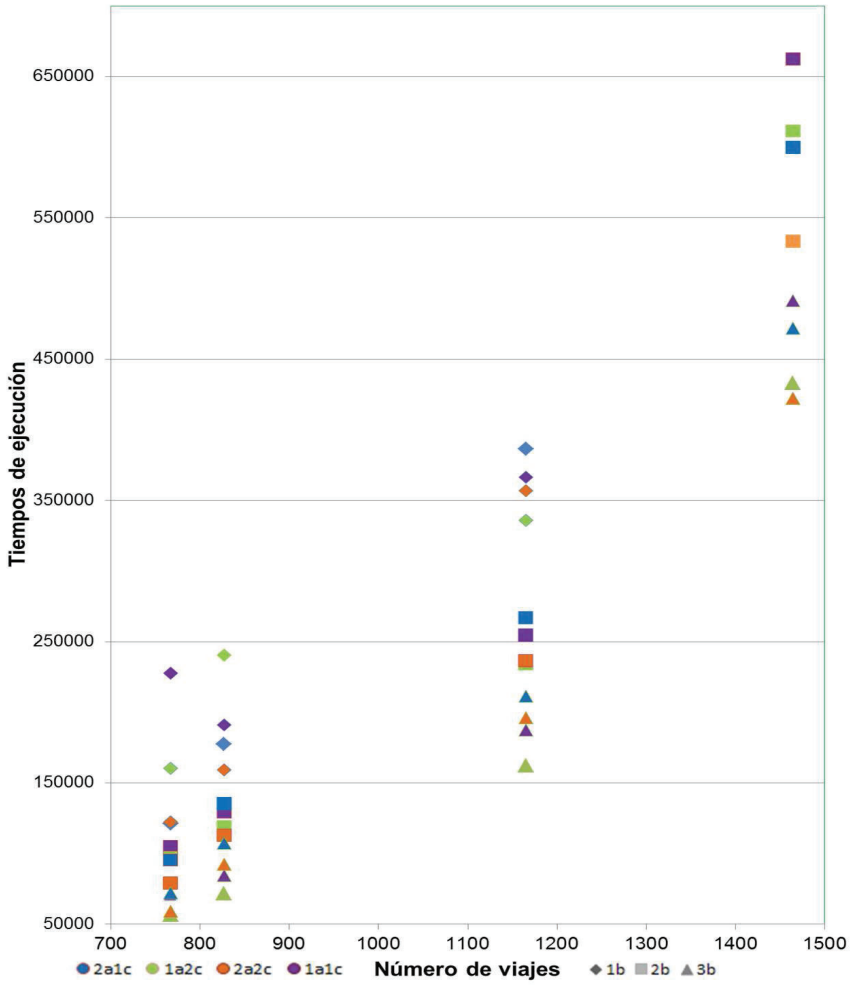

Figura 5.1: Tiempo de ejecución de los algoritmos en comparación con el número de viajes de cada instancia de prueba.

En la Tabla 5.7 se reportan los mejores resultados obtenidos para cada una de las cuatro instancias. Se presenta el score de la solución, el número de buses, el número de conductores, el tiempo de ejecución y el número de trippers obtenidos en la solución. El ahorro en términos del score total con respecto a la solución manual es significativo. Tomando como referencia la primera instancia (lunes-jueves), el score de la solución actual es de 35.228,5, frente al cual nuestro método presenta un ahorro del 57 %. El incremento en el número de buses se justifica por el hecho de que la solución actualmente implementada en el Trolebús no satisface las restricciones laborales para todas las jornadas, sólo lo hace en un 91 %. Todas las soluciones presentadas, respetan en su totalidad las reglas, además de que ningún

viaje queda sin ser cubierto.

Instancias	Score	Buses	Conductores	CPU	Trippers
Instancia 1	15.156	96	140	02d 22h 44m	0
Instancia 2	22.047,5	127	187	07d 15h 58m	0
Instancia 3	11.486,5	70	89	02d 18h 48m	0
Instancia 4	12.379	63	86	01d 05h 06m	0

Tabla 5.7: Mejores soluciones obtenidas por nuestro algoritmo en cada instancia.

En la Tabla 5.8 se muestra información más detallada de los mejores resultados obtenidos para cada una de las instancias. Con respecto a los vehículos utilizados, se reportan al lado izquierdo de la tabla el número de buses que son asignados a dos conductores (rutas compuestas por dos tareas), el número de buses que son asignados a un solo conductor y el número de buses de cada tipo de flota requerido. La parte correspondiente a los conductores se encuentra al lado derecho de la tabla, donde se reportan el número de conductores para los diferentes tipos de tareas, las mismas que de acuerdo a su estructura son las siguientes:

- Horario-partido(HP): poseen dos subtareas, una en la mañana y otra en la tarde.
- Continuas(C): poseen dos subtareas con una separación máxima entre ellas de dos horas.

	Vehículos				Conductores	
	1 conductor	2 conductores	flota I	flota II	H-P	C
Instancia 1	52	44	30	66	22	118
Instancia 2	67	60	51	76	22	165
Instancia 3	51	19	12	58	37	52
Instancia 4	40	23	17	46	11	75

Tabla 5.8: Descripción detallada de los mejores resultados.

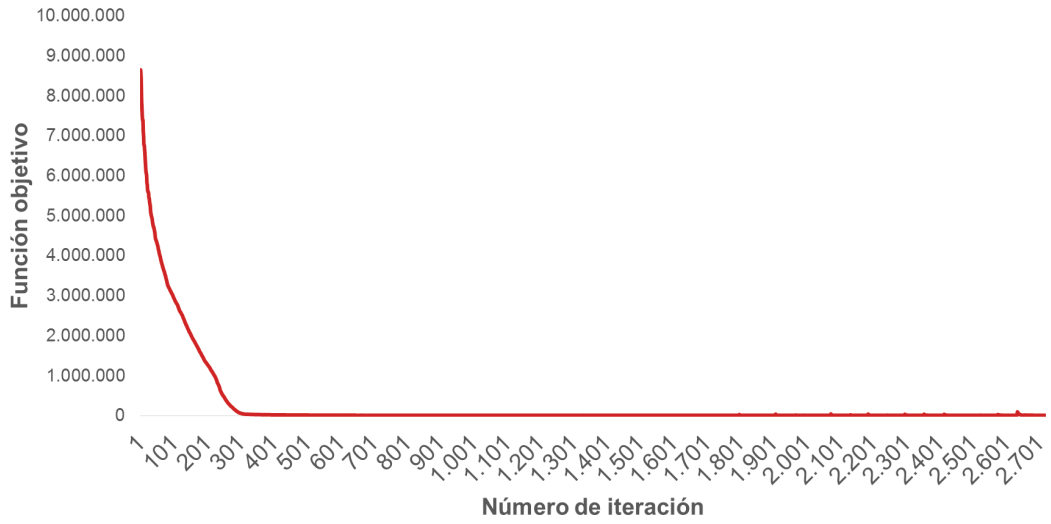

Figura 5.2: Valor de la función objetivo respecto al número de iteraciones para la instancia lunes-jueves.

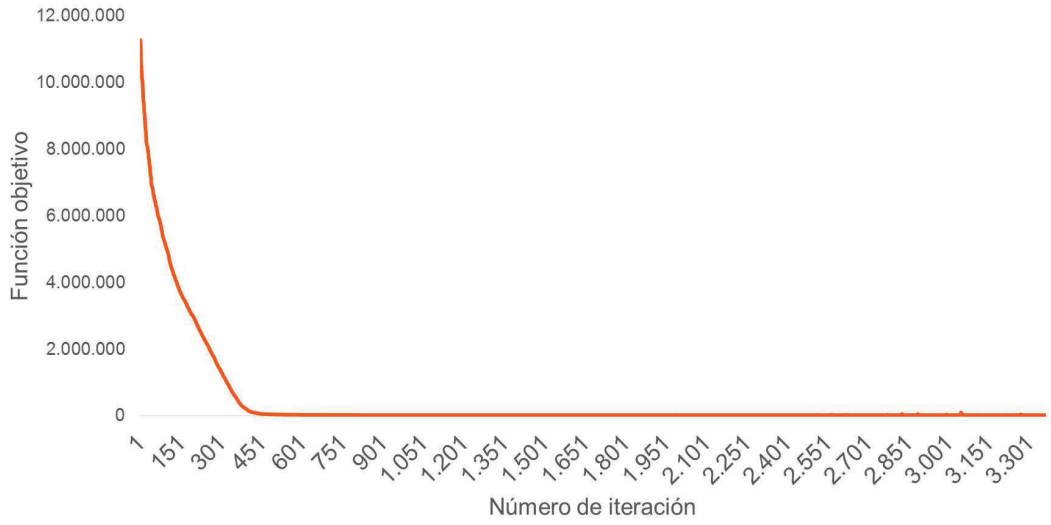

Figura 5.3: Valor de la función objetivo respecto al número de iteraciones para la instancia del día viernes.

Las Figuras 5.2, 5.3, 5.4 y 5.5, indican la manera en que cambia el valor de la función objetivo de acuerdo al número de iteraciones del algoritmo, para cada unas de las instancias. En todas las figuras se observa que aproximadamente en la iteración 300 el valor de la función objetivo se estabiliza y es ahí donde empieza la fijación de variables. Se evidencia el cambio que produce la fijación de variables

en el valor de la función objetivo especialmente en la últimas iteraciones.

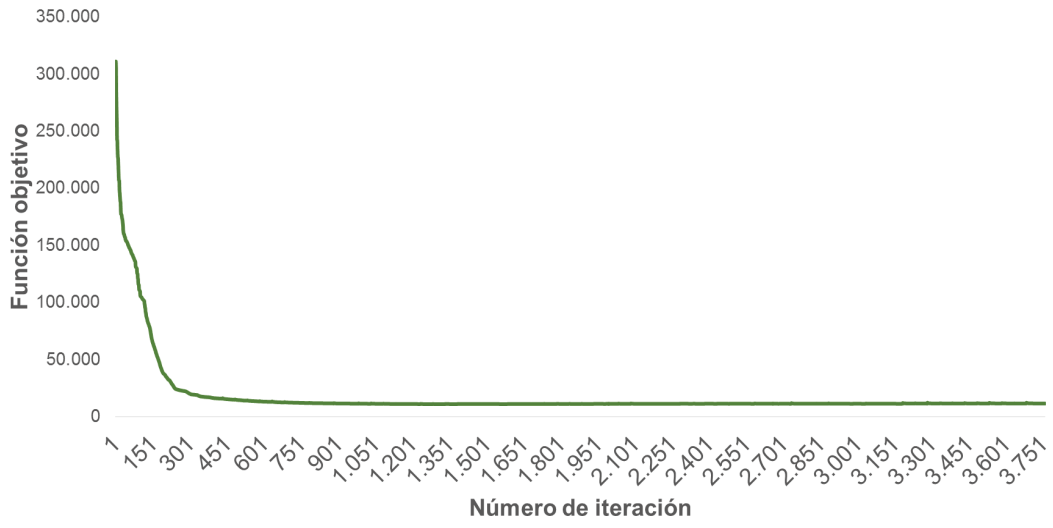

Figura 5.4: Valor de la función objetivo respecto al número de iteraciones para la instancia del día sábado.

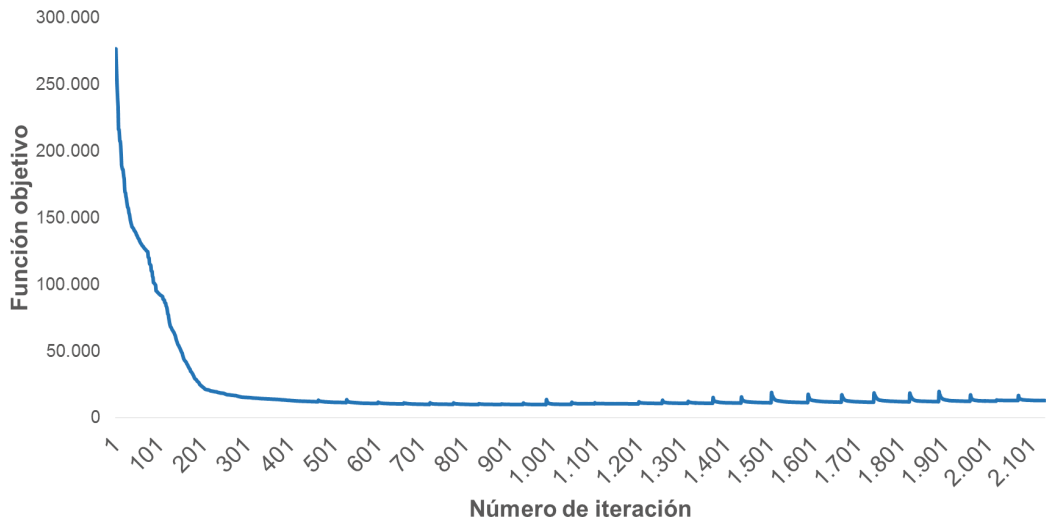

Figura 5.5: Valor de la función objetivo respecto al número de iteraciones para la instancia del día domingo.

En la Figura 5.6 se muestra un acercamiento de las últimas iteraciones del algoritmo para la instancia del día viernes. En esta figura se observa claramente el cambio que produce en el valor de la función objetivo la fijación de variables. Cada vez que se fijan variables, se obliga a que las variables fijadas tomen el valor de 1 en la solución

en lugar de su valor original (un número real entre 0 y 1). Esta modificación provoca que el valor de la función objetivo aumenta bruscamente. Luego de la fijación el valor de la función objetivo vuelve a decrecer debido a que es un problema de minimización, hasta que se vuelvan a fijar variables.

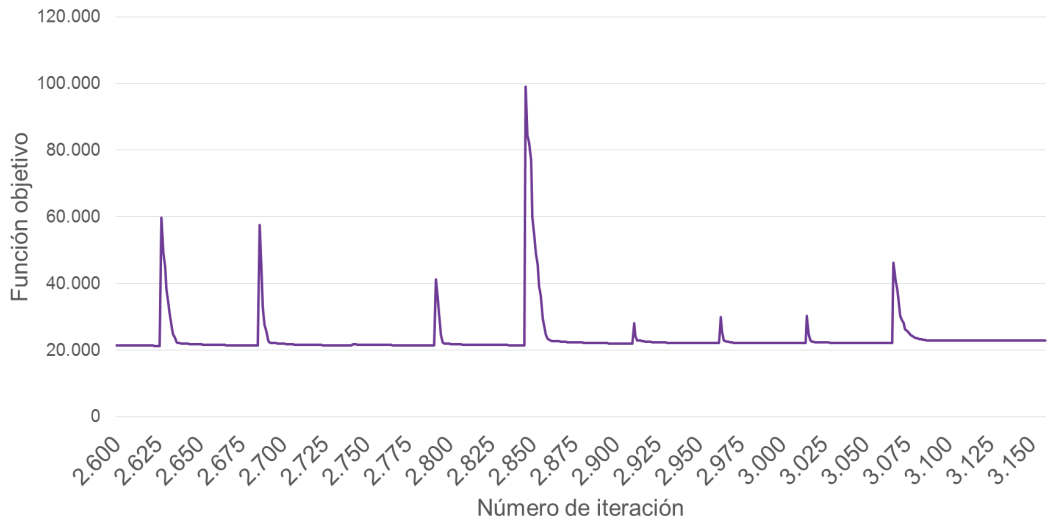

Figura 5.6: Valor de la función objetivo respecto a las últimas iteraciones del algoritmo para la instancia del día viernes.

Capítulo 6

Conclusiones

- El problema de asignación de tareas de conducción en el sistema Trolebús, puede ser formulado como un problema de particionamiento añadiendo restricciones respecto al número de trabajadores y al número de buses disponibles. Dado que existe una gran cantidad de combinaciones de viajes con los que se pueden generar una jornada laboral, es prácticamente imposible escribir todas las columnas (variables del problema) explícitamente en el programa lineal, por lo cual se hace necesario utilizar el método de generación de columnas. El subproblema de costeo que resulta se lo puede modelar como un problema diferente de camino más corto con dos restricciones de recursos (tiempo y descanso) para cada tipo de tarea, el recurso tiempo indica el tiempo transcurrido desde el inicio de la jornada y el recurso descanso es utilizado para que los conductores pueden tomar un descanso después de un determinado tiempo de conducción.

- El algoritmo que se utiliza para resolver el problema de camino más corto con restricciones de recursos es el algoritmo de *fijación de etiquetas*; sin embargo, el concepto de etiquetas eficientes no es aplicable a nuestro caso de estudio. Para enfrentar este inconveniente se modifica el algoritmo de solución. Se encuentra el camino más corto con restricción de recursos mediante el algoritmo de fijación de etiquetas, para cada uno de los posibles nodos iniciales. Los nodos iniciales se fijan de acuerdo a un rango de tiempo, pues al ser los viajes con los que inicia una jornada laboral se puede tener

una estimación del rango apropiado, por ejemplo las jornadas de la mañana pueden tener como nodos iniciales todos los viajes cuya hora de salida se encuentre entre las 5 y las 7 de la mañana. Cada nodo poseerá a lo sumo dos etiquetas, una que contenga en el campo del recurso descanso el valor 0 y otra el valor de 1, al ser el recurso tiempo igual para todas las etiquetas del mismo nodo, sólo se comparará el costo para determinar si una etiqueta domina o no a otra. Una vez que se han encontrado todos los caminos más cortos para cada uno de los posibles viajes iniciales, se elige el camino de mínimo costo de entre todos los caminos obtenidos en el paso anterior.

- La fijación de variables se hizo necesaria pues, al realizar pruebas computacionales sin esta técnica para la primera instancia, quedaban alrededor del 25 % de viajes como trippers al resolver el programa entero debido a que en la solución del programa lineal, los valores de las variables en la función objetivo eran demasiado pequeños (incluso menor a 0,05), además de que el tiempo computacional era de alrededor de 13 días.

- Al comparar los resultados obtenidos para los diferentes valores de épsilon (1c=0,05 y 2c=0,10), se nota que en algunos casos se obtienen mejores resultados con un valor de 0,05 y en otros con 0,10. Esto se debe a que al utilizar el valor de 0,10 puede que se estén fijando columnas que no son necesariamente buenas (pues la función objetivo aún no se estabiliza y no se cuenta con un número suficiente de columnas para realizar la fijación de variables) ya que el proceso de fijado resulta ser muy rápido; por otro lado, al usar el valor de 0,05 puede que se esté dando lo contrario, que se estén fijando columnas buenas pero que el proceso de fijado sea muy lento que al terminar el algoritmo no se han fijado suficientes variables.

- Al usar la estrategia de fijar una sola variable (1b) se obtienen los valores de la función objetivo más altos de todas las pruebas computacionales, además de que es la estrategia donde quedan más trippers en la solución. Al fijar solamente una variable cada determinado tiempo, cuando termina el algoritmo no se habrán fijado suficientes variables y por tanto en la función objetivo la mayoría de variables no fijadas tomarán valores muy pequeños, esto provoca

que al resolver el programa entero queden varios trippers. Al unir los trippers que quedan, a las jornadas laborales obtenidas en la solución, por la manera en que se define el score el valor de la función objetivo es alto.

- Al fijar diez variables (3b) cada determinado tiempo, resulta que se fijan muchas variables de manera muy rápida, provocando que las columnas que hemos fijado no sean de buena calidad (que tengan un alto score) y al mismo tiempo dejando viajes que ya no se puedan concatenar de manera adecuada para formar una jornada laboral factible.

- El uso de la estrategia de fijar tres variables (2b) cada determinado tiempo, es la estrategia que mejores resultados arrojó, con esta estrategia se tiene un equilibrio en el sentido de que al finalizar la generación de columnas, se han fijado suficientes variables y de buena calidad, lo que hace que el valor de la función objetivo sea el más bajo al usar esta estrategia.

- De manera general se observa que la estretegia para unir los trippers de la solución final a jornadas laborales da buenos resultados, pues en la mayoría de pruebas logra cubrir todos los viajes.

- En todas las pruebas computacionales la estrategia que presentó mejores soluciones en la mayoría de instancias, fue la de fijar 3 variables que presenten el mayor valor en la función objetivo ($\xi_j^\star$), usando un épsilon de 0,05.

- El ahorro de nuestras soluciones en términos del score total es significativo. Tomando como referencia la primera instancia (lunes-jueves), el score de la solución actual es de 35.228,5, la cual comparada con nuestro método (15.156) presenta un ahorro del 57 %. Los ahorros en los scores de las instancias viernes, sábado y domingo son del 59 %, 49 % y 39 %, respectivamente.

- El incremento en el número de buses de nuestras soluciones para todas las instancias, se justifica por el hecho de que todas estas soluciones presentan un menor score que las soluciones actuales, además de que la solución manual del Trolebús no respeta en su totalidad las restricciones laborales,

sólo lo hacen en un 91 % para la instancia de lunes-jueves y en un 82 % para la instancia del viernes. Todas nuestras soluciones presentadas, respetan en su totalidad las reglas, además de que ningún viaje queda sin ser cubierto.

- La formulación e implementación de un modelo de programación lineal entera permitió obtener una asignación de tareas de conducción que cubre todos los viajes programados del Corredor Central del Sistema Metrobús-Q (Trolebús) satisfaciendo las restricciones laborales al menor score.

- Una de las desventajas que presenta nuestro algoritmo es el incremento en el tiempo de ejecución a medida que aumenta el número de viajes, pasando de 1,21 días con 767 viajes a 7,66 días con 1.464 viajes. Sin embargo, cabe recalcar que la asignación de tareas de conducción dentro de la planificación operativa es una operación que se realiza en el Trolebús a lo sumo una vez al año, por lo cual el tiempo de ejecución de nuestro algoritmo resulta razonable.

Bibliografía

Yash P. Aneja, V. Aggarwal, and K. P. K. Nair. Shortest chain subject to side constraints. *Networks*, 13(2):295–302, 1983.

Augusto Barrera. Informe del cuarto año de gestión, agosto 2009 a julio 2013. *Questiones Urbano Regionales*, 2(2):123–219, 2013.

Bellman. On a routing problem. *Quarterly of Applied Mathematics*, 16:87–90, 1958.

Ralf Borndörfer. Discrete optimization in public transportation. ZIB-Report 08-56, february 2008.

Ralf Borndörfer, Martin Grötschel, and Andreas Löbel. Scheduling duties by adaptive column generation. ZIB-Report, 2001.

Alberto Caprara, Matteo Fischetti, and Paolo Toth. A heuristic method for the set covering problem. *Operations Research*, 47(5):730–743, 1999.

Chu and Beasley. A genetic algorithm for the set partitioning problem. *Imperial College*, april 1995.

Denardo and Fox. Shortest-route methods: 1. Reaching, pruning, and buckets. *Operations Research*, 27(1):161–186, 1979.

Guy Desaulniers and Mark D. Hickman. Public transit. In C. Barnhart and G. Laporte, editors, *Handbook in OR and MS*, volume 14, chapter 2. 2007.

Desrochers. An algorithm for the shortest path problem with resource constraints. Technical report, Centre de recherche sur les transports, Montréal, 1986.

Desrochers and Soumis. A generalized permanent labelling algorithm for the shortest path problem with time windows. *INFOR*, 26:191–212, 1988.

Martin Desrochers and François Soumis. A column generation approach to the urban transit crew scheduling problem. *Transportation Science*, 23(1):1–13, 1989.

Pelletier Desrosiers and Soumis. Plus court chemin avec contraintes d'horaires. *Rairo. Inf. Théor.*, 17:357–377, 1983.

Empresa Metropolitana de Transporte de Pasajeros. Encuesta domiciliaria de movilidad (EDM11) del distrito metropolitano de quito. Technical report, marzo 2012a.

Empresa Metropolitana de Transporte de Pasajeros. Informe de gestión. Technical report, 2012b.

D. Villeneuve G. Desaulniers. The shortest path problem with time windows and linear waiting costs. *Transportation Science*, 34(3):312–319, 2000.

Gilmore and Gomory. A linear programming approach to the Cutting-Stock problem. *Operations Research*, 9(6):849–859, 1961.

Gilmore and Gomory. A linear programming Approach to the Cutting-Stock problem part II. *Operations Research*, 11(6):863–868, 1963.

Tal Grossmana and Avishai Wool. Computational experience with approximation algorithms for the set covering problem. *European Journal of Operational Research*, 101(1):81–92, 1997.

Gurobi Optimization. *Gurobi Optimizer Reference Manual*. Gurobi Optimization, Inc, 6 edition, 2014.

Mark Hickman, Pitu Mirchandani, and Stefan Voss, editors. *Computer-Aided Scheduling of Public Transport*. 600. Springer-Verlag Berlin Heidelberg, 1 edition, 2008.

Helena Lourenço, José Paixão, and Rita Portugal. Multiobjective metaheuristics for the bus driver scheduling problem. *Transportation Science*, 35(3):331–343, 2001.

Silvano Martello and Paolo Toth. *Knapsack problems: Algorithms and Computer Implementations*. Wiley, New York, NY, USA, 1990.

Municipio del Distrito Metropolitano de Quito. Plan de desarrollo 2012 – 2022. Technical report, diciembre 2011.

Daniel Potthoff, Dennis Huisman, and Guy Desaulniers. Column generation with dynamic duty selection for railway crew rescheduling. *Transportation Science*, 44 (4):493–505, 2010.

Luis Miguel Torres. *Online Vehicle Routing- Set Partitioning Problems*. PhD thesis, Technischen Universität Berlin, December 2003.

Trolebús. Empresa Pública Metropolitana de Transporte de Pasajeros. Accedido en 30-04-2015. URL `http://www.trolebus.gob.ec/`.

Stefan Voss and Joachim Daduna, editors. *Computer-Aided Scheduling of Public Transport*. 505. Springer-Verlag Berlin Heidelberg, 1 edition, 2001.

Weider. *Integration on Vehicle and Duty Scheduling in Public Transport*. PhD thesis, Technischen Universität Berlin, 2007.