UNIVERSIDAD DEL VALLE SEDE TULUÁ

PROTOTIPO DE SOFTWARE PARA LA OPTIMIZACIÓN DE RUTAS DE TRANSPORTE PÚBLICO USANDO PROGRAMACIÓN POR RESTRICCIONES

por

Julián Felipe Bonilla Gómez, Andrés Felipe Lince Pineda

201056276, 201056278

Facultad de Ingeniería

Escuela de Ingenería en Sistemas y Computación

16 de febrero de 2016

UNIVERSIDAD DEL VALLE SEDE TULUÁ

PROTOTIPO DE SOFTWARE PARA LA OPTIMIZACIÓN DE RUTAS DE TRANSPORTE PÚBLICO USANDO PROGRAMACIÓN POR RESTRICCIONES

por

Julián Felipe Bonilla Gómez, Andrés Felipe Lince Pineda
201056276, 201056278

Director:

Pável Franco Marín, MAE.

Trabajo de grado para optar por el título de
Ingeniero de Sistemas

en la

Facultad de Ingeniería
Escuela de Ingenería en Sistemas y Computación

16 de febrero de 2016

UNIVERSIDAD DEL VALLE SEDE TULUÁ

Abstract

Facultad de Ingeniería
Escuela de Ingenería en Sistemas y Computación

Ingeniero de Sistemas

por Julián Felipe Bonilla Gómez, Andrés Felipe Lince Pineda

El diseño de redes de transporte (**TND**, Transit Network Design) es un problema de optimización combinatoria complejo que busca obtener un sistema de transporte público masivo optimizado [1]. En este proyecto de grado se modela y construye una aplicación prototipo para el diseño de rutas y frecuencias de transporte público masivo mediante el empleo de técnicas de programación por restricciones, paradigma que ha sido poco usado en la búsqueda de soluciones para TND.

Mediante un proceso de investigación con la secretaría de Tránsito del municipio de Tuluá [2] se desarrolla el prototipo buscando asemejarse a las características y realidades del municipio usando datos reales de las necesidades de transporte de la población, ajustando el modelo seleccionado a las condiciones de la zona, y buscando presentar los resultados de una forma agradable al usuario.

Este proyecto busca generar un primer eslabón para el desarrollo de una herramienta que permita optimizar todos los aspectos del transporte público masivo, teniendo en cuenta las necesidades de usuarios y operadores, características de los sistemas más complejos y condiciones de la prestación del servicio en el área objetivo.

Palabras clave: *Transporte público masivo, programación por restricciones.*

Índice general

Índice de figuras

Índice de cuadros

Abreviaciones

TND	**T**ransit **N**etwork **D**esign
CSP	**C**onstraint **S**atisfaction **P**roblems
JSON	**J**ava**S**cript **O**rdinary **O**bject

Capítulo 1

Introducción

El objetivo de este capítulo es brindar una descripción del problema que aborda este trabajo de grado, justificar su solución y establecer los objetivos generales y específicos que se cumplieron para su solución.

1.1. Descripción del problema

Según el Ministerio de Transporte de Colombia "el transporte público es una industria encaminada a garantizar la movilización de personas o cosas, por medio de vehículos apropiados, en condiciones de libertad de acceso, calidad y seguridad de los usuarios y sujeto a una contraprestación económica". [3]

El transporte permite a la gente hacer parte de las actividades sociales y nace de la necesidad de movilizar una gran cantidad de personas en un número determinado de vehículos –número menor que la cantidad creciente de vehículos particulares–. Este se compone de elementos como la flota de unidades de transporte de las empresas prestadoras del servicio, el conjunto de rutas, la infraestructura vial y finalmente la calidad del servicio, elementos que deben comportarse acorde a la demanda para que el sistema sea eficiente[4], mientras que en términos económicos, posibilita el desplazamiento masivo incrementando las economías de escala y en general, la productividad de una ciudad. [5]

La sociedad crece cada vez más y las ciudades tienden a expandirse en términos geográficos y de infraestructura, por lo que la demanda del transporte urbano incrementa. Esta situación agrega complejidad a cualquier sistema de transporte y genera problemas como contaminación, dificultad para estacionarse, congestión vehicular y accidentes que se hacen más frecuentes. Los problemas de movilidad se acentúan en los centros de las ciudades y se debe a que la mayor parte de las actividades urbanas se realizan allí. [6]

En el diseño de un sistema de red de transporte eficiente se deben tener en cuenta todos los factores que interactúan en éste, por ejemplo: usuarios, estaciones, trayectos, recursos, etc. Debido al crecimiento de las ciudades realizar viajes de un punto a otro mediante transporte particular se vuelve complejo, especialmente por la congestión que se genera. Este es el momento en que el transporte público cobra más relevancia, ya que su capacidad es mayor que la de los automóviles particulares porque se utilizan menos vehículos transportando a más personas, contribuyendo a mejorar el flujo del tráfico en cualquier ciudad[7]. Adicional a esto, la mayor parte de la población de las ciudades, está comprendida entre los niveles socio-económicos medio y bajo, en los cuales no siempre se puede acceder a un tipo de transporte particular, siendo el transporte público la opción elegida.[4]

Con base en lo anterior, los gobiernos locales se ven enfrentados a realizar una planeación adecuada sobre sus sistemas de transporte público con el fin de asegurar el flujo diario de los ciudadanos y mitigar los problemas mencionados antes. Esta planeación debe optimizar el rendimiento del sistema de transporte público para beneficio de la ciudadanía, haciendo el mejor uso posible de los recursos disponibles para ello, incentivando el uso de este medio de transporte para la aceptación y percepción de los ciudadanos.[7]

En consecuencia, el escenario descrito permite inferir una situación en la que se ven involucrados diversos intereses, pues las necesidades de los usuarios entran en conflicto con las de los operadoras de transporte porque los usuarios buscan tiempos de viajes cortos, seguridad y bajos precios, mientras que los operadores buscan beneficios económicos, rendimiento y mínimo riesgo, siendo la búsqueda del equilibrio entre ambas partes el objetivo de la planeación de los sistemas de transporte público.[8]

Por lo planteado anteriormente, se han realizado diversas investigaciones desde diferentes campos del conocimiento para intentar encontrar soluciones óptimas del problema de diseño de redes de tránsito, pero por su complejidad no se ha evidenciado una solución satisfactoria en la literatura consultada. Por esto, muchas agencias de tránsito recurren a su experiencia para diseñar sus sistemas de transporte público, a lo sumo usando herramientas de simulación por computador para evaluar configuraciones distintas. De esta manera los resultados pueden verse afectados por la subjetividad de los diseñadores del sistema, o pueden no ser los más óptimos por la complejidad y la tardanza de buscar todas las soluciones manualmente[2]. Por esto es importante seguir investigando en el área y explorar a través de varias técnicas para encontrar solución a este problema de optimización, aprovechando la capacidad creciente de procesamiento computacional con la que se cuenta en la actualidad.

1.2. Formulación del problema

Después de este análisis se plantearon las siguientes preguntas:

1.2.1. Pregunta principal de investigación

¿Cuál sería una herramienta adecuada para los gobiernos locales en relación a la planificación de sus sistemas de transporte público?

1.2.2. Preguntas secundarias de investigación

¿Cuáles deberían ser las variables a tener en cuenta para planear la dinámica de un sistema de transporte público?

¿Cuál es la mejor manera de integrar las variables inherentes al sistema de transporte público?

¿Cómo garantizar que la solución propuesta es adecuada para para los gobiernos locales en relación a la planeación de sus sistemas de transporte público?

1.3. Objetivos

1.3.1. Objetivo general

Construir un modelo de red de buses de transporte público en la ciudad de Tuluá, buscando instancias de rutas mejores mediante la programación por restricciones.

1.3.2. Objetivos específicos

1. Investigar un modelo sobre diseño de red de transporte basado en satisfacción de restricciones.

2. Implementar el modelo escogido para la ciudad de Tuluá identificando datos reales y diseñando rutas para ser evaluadas.

3. Construir una herramienta prototipo mediante programación por restricciones e identificar instancias óptimas que cumplan con las condiciones establecidas en el modelo.

4. Interpretar los resultados de las ejecuciones y proponer acciones a tomar para mejorar el sistema de buses de transporte público en Tuluá.

1.4. Justificación

El transporte es indispensable en todas las civilizaciones, culturas y sociedades. La manera en que una persona puede desplazarse desde su ubicación actual a un lugar deseado, en donde pueden ejecutar sus actividades para el sustento básico y desarrollo de sus actividades económicas, es lo que hace crecer a la industria y mover los mercados. Existen diferentes medios de transporte y se adaptan a las necesidades de las personas.

Actualmente la mayor parte de la población mundial se concentra en las ciudades principales provocando su expansión y las distancias entre dos puntos dados dentro del área urbana se hacen cada vez más grandes. La congestión por parte de los automóviles y otros modos de transporte particulares, los costos de acceder a ellos, la contaminación generada por la gran cantidad de gases vehiculares, entre otros, permiten que el transporte público masivo se convierta en un medio aceptado por la mayoría de la población, y a su vez, sea amigable con el ambiente.

El problema conocido como **TNDP** (*Transit Network Design Problem* o *Problema de Diseño de Rutas de Transporte*) consiste en hallar un conjunto de recorridos y valores de frecuencias, optimizando los objetivos de usuarios (pasajeros) y operadores (empresas de transporte), con base en información geográfica y de demanda.

La resolución del TNDP tiene sentido en el contexto de planificaciones estratégicas, donde existe una autoridad reguladora, que actúa sobre determinados componentes del sistema de transporte público, en particular los trazados de los recorridos y los valores de las frecuencias. Los objetivos de usuarios y operadores son contrapuestos, por lo que en contextos donde existe regulación, es responsabilidad de las autoridades determinar un nivel de compromiso adecuado [9].

Los objetivos de las investigaciones realizadas en este campo apuntan hacia soluciones óptimas que generen beneficios para los usuarios (siendo usuarios potenciales la ciudadanía en general), para las empresas prestadoras del servicio de transporte, y finalmente, a la ciudad en su totalidad.

En este proyecto se busca usar el conocimiento generado por múltiples investigaciones en el campo para desarrollar una aplicación prototipo que genere rutas óptimas, frecuencias para un área geográfica y una demanda de transporte dada. Se pretende hacer uso de la capacidad computacional con la que se cuenta hoy en día para brindar propuestas de solución automáticas a los responsables de la planificación de tránsito, como apoyo a la toma de decisiones.

El prototipo desarrollado se basa en un modelo de satisfacción de restricciones propuesto en otra investigación, y se espera que pueda ser usado en trabajos futuros para ser

complementado en miras a un enfoque global del problema de la planificación de tránsito. Debido a las múltiples variables y restricciones del problema, la programación por restricciones surge como un enfoque interesante por su capacidad de tratar con éxito problemas de optimización combinatoria, y por la ventaja que ofrece para representar el problema de forma más natural, como lo es un modelo de satisfacción de restricciones.

Los autores de este proyecto consideran, basados en sus experiencias personales, que algunos sistemas de transporte público existentes no son totalmente eficientes, presentando situaciones como largos tiempos de viaje, tiempos de espera altos para abordar un bus, falta de organización en los horarios de las rutas, entre otros. Proponen el paradigma de programación por restricciones porque es una técnica capaz de tratar con problemas de complejidad importantes, por su simplicidad a la hora de representar el modelo del problema y por su flexibilidad; consideran, además, que el trabajo con herramientas que usen este paradigma aportará información valiosa para su formación profesional.

1.5. Metodología

El ciclo de vida del proyecto se dividió en cuatro fases: análisis, modelación, construcción y evaluación. Al finalizar la etapa de análisis se obtuvieron las variables requeridas para la modelación del problema. El resultado de la etapa de modelación fue un modelo de satisfacción de restricciones amoldado a la zona de estudio. El resultado de la etapa de construcción fue la herramienta prototipo desarrollada en su totalidad. Finalmente en la etapa de evaluación se recogieron conclusiones importantes sobre el desarrollo del proyecto y se hicieron recomendaciones finales inherentes a los hallazgos. Se elaboró la documentación necesaria de la implementación a la par con el desarrollo. Se debe tener en cuenta que cada fase representa un objetivo específico del proyecto.

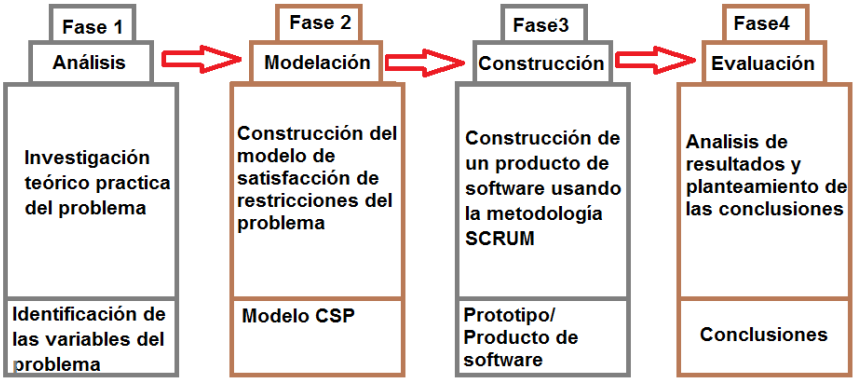


FIGURA 1.1: Etapas del desarrollo del proyecto

Teniendo en cuenta la figura anterior, a continuación se presentan cuadros correspondientes a cada fase en cuestión, donde se plasman de manera detallada las actividades desarrolladas.

Ítem	Actividad
1	Revisión del estado del arte.
2	Investigación teórica sobre el problema y los diferentes enfoques planteados para su solución.
3	Investigación de campo para buscar una empresa de transporte público local que nos provea de variables reales para el problema.

CUADRO 1.1: Actividades de la etapa de análisis

Ítem	**Actividad**
1	Investigación sobre la creación el modelamiento de un CSP (Constraint Satisfaction Problems).
2	Análisis de la tipología de las variables, sus dominios finitos o infinitos, y tipos de restricciones en el problema.
3	Planteamiento de un modelo de restricciones adecuado para la especificación del problema y capaz de captar las distintas tipologías de restricciones que puedan aparecer.

Cuadro 1.2: Actividades de la etapa de modelación

Ítem	**Actividad**
1	Determinar las herramientas a usar en la construcción de la aplicación.
2	Desarrollo de la aplicación utilizando la metodología SCRUM.
3	Pruebas a la aplicación (integradas en el desarrollo).

Cuadro 1.3: Actividades de la etapa de construcción

Ítem	**Actividad**
1	Implementación de la aplicación..
2	Recoger concepto de una autoridad local en transporte público.
3	Plantear conclusiones sobre la aplicación desarrollada y sus resultados mostrados
4	Elaborar conclusiones
5	Revisar el informe final del proyecto de grado

Cuadro 1.4: Actividades de la etapa de evaluación

Capítulo 2

Marco referencial

El transporte mueve a la gente y a los bienes materiales de un punto a otro usando una variedad de vehículos a través de diferentes sistemas de infraestructura. Esto requiere de tecnología -vehículos, energía, infraestructura- además del tiempo y esfuerzo de la gente, lo que genera además de los viajes de pasajeros y carga inconvenientes como contaminación, ruido, congestión y accidentes. El transporte público se define como un modo de transporte que mueve múltiples personas de un punto a otro usando un vehículo común [10]. Existen diferentes vehículos que pueden prestar este servicio, pero para efectos del caso solo se hablará de buses.

El proceso de generar un sistema de transporte público ha sido abordado desde distintas áreas, debido a que el desarrollo de este modo de transporte es crucial en la sociedad moderna. Los usuarios pueden optar usar el transporte público para evadir los problemas mencionados anteriormente si el sistema dispuesto es de calidad, accesible y satisface sus necesidades. La tarea de las agencias de tránsito es brindarle a la sociedad un sistema de transporte público que cumpla con las necesidades de los usuarios, manteniendo a la vez el menor costo posible. Por esto, esta situación es vista desde varios puntos de vista como un problema de optimización de varios objetivos. [8]

A continuación se hablará sobre el objetivo de la planeación de redes de transporte público y la programación por restricciones.

2.1. Planeación de transporte público

El proceso de planeación de transporte público es dividido en cinco etapas:

- Diseño de rutas.
- Ajuste de frecuencias.
- Horarios de servicio.
- Programación de vehículos.
- Programación de conductores. [8]

La primera etapa es llamada planeación estratégica, la segunda y tercera etapa son llamadas planeación táctica, y la cuarta y quinta etapa entran en la planeación operacional [6]. El presente trabajo se centrará en la primera etapa.

Toda la información requerida por los usuarios, como lo es el conjunto de rutas de tránsito, las frecuencias y los tiempos de salida, se generan en las primeras tres etapas. Podría pensarse que esta situación va enfocada a los usuarios. Sin embargo, las agencias de tránsito buscan obtener beneficios económicos minimizando los costos, por lo que los intereses de ambas partes entran en conflicto. [8]

La información de entrada en el proceso de planeación de transporte público está basada en un área geográfica con características topológicas, una demanda pública de transporte (representada generalmente como una matriz origen-destino), un conjunto de buses y un conjunto de conductores. El resultado es un conjunto de rutas y valores en frecuencias y horarios que se asignan a los buses y los conductores. El problema global es considerado NP-duro en complejidad computacional, así como cada uno de los subproblemas que representa cada etapa. [8]

El diseño de red de transporte (TND) tiene como propósito definir un conjunto de rutas de buses en un área particular, cada ruta siendo determinada por una secuencia de paraderos. Recibe un área con características topológicas y una matriz origen-destino que representa la demanda. La topología del área está definida por los caminos que posee y las áreas que pueden ser consideradas paraderos de bus o zonas de transferencia. Se llama transferencia al cambio de ruta que debe hacer un pasajero al no alcanzar su lugar de destino con la ruta escogida. Las matrices origen-destino tienen un conjunto de puntos de paradero como coordenadas. Las abscisas corresponden a los orígenes y las ordenadas a los destinos de los usuarios. Cada par de coordenadas contiene un número de pasajeros deseando viajar del punto inicial al final en un periodo de tiempo dado. [8]

Existen algunas restricciones y objetivos planteados en la literatura [8], como:

- Redes existentes en las que por razones políticas puede ser indeseable cortar el servicio.
- El porcentaje de cobertura de área que cubre el transporte público, que depende de características como longitud de ruta, densidad, espaciado de ruta y paradas de bus. Por norma general se dice que la población ubicada en un radio de 400 a 500 metros de un paradero de bus entra en este porcentaje.
- Rutas directas.
- Satisfacción de la demanda, que puede verse en términos de distancias cortas y directas, y un número de transferencias mínimo.
- La longitud de las rutas o el número de líneas, esto es importante para las agencias de tránsito ya que debe minimizarse la longitud total de las rutas para reducir la cantidad de recursos (vehículos y empleados) requeridos para mantener el sistema. Es importante que las rutas no sean muy cortas ni muy largas.

Las soluciones en el diseño de red de transporte se forman idealmente por dos componentes principales: un conjunto de rutas y un conjunto de caminos. [8]

El conjunto de rutas está compuesto por todas las rutas encontradas en la posible solución, y cada ruta o línea está compuesta por una secuencia de nodos desde el nodo inicial al final, y por su frecuencia, es decir, el número de salidas en el periodo de tiempo escogido. [8]

El conjunto de caminos está compuesto por el camino que representa cada par de coordenadas origen-destino en la red. El camino puede ser formado por un segmento de una sola ruta o también por caminos que se transitan a pie, por ejemplo el camino desde la residencia a la parada del bus. [8]

Para construir una ruta de transporte público masivo se deben tener en cuenta varios puntos [11]:

- *Identificar los puntos de inicio y final de la ruta:* Estos puntos deben ser zonas importantes, con afluencia de población o relevantes debido a su ubicación.
- *Seleccionar camino:* En lo posible se debe elegir el camino más corto entre los puntos elegidos. En general se deben evitar los desvíos de este camino; sin embargo si un desvío leve que permita acaparar un número considerable de pasajeros es

viable, se puede tomar para aumentar la cobertura del servicio. Como medida se sugiere hacer un desvío si se recoge un número de pasajeros de al menos el 10 % del tiempo de la ruta.

- *Reducción de las paradas de bus:* Al establecer muchas puntos de parada el servicio se sentirá lento e ineficiente, además de causar una posible congestión del sistema. En lo posible se debe mantener un número de puntos de parada bajo.
- *Tiempos de ruta fijos:* La ruta debe tener un itinerario definido y accesible al público. El usuario debe saber cuándo va a empezar el recorrido de una ruta y cuándo va a terminar.
- *Velocidad constante:* Se debe respetar las normas de tránsito, además de no ir a una velocidad muy baja o muy alta.
- *Cálculo de frecuencia:* Para calcular la frecuencia de una nueva ruta es necesario hacer varios viajes sobre esta para determinar tiempos, cantidad de pasajeros transportada y demás parámetros útiles para establecer la importancia de esta ruta y con esto asignar una frecuencia acorde a la situación. Estas frecuencias deben ser frecuencias divisibles por 60 (10, 15, 20, 30, 60) para mayor facilidad de operación.
- *Asignar flota acorde a la demanda:* Es importante determinar la capacidad de transporte cada una de las rutas en el conjunto de rutas y en base a esto asignar un número adecuado de vehículos para prestar el servicio de una manera eficiente.
- *Tiempos de espera para múltiples viajes de un vehículo:* Es necesario que el conductor se mantenga en buena forma física y mental para operar con eficiencia el servicio. Se sugiere tomar descansos de 10 a 15 % del tiempo total del viaje de la ruta antes de comenzar un nuevo viaje.

2.2. Programación por restricciones

Problema de Satisfacción de Restricciones

Un problema de satisfacción de restricciones (CSP) es un problema combinatorio que puede ser resuelto por búsqueda. Existe un algoritmo trivial que resuelve dichos problemas o encuentra que no existe solución. Este algoritmo genera todas las posibles combinaciones de los valores y, a continuación, comprueba si la combinación dada de valores satisface todas las restricciones o no (en consecuencia, este algoritmo se llama generación y prueba). Claramente, este algoritmo toma un tiempo muy largo para ejecutarse por

lo que la investigación en el área de concentrado de satisfacción de restricciones en la búsqueda de algoritmos que resuelven el problema de manera más eficiente, al menos para un subconjunto dado de problemas [12].

La satisfacción de restricciones surgió de la investigación en Inteligencia Artificial (problemas combinatorios, búsqueda) y gráficos por ordenador (sistema Sketchpad, que expresa la coherencia geométrica en el caso del análisis de la escena).

Este paradigma permite tratar una amplia tipología de problemas de combinatoria que pueden ser definidos en variables que toman valores enteros positivos, en los que la solución se puede especificar mediante el cumplimiento de unas determinadas restricciones. Estos problemas aparecen en áreas tales como por ejemplo puzzles, asignación de horarios (scheduling), ubicación de bodegas y asignación de recursos, planificación, almacenamiento y recuperación de información, empaquetamiento, timetabling, modelado, CAD/CAM, entre otros. En su aplicación inciden un conjunto de técnicas informáticas provenientes fundamentalmente de la lógica, matemática, investigación operativa e inteligencia artificial [13].

Las técnicas básicas de este paradigma son la propagación y la distribución de restricciones. La propagación consiste en inferir restricciones a partir de otras ya definidas, esto se logra mediante propagadores que acumulan información en un almacén de restricciones. La distribución divide un problema en varios casos cuando la propagación no puede avanzar más en la búsqueda. Mediante la iteración de estos dos mecanismos, eventualmente se encontrarán soluciones al problema a menos que no sea consistente. [14]

La distribución de restricciones puede elevar la complejidad de la búsqueda del problema a un nivel exponencial debido a la cantidad de subproblemas que pueden ser creados. Para esto, se pueden desarrollar estrategias heurísticas usando propagadores para podar el árbol de búsqueda del problema. [14]

Conceptos

Un dominio finito es un conjunto finito de enteros no negativos. Se usa la notación $m\#n$ para el conjunto que abarca los enteros {m. . . n}.[14]

Una restricción es una fórmula de lógica de predicados. Un problema de dominio finito es un conjunto de restricciones las cuales contienen una restricción de dominio por cada variable definida. Una solución de un problema de dominio finito es una asignación de valores a las variables del problema tal que todas las restricciones sean satisfechas.[14]

La propagación de restricciones es una regla de inferencia para problemas de dominio finito que acorta los dominios de las variables definidas. Por ejemplo, dada la ecuación $X < Y$ con las restricciones de dominio $X \in 23\#32$ y $Y \in 1\#33$, la propagación restringe los dominios de X y Y a $X \in 23\#32$ y $Y \in 24\#33$.[14]

La arquitectura sobre la que está basada la propagación de restricciones es llamada espacio computacional y consiste en un número de propagadores conectados a un almacén de restricciones (fig. 2.1), el cual contiene un conjunto de restricciones básicas sujetas a equivalencia lógica. Los propagadores imponen restricciones no básicas con el fin de recortar los dominios de las variables presentes en el problema. Varios propagadores que comparten una variable pueden comunicarse a través del almacén de restricciones para acotar la asignación de valores posible.[14]

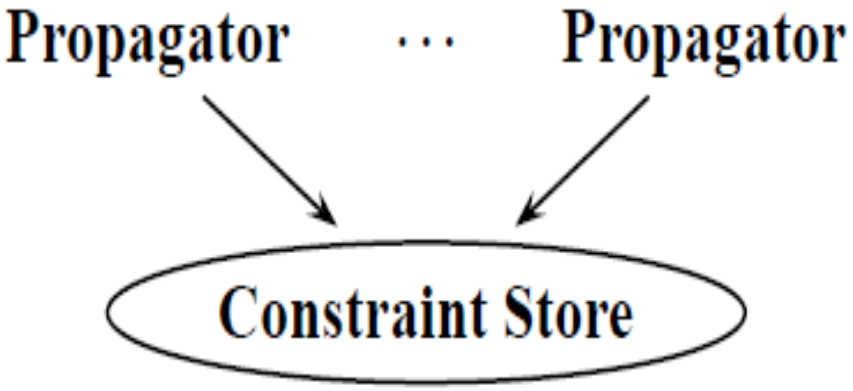


FIGURA 2.1: Arquitectura de propagación de restricciones

Fuente: Tomado de [14]

Un espacio se considera fallido si uno de sus propagadores falla, es estable si todos sus propagadores son estables, y se considera resuelto si no ha fallado y ha consumido todos sus propagadores. Una asignación de variables es considerada una solución si satisface todas las restricciones del almacén y las restricciones impuestas por los propagadores. [14]

Distribuyendo restricciones en los espacios se genera un árbol de búsqueda (fig. 2.2), en el cual cada nodo corresponde a un espacio, y cada hoja representa el estado final del espacio, es decir, solucionado (rombos) o fallido (cuadrados). [14]

Distribución

Un distribuidor es un agente computacional que implementa una estrategia de distribución y se activa cuando la propagación no puede continuar por su cuenta. Generalmente

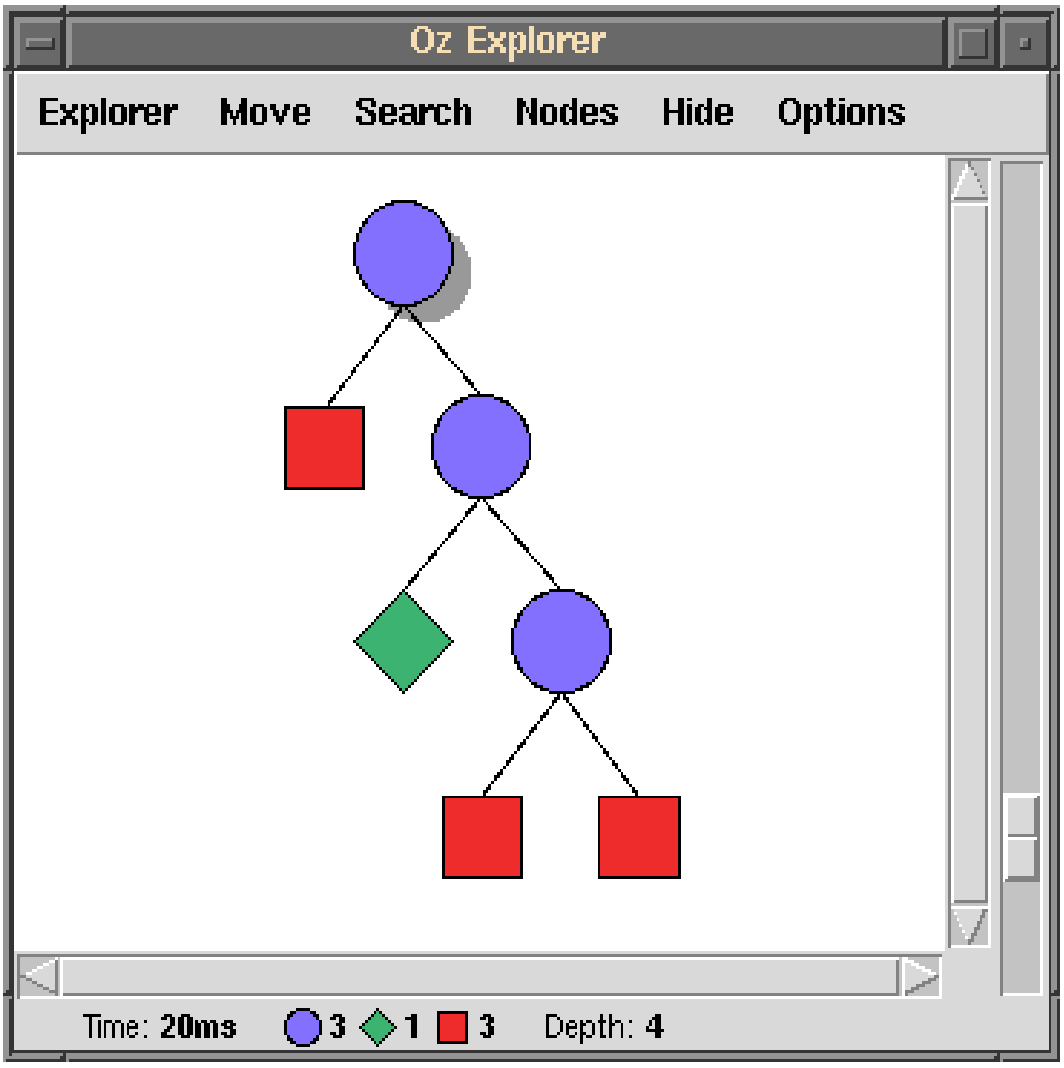


FIGURA 2.2: Árbol de búsqueda de Mozart

un distribuidor se define sobre una secuencia de variables x1...xn, y en caso de ser necesaria una estrategia de distribución, se elige una variable que aún no esté determinada y se distribuye sobre ella. Generalmente se distribuye sobre una variable x usando sus posibles valores límites o en el valor medio del dominio. [14]

Una estrategia de distribución ingenua seleccionará la variable indeterminada más a la izquierda en la secuencia. [14]

Una estrategia de distribución al primer fallo (*first fail*) seleccionará la variable indeterminada más a la izquierda en la secuencia cuyo dominio en el almacén de restricción tiene un tamaño mínimo. En otras palabras, se selecciona la variable más a la izquierda indeterminada para los cuales el número de diferentes valores posibles es mínimo. Se asume que el orden de búsqueda en los árboles en Oz es en profundidad (depth-first). [14]

Resolución de un CSP

La rsolución de Restricciones difiere de satisfacción de restricciones mediante el uso de las variables con dominios infinitos. Además, las limitaciones individuales son más complicados, por ejemplo, igualdades lineales. En consecuencia, la restricción de la solución de algoritmos utiliza los métodos algebraicos y numéricos en lugar de combinaciones y búsqueda. Sin embargo, existe un enfoque que discretiza el dominio infinito en número finito de componentes y, a continuación, se aplican las técnicas de satisfacción de restricciones [10].

Un problema de dominio finito es un conjunto finito de restricciones P tal que P contiene una restricción de dominio para cada variable que ocurren una restricción de P. Una asignación de variables es una función de las variables de asignación de números enteros [14].

Una solución de un problema de dominio finito P es una asignación variable que satisface cada restricción en P. Se tiene en cuenta que un problema de dominio finito tiene a lo sumo un número finito de soluciones, siempre y cuando se considere sólo las variables que se producen en el problema (ya que el problema contiene un número finito de restricciones en el dominio para cada variable que ocurren en él)[14].

Capítulo 3

Modelo de satisfacción de restricciones

En este capítulo se hablará sobre la investigación sobre modelos de redes de transporte público, que incluye la revisión de trabajos posteriores, análisis del problema y recolección de datos con la Secretaría de Tránsito del municipio de Tuluá, selección de variables y finalmente el modelo de satisfacción de restricciones escogido para construir la aplicación.

3.1. Modelos revisados

El diseño de redes de transporte público ha sido un tema de amplia investigación por parte de la comunidad científica, siendo abordado desde distintos métodos de solución. En el trabajo de Farahani et al. Se recopilaron numerosos estudios teniendo en cuenta el tipo de problema a enfrentar, el método de solución y la aplicación real de dichos estudios [6]. Se presentarán los tres antecedentes más recientes.

En el estudio presentado por Yu et al.[15], el objetivo era maximizar el número de viajeros directos por unidad de distancia. El término "directo" indica rutas que se desvíen en la menor medida posible del objetivo, siendo lo ideal una línea recta. La condición principal establecida fue la longitud de las rutas, y el método de solución usado fue el algoritmo de la colonia de hormigas, el cual es usado para encontrar los mejores caminos en grafos.[10]

Mauttone [9] y Urquhart trabajaron sobre la satisfacción de la demanda, buscando optimizar los beneficios para los pasajeros. En este trabajo se construyó un conjunto de rutas aplicando metaheurísticas con el objetivo de minimizar el número de rutas y el tiempo de viaje en el sistema.[6]

En el trabajo presentado por Owais et al. se hace énfasis en la cobertura del servicio, la longitud y la rectitud de las rutas, haciendo uso de una topología circular que permite cubrir la extensión del área objetivo de forma más eficiente. El cálculo de las rutas se hace a través de un algoritmo determinista que incluye entre otras técnicas, programación lineal y entera.[6]

La investigación sobre el problema de diseño de redes de transporte público haciendo uso de la programación por restricciones fue iniciada por un grupo de investigadores de diferentes universidades del mundo (Barra et al). En el trabajo presentado, se sugiere usar esta técnica debido a la capacidad para enfrentar problemas combinatorios de alta complejidad, citando el problema del agente viajero. Los autores proponen un nuevo modelo completo basado en Satisfacción de Restricciones, el cual comprende restricciones esenciales y complementarias al problema, teniendo en cuenta las características del sistema, las necesidades de los pasajeros y los tipos de niveles de servicio [2]. A la hora de implementar el modelo en una herramienta de programación por restricciones, se observó que funcionó bien con instancias pequeñas, pero con instancias grandes la herramienta requirió tiempos muy elevados para procesarlas. La investigación propuesta se basó en este trabajo, ya que dicta las bases para el tratamiento del problema presentado bajo programación por restricciones, teniendo en cuenta que se eligió otra herramienta de software diferente a la usada allí.

3.2. Variables seleccionadas

Para verificar que el modelo escogido fuera correcto, se hicieron reuniones con la Secretaría de Tránsito del municipio de Tuluá con el fin de verificar y corregir las variables seleccionadas para ser utilizadas en el modelo. A cada variable presentada se le asignó una prioridad numérica, siendo 1 la prioridad más baja y 5 la prioridad más alta. Dichas variables fueron escogidas de acuerdo a aspectos en particular referentes al sistema de tránsito que se explicarán a continuación.

Matriz origen-destino: Son necesarias para definir una red de rutas de tránsito que satisfaga la demanda de la comunidad al máximo posible. Una matriz origen-destino tiene un conjunto de puntos de parada como coordenadas. Las abscisas corresponden a los orígenes y las ordenadas a los destinos de los usuarios. La matriz contiene el número de pasajeros que desean transportarse de cada origen a cada destino en un periodo de tiempo dado. **Prioridad**: 5

Longitud de ruta: Un objetivo importante para las empresas de tránsito es minimizar la longitud total de las rutas con el fin de reducir el número de vehículos y personal

necesarios para mantener el sistema funcionando. Se debe tener en cuenta que las rutas no pueden ser muy largas o muy cortas, esto es buscando rendimiento económico. **Prioridad**: 5

Flota operativa: Generalmente, las frecuencias en las rutas dependen de la capacidad de la flota operativa y de cada vehículo de transporte. **Prioridad**: 4

Frecuencia y adelanto: La frecuencia en las rutas debe alcanzar la demanda lo máximo posible para evitar sobrecupos y adelantos excesivos entre los vehículos, consiguiendo así reducir los tiempos de espera de los usuarios. Las entidades reguladoras pueden imponer adelantos mínimos y máximos en algunas rutas a las empresas de tránsito. **Prioridad**: 5

Cabe resaltar que si bien los transbordos (es decir, el uso de dos o más rutas en un solo viaje) son un factor importante en los sistemas de tránsito, en este trabajo no fueron considerados debido a las características de la zona objeto de estudio y a la complejidad añadida que representar en el modelo. Además de esto se pensó en implementar una restricción con el fin de permitir una variación de la flota requerida para operar la solución encontrada, pero se concluyó que no sería significativa debido a las condiciones de las empresas de transporte existentes en el municipio y se descartó para simplificar el modelo.

3.3. Modelo de satisfacción de restricciones

Teniendo en cuenta el modelo seleccionado, como se indicó anteriormente en el marco de este trabajo, se hace una adaptación partiendo de dicha base [2] debido a su simpleza y naturalidad en la etapa de formulación, al tiempo que expone robustez y cumplimiento con las necesidades planteadas en el marco de este trabajo y por la Secretaría de Tránsito en relación a las variables planteadas.

De otro lado, también se tuvo en cuenta su carácter de satisfacción de restricciones y su ausencia de función objetivo, lo que permitió un enfoque orientado a las estrategias de búsqueda y las heurísticas necesarias para diferenciar la solución más óptima de las demás. Cabe aclarar que a diferencia del modelo base [2], en este trabajo no se tendrán en cuenta los transbordos debido a la naturaleza del área sobre la que se va a trabajar y las condiciones propias del Municipio de Tuluá, ni la variación en la flota requerida para la solución sino que se trabajará con un número fijo.

En este orden de ideas, a continuación se plantea la lógica propuesta para la adaptación del modelo y que al tiempo, servirá como solución a la situación problemática planteada en este trabajo.

3.3.1. Anotaciones

Una solución del diseño de redes de tránsito se compone de un **conjunto de rutas**, el cual está compuesto por todas las rutas encontradas en la posible solución, y cada ruta de tránsito o línea está compuesta por su itinerario (secuencia de nodos o secuencia de arcos desde el inicio hasta fin del recorrido) y su frecuencia (número de salidas en el periodo de tiempo predeterminado).

La topología del área evaluada se define por los caminos de circulación vial existentes, y sitios de interés o paradas de bus que sirven como centroides. Representándose como un grafo, se toman las vías como aristas y los centroides como nodos. Este aspecto debe ser definido antes de ejecutar la aplicación, ya que esta es la base para trazar las nuevas rutas encontradas.

3.3.2. Índices y conjuntos

i,$j \in N$ = nodos; $r_k \in R$ = conjunto de rutas de tránsito

3.3.3. Datos de entrada

- n: número de nodos en la instancia.
- C_{ij}: matriz de costos (tiempo) entre los nodos i y j.
- $nRutas$: número de rutas de tránsito en la solución. ($k = 1...nRutas$)
- $minCostoRuta$: mínima longitud de ruta.
- $maxCostoRuta$: máxima longitud de ruta.
- $minPasajerosRuta$: mínimo de pasajeros en la ruta.
- $maxCapRuta$: máxima capacidad de la ruta.
- $minH$: adelanto mínimo.
- $maxH$: adelanto máximo.
- $actualFl$: flota operacional activa en la red de tránsito actual.

- D_{ij}: matriz origen-destino, siendo el nodo i el origen y j el destino. Cada valor d_{ij} representa la demanda.
- CUT: capacidad de la unidad de tránsito (bus).

3.3.4. Restricciones

Restricciones de delimitación

Dado un grafo dirigido $G = (N, A)$ compuesto por un conjunto finito de N nodos y A arcos los cuales conectan pares de nodos, se debe encontrar una solución que contiene una red de tránsito (conjunto de rutas), sujeto a:

$(a_1 \wedge a_2)$ $\forall r_k \in R, minCostoRuta \leq CostoRuta_k \leq maxCostoRuta$ (la longitud de ruta se limita a sus valores mínimo y máximo);

(b) $\forall r_k \in R, PasajerosRuta_k \leq MaxCapRuta$ (el número de pasajeros diario debe ser igual o menor que la capacidad de la línea $MaxCapRuta = maxF_k * CUT$), donde $maxF_k$ es la máxima frecuencia de la línea k y CUT es la capacidad de la unidad de tránsito. Discreto;

(c) $\forall r_k \in R, PasajerosRuta_k \geq minPasajerosRuta$ (el número de pasajeros diario en cada ruta debe valer al menos $minPasajerosRuta$, esto para no perder rentabilidad). Discreto;

(d) $\forall r_k \in R, H_k \geq minH$ (el adelanto H de cada línea k es al menos $minH$). Discreto;

(e) $\forall r_k \in R, H_k \leq maxH$ (el adelanto H de cada línea está limitado por $maxH$). Discreto;

Algunos datos de entrada pueden fijarse a todas las $nRutas$, como $MaxCapRuta$, CTU o F.

El adelanto H es el tiempo (generalmente en minutos) entre dos salidas. $minH$ se relaciona con restricciones técnicas del modo de transporte, y depende de la tecnología de control usada, el nivel de segregación del tráfico general y las congestiones. Si $H < minH$, es probable que ocurran accidentes o problemas operacionales (ej: dos buses de la misma ruta creando un convoy inesperado). Por otro lado, $maxH$ se refiere al nivel de servicio deseado.

La frecuencia F es el número de vehículos por hora en una ruta, $F_k = 60/H_k$ y $maxF = 60/minH$.

$PasajerosRuta$ debe alcanzar al menos el valor de $minPasajerosRuta$ en el día para tener viabilidad financiera. El límite superior ($maxCostoRuta$) y el límite inferior ($minCostoRuta$) de la longitud de la ruta ($CostoRuta_k$) deben ser definidos por el especialista de acuerdo a restricciones técnicas y particularidades de la región a evaluar.

Restricciones operativas

La matriz de adyacencia $A(G)$ de cada ruta encontrada es de tamaño $n \times n$, siendo cada elemento $a_{ij} \in 0, 1$ (1 si existe una arista que une a i y j, 0 si no)

Las restricciones 1, 2 y 3 tienen como finalidad garantizar la validez de la solución verificando la existencia y unicidad de las conexiones entre los nodos adyacentes. Las restricciones 4, 5 y 6 calculan los parámetros de costo en tiempo, demanda transportada y adelanto de la ruta $r_k \in R$:

1. Un nodo solo puede recibir una arista (la suma de cada columna no puede ser mayor a 1)
$$\sum_{j=1}^{n} a_{ij} \leq 1$$

2. Una única arista sale de un nodo (la suma de cada fila no puede ser mayor a 1)
$$\sum_{i=1}^{n} a_{ij} \leq 1$$

3. Un nodo no se puede conectar con sí mismo (la sumatoria de la diagonal principal de la matriz debe ser 0)
$$\sum_{i=1}^{n} a_{ii} = 0$$

4. El costo es igual al producto punto entre la matriz de costos y la matriz de adyacencia.
$$CostoRuta_k = C_{ij} \cdot A_{ij}$$

5. La demanda es igual al producto punto entre la matriz de demanda y la matriz de adyacencia.
$$PasajerosRuta_k = D_{ij} \cdot A_{ij}$$

6. El adelanto o tiempo entre salidas es igual a la división entre el costo y la flota actual con la función piso.
$$H_k = \lfloor \frac{CostoRuta_k}{actualFl} \rfloor$$

Observaciones

Las restricciones de delimitación propuestas fueron clasificadas de acuerdo a su importancia (esencial o complementaria) y la función principal por la que fueron seleccionadas (por razones técnicas u operacionales, mejorar el nivel de servicio, disminuir costos operacionales), como por ejemplo:

Importancia	**Restricción**	**Nivel de servicio**	**Operacional**	**Costos**
	a1 y a2	-	X	-
	b	-	X	-
Esencial	d	-	X	-
	c	-	-	X
Complementaria	e	X	-	-

CUADRO 3.1: Clasificación de las restricciones propuestas

Fuente: Adaptado de [2]

Se han seleccionado 6 restricciones (la restricción 'a' se agrupa en dos), que se dividen en 4 esenciales y 2 complementarias, con 4 restricciones correspondientes a razones técnicas u operacionales, 1 detallando el nivel de servicio y 1 sobre los costos operacionales.

Para optimizar las soluciones encontradas se deben ingresar los parámetros de entrada de tal forma que prioricen las necesidades, es decir, ajustar con precisión ya sea el tiempo de la ruta, el tiempo entre salidas, la demanda transportada o combinaciones de estos parámetros.

Capítulo 4

Implementación del modelo usando programación por restricciones

En este capítulo se hablará sobre la metodología de desarrollo usada para llevar a cabo la implementación del modelo, la estructura del aplicativo y aspectos relevantes que tienen que ver con la estructura de datos y la arquitectura de la aplicación basada en restricciones.

Es importante mencionar que a la hora de implementar el modelo escogido en Mozart se presentaron problemas para modelar conjuntos de más de una ruta, por lo que se decidió trabajar con **una única ruta** en la solución encontrada.

4.1. Consideraciones

Para realizar el proceso de implementación del proyecto se tomaron en cuenta algunas de las pautas que se sugieren en la metodología para aplicaciones web del grupo Avispa[16], las cuales son:

- Formatos de entrada y salida.
- Parámetros de la aplicacion.

4.1.1. Formato de entrada

Para las entradas se utiliza el formato XML usando la especificación dict de XCSP [17]. Para definir una entrada de acuerdo al formato dict, se debe asociar un orden convencional a un diccionario. Este orden convencional especifica un orden de llaves que pueden ser usadas para acortar notaciones. Mientras el grupo de llaves pueda ser conocido desde el contexto, los valores de cada llave pueden ser escritos en una notación abreviada, listando los valores de cada llave que son conocidas en el diccionario.

Para este proyecto se definieron las siguientes llaves:

Llave	**Descripción**
cost	Matriz de adyacencia y costos en tiempo del área a evaluar.
demand	Matriz de origen-destino que representa la demanda del área a evaluar.
nodes	Número de nodos en la instancia.
minRouteCost	Costo de ruta mínimo en tiempo de la ruta solución.
maxRouteCost	Costo de ruta máximo en tiempo de la ruta solución.
minRidership	Demanda mínima que debe transportar la ruta solución.
minH	Adelanto o tiempo entre buses mínimo en la ruta solución.
maxH	Adelanto o tiempo entre buses máximo en la ruta solución.
actualFl	Número de buses disponibles.
CTU	Capacidad de cada bus.

CUADRO 4.1: Llaves del formato de entrada

Fuente: Adaptado de [18]

Una instancia es una representación de una zona geográfica que contiene datos sobre la topografía, los costos y la demanda entre los puntos escogidos. Para representar la topografía de la zona se usa un grafo, siendo los nodos los puntos de interés o centroides de áreas importantes (ej: barrios) y las aristas las conexiones mediante vías entre los puntos. Para los costos y la demanda se usan dos matrices, de tamaño $n * n$, siendo n el número de nodos y cada índice de las matrices representa el costo en tiempo o la demanda de pasajeros del nodo i al nodo j.

La representación del grafo se encuentra en formato JSON, un ejemplo de este archivo puede encontrarse en el anexo de este documento. Los nodos tienen como propiedades un identificador (desde 0 a n, siendo n el número de nodos), un nombre que en este caso corresponde al del punto representado, coordenadas (x, y) para ubicarlos en el plano a la hora de ser dibujados, y un tamaño de visualización. Las aristas tienen como propiedades un identificador, un nodo origen, y un nodo destino.

Es importante resaltar que la matriz de adyacencia que retorna el núcleo de la aplicación abarca todas las aristas entre nodos posibles así no sean utilizadas, por lo tanto el identificador del archivo JSON sirve para encontrar aquellas aristas que deban ser

manipuladas. Este identificador se compone de la letra e y un número que se calcula con $i \times n + j$, siendo i el nodo origen y j el nodo destino.

4.1.2. Formato de salida

Para la salida se define un formato de archivo XML, se puede ver un ejemplo en los anexos del presente documento.

Los campos de información de la **ejecución** generados por el aplicativo se definen de la siguiente manera:

Campo	**Descripción**
spacesFailed	Matriz de adyacencia y costos en tiempo del área a evaluar.
spacesCreated	Matriz de origen-destino que representa la demanda del área a evaluar.
spacesSucceeded	Número de nodos en la instancia.
variablesCreated	Costo de ruta mínimo en tiempo de la ruta solución.
propagatorsCreated	Costo de ruta máximo en tiempo de la ruta solución.
memoryUsed	Demanda mínima que debe transportar la ruta solución.
totalTime	Adelanto o tiempo entre buses mínimo en la ruta solución.

CUADRO 4.2: Campos de información de la ejecución

Fuente: Adaptado de [18]

Los campos de información de la **solución** generados por el aplicativo se definen de la siguiente manera:

Campo	**Descripción**
timeCost	Costo en tiempo total de la solución.
edges	Matriz de adyacencia de la nueva ruta.
headway	Adelanto o espacio entre buses.
demand	Demanda transportada.

CUADRO 4.3: Campos de información de la solución

Fuente: Adaptado de [18]

4.1.3. Parámetros de la aplicación

Los parámetros de la aplicación se definen así:

Parámetro	**Descripción**
–file	Ruta del archivo de entrada.
–salida	Ruta donde se va a guardar el archivo de salida.
–tiempo	Tiempo de ejecución de la aplicación.
–nivelrecomputacion	Nivel de recomputación.
–numeromaxsoluciones	Número máximo de soluciones a calcular.
–tipodebusqueda	Estrategia de distribución elegida.

CUADRO 4.4: Parámetros de la aplicación

Fuente: Adaptado de [18]

4.2. Metodología de desarrollo

Al momento de empezar el proceso de construcción del prototipo de software fue necesario adoptar una metodología de desarrollo, y dado el enfoque de exploración e investigación del proyecto se decidió escoger la metodología ágil SCRUM que permite un desarrollo iterativo con retroalimentación constante a lo largo del ciclo de vida del proyecto.

Se levantaron tres historias de usuario y se definieron cinco sprints para cubrir cada necesidad correspondiente. Cabe mencionar que en los primeros cuatro sprints se trabajó en el núcleo de la aplicación el cual no muestra una funcionalidad específica a la vista del usuario, por lo que no se menciona en la documentación. Esta documentación va anexa en el CD que se entrega junto a este documento.

A continuación se presenta una descripción de los sprints y las actividades realizadas.

1. **Sprint 1**
   - Diseño general de la aplicación.
   - Formulario de ingreso de datos.
   - Generación de archivo de entrada XML a partir del formulario.
   - Implementación de restricciones operativas 1, 2 y 3.
2. **Sprint 2**
   - Diseño de presentación de los resultados.
   - Implementación de restricciones de delimitación a_1, a_2, b y c.
3. **Sprint 3**
   - Lectura de resultados del archivo de salida XML.
   - Implementación de restricciones de delimitación d y e.

4. **Sprint 4**
   - Implementación de librería de dibujo de grafos.
   - Implementación de restricciones operativas 4, 5 y 6.
5. **Sprint 5**
   - Lectura y manipulación del archivo JSON para mostrar la ruta solución.
   - Implementación de parámetros de la aplicación.

Las historias de usuario y el product backlog se definen en las siguientes figuras.

	Enunciado de la Historia				Criterios de Aceptación			
Identificador (ID) de la Historia	Rol	Característica / Funcionalidad	Razón / Resultado	Número (#) de Escenario	Criterio de Aceptación (Título)	Contexto	Evento	Resultado / Comportamiento esperado
OR-001	Como un usuario	Necesito ingresar parámetros a la aplicación mediante un formulario	Con la finalidad de delimitar la solución esperada	1	Visualizar formulario	En caso que el usuario desee delimitar la solución.	Al cargar la página.	el sistema mostrará los campos habilitados para el ingreso de los parámetros.
				2	Validación de parámetros	En caso que los parámetros estén erroneos o los campos vacios.	Al enviar los parámetros.	El sistema mostrará un mensaje de error debido a la mala disposición de los parámetros o a la ausencia de los mismos.
						En caso que los parámetros estén correctos.	Al enviar los parámetros.	El sistema procedera con su ejecución.
				3				
OR-002	Como un usuario	Necesito visualizar los datos correspondientes a la ruta solución	Con la finalidad de interpretar la solución ofrecida por el sistema	1	Visualizar resultados	Para anáilzar los resultados.	Al terminar la ejecución.	El sistema mostrará los datos correspondientes a la ruta solución.
				2				
				3				
				4				
OR-003	Como un usuario	Necesito visualizar la ruta gráficamente	Con la finalidad de ver la solución tangible del problema	1	Visualizar la gráfica de la solución	En caso de requerir una representación de la solución.	Al terminar la ejecución.	El sistema mostrará la gráfica correspondiente a la ruta solución.
				2	Detallar la gráfica	En caso de requerir los detalles graficos de la solución.	Al hacer uso de las funciones del mouse.	El sistema permitirá que la gráfica de la ruta solución pueda ser manipulada para que usuario pueda identificar cada parte de la misma
				3				
				4				

FIGURA 4.1: Historias de usuario del proyecto

Fuente: Adaptado de [19]

Identificador (ID) de la Historia	Enunciado de la Historia	Alias	Estado	Dimensión / Esfuerzo	Iteración (Sprint)	Prioridad
OR-001	Como un usuario, necesito ingresar parámetros a la aplicación mediante un formulario, con la finalidad de delimitar la solución esperada	Ingreso de datos	Hecho	3	1	Media
OR-002	Como un usuario, necesito visualizar los datos correspondientes a la ruta solución, con la finalidad de interpretar la solución ofrecida por el sistema	Informe de resultados	Hecho	5	2,3	Alta
OR-003	Como un usuario, necesito visualizar la ruta gráficamente, con la finalidad de ver la solución tangible del problema	Grafo	Hecho	5	4,5	Alta

FIGURA 4.2: Product Backlog del proyecto

Fuente: Adaptado de [19]

4.3. Estructura de la aplicación

El núcleo del aplicativo fue construido en el lenguaje Mozart/Oz, el cual cuenta con un motor de restricciones de dominios finitos. La carpeta del aplicativo cuenta con varios scripts y archivos ejecutables que permiten generar soluciones basadas en satisfacción

de restricciones. Estos archivos son llamados desde la interfaz web cuando el usuario ha ingresado los datos necesarios para empezar la ejecución.

Las estrategias de distribución son las implementadas por defecto en Mozart, como son:

- FF: Intenta asignar la primera variable de la lista con el dominio más pequeño por el menor valor del dominio.
- Naive: Intenta asignar la primera variable de la lista por el menor valor del dominio.
- Split: Intenta asignar la primera variable de la lista por el valor medio del dominio.

La interfaz web es la capa de la aplicación que interactúa directamente con el usuario, recibiendo las entradas y parámetros del aplicativo. Estos datos se presentan en los formatos definidos previamente para que la aplicación pueda leerlos y procesarlos. Al terminar la ejecución de la aplicación, la interfaz web es la encargada de desplegar los resultados de forma que sean fácilmente entendibles por el usuario.

Un diagrama de flujo de la aplicación está disponible en el anexo al final de este documento.

La aplicación presenta las siguientes capas:

Vista: Se compone de las interfaces web que permiten al usuario interactuar con la aplicación y ver los resultados de su ejecución. Estas interfaces están implementadas en HTML5 usando la librería Materialize como estilo CSS.

Comunicación: Permite conectar las capas de vista y ejecución a través de diferentes funciones cargadas en scripts de lenguajes PHP y JavaScript, estas funciones permiten tomar la información del usuario, procesarla y mostrar una solución de acuerdo a los parámetros elegidos.

Ejecución: Esta capa recibe de la capa de comunicación la información dada por el usuario, la procesa aplicando los algoritmos de programación por restricciones codificados en Mozart/OZ y envía información de la solución encontrada para ser visualizada.

Persistencia: Está compuesta por varios archivos de entrada y un archivo de salida. Los archivos de entrada se componen de:

- Archivo en formato XML para la información del área geográfica a estudiar, es decir, costos y valores de demanda.
- Archivo en formato XML para capturar la información suministrada por el usuario.

- Archivo en formato JSON que contiene el grafo correspondiente a la topografía del área para ser visualizado)

Cuando se genera el archivo XML que captura los datos del formulario, estos datos se mezclan con los datos de costos y demanda bajo las llaves definidas previamente. Un ejemplo de este formato se puede encontrar en el anexo al final de este documento.

Cálculo de costos: Para el cálculo de costos se hace uso de las funciones reificadas del módulo FD. Es importante aclarar que una restricción reificada presenta una salida binaria: 0 si no se cumple o 1 si se cumple; además no esuna restricción que se imponga en la ejecución, es decir, si esta falla la ejecución continúa sin problema.

Conversor de entradas: Para la conversión de entradas se hace uso del parser conocido como XML Parser, que permite tomar una estructura de archivo en formato XML y transformarlo en un árbol de etiquetas que facilita la extracción de la información. El archivo de salida es generado por la capa de ejecución en formato XML y se despliega al usuario al final de la ejecución.

Graficador: Para dibujar el grafo correspondiente a la red de tránsito y la ruta solución encontrada se usa la librería Sigma.js basada en JavaScript, la cual permite modelar el grafo, manipularlo dinámicamente y brindar interactividad con el usuario. Un ejemplo de grafo dibujado por esta librería se puede encontrar en el anexo del manual de usuario al final de este documento.

4.4. Interfaz web

A continuación se describe la implementación de la interfaz web utilizada para el proyecto:

- **Formulario**, mediante el cual se envían parámetros y archivos al aplicativo.
- **CSS y JavaScript**, para modelar la vista general, desplegar los resultados y permitir interacción con el usuario.

Las diferentes vistas pueden ser consultadas en el manual de usuario anexo al final de este documento.

Los resultados de los aplicativos se despliegan así:

- Área de dibujo o canvas sobre el cual se traza el grafo correspondiente al área evaluada y la ruta elegida por el aplicativo.

- Tablas con información sobre la ejecución del aplicativo (estos datos se encuentran detallados en el formato de salida XML definido previamente) y los datos de la solución obtenida (costo, tiempo entre salidas y demanda transportada).

Un ejemplo de la visualización de los resultados se puede observar en el anexo del manual de usuario al final de este documento.

Capítulo 5

Evaluación y resultados

En este capítulo se describe un proceso de pruebas a la aplicación desarrollada en el proyecto, para realizar un estudio con respecto a qué ventajas presentan ciertas características de la aplicación basada en programación por restricciones, como son las estrategias de distribución y el tamaño de las entradas.

5.1. Descripción de las pruebas

Para probar la eficacia y la capacidad del aplicativo se decidió aplicar cuatro entradas que varían en el número de nodos:

- Diminuta: cinco nodos.
- Pequeña: siete nodos.
- Mediana: diez nodos.
- Grande: treinta nodos.

Las instancias se denotan con la letra inicial de su tamaño y un número que identifica la estrategia de distribución:

- D1: Instancia diminuta con estrategia de distribución FF.
- D2: Instancia diminuta con estrategia de distribución Naive.
- D3: Instancia diminuta con estrategia de distribución Split.
- M1: Instancia pequeña con estrategia de distribución FF.

- M2: Instancia pequeña con estrategia de distribución Naive.
- M3: Instancia pequeña con estrategia de distribución Split.
- P1: Instancia mediana con estrategia de distribución FF.
- P2: Instancia mediana con estrategia de distribución Naive.
- P3: Instancia mediana con estrategia de distribución Split.
- G1: Instancia grande con estrategia de distribución FF.
- G2: Instancia grande con estrategia de distribución Naive.
- G3: Instancia grande con estrategia de distribución Split.

Esta categorización del tamaño de entrada se basó en los resultados de la publicación que sirve de guía a este trabajo, la cual indica que se tuvo éxito en instancias con quince nodos o menos[2]. Se propuso que cada entrada fuera evaluada mediante las estrategias de distribución disponibles en Mozart para encontrar diferencias en la búsqueda de soluciones. Los parámetros fueron asignados de forma que se asemejen a la vida cotidiana.

5.2. Pruebas sobre la aplicación

Los parámetros de desempeño y su respectiva codificación son los siguientes:

- EF: Espacios de computación fallidos.
- EC: Espacios de computación creados.
- EE: Espacios de computación exitosos.
- VC: Variables creadas.
- PC: Propagadores creados.
- MU: Memoria utilizada en bytes.
- P: Tiempo total en milisegundos.

Los parámetros de la ruta solución encontrada y su respectiva codificación son los siguientes:

- CT: Costo en tiempo.

- TS: Tiempo entre salidas o adelanto.
- D: Demanda o número de pasajeros transportado.

Los parámetros de entrada elegidos por el usuario fueron:

- Tiempo de ejecución en segundos: 20 segundos.
- Número de nodos según la entrada: 5, 7 y 10.
- Distribución: FF, Naive y Split.
- Costo en tiempo mínimo de ruta: 15
- Costo en tiempo máximo de ruta: 60
- Mínima cantidad de pasajeros en la ruta: 15
- Adelanto mínimo: 5
- Adelanto máximo: 25
- Número de buses existentes: 5
- Capacidad de la unidad de transito: 20

Estas pruebas arrojaron los siguientes resultados:

Prueba	**Solución**			**Recursos gastados**						
Nombre	CT	TS	D	EC	EF	EE	VC	PC	MU	P
D1	31	6	37	2	1	2	28	116	558676	128
D2	31	6	37	2	1	2	28	116	558676	128
D3	31	6	37	2	1	2	28	116	558676	134
P1	36	7	38	274	1	6	52	348	558676	155
P2	36	7	38	252	1	6	52	347	558676	147
P3	36	7	38	237	1	6	52	346	558676	152
M1	22	7	30	83	1	4	103	989	357164	179
M2	20	6	30	63	1	3	103	972	357164	159
M3	22	7	30	83	1	4	103	989	357164	178
G1	x	x	x	x	x	x	x	x	x	x
G2	x	x	x	x	x	x	x	x	x	x
G3	x	x	x	x	x	x	x	x	x	x

Cuadro 5.1: Pruebas de la aplicación

Fuente: Elaboración propia

5.3. Resultados

5.3.1. Conclusiones

- Comparando entre sí las estrategias de distribución que provee el Mozart no presentan diferencias observables para una misma entrada.
- El consumo de memoria de la ejecución depende del motor virtual de Mozart y no del tamaño de la entrada.
- Si la entrada es muy grande no funciona la aplicación, es decir después de 10 nodos.
- Tomando como referencia otros proyectos sobre programación por restricciones [18] se observo que el nivel de recomputación no afecta sensiblemente el desempeño de la aplicación.

5.3.2. Desventajas

La estructura de datos manejada en el proyecto tiene una complejidad de $|N| \times |A|$ siendo N el número de nodos y A el número de aristas de la instancia. Así mismo los propagadores que se deben crear para implementar las restricciones dependen mayormente del tamaño de la entrada, por lo tanto el aplicativo puede fallar para entradas grandes al expandirse demasiados nodos que generan que la maquina virtual de Mozart se desborde.

En las pruebas se evidenció que al pasar de 100 aristas en la matriz de adyacencia, el parser generador de XML de Mozart presenta problemas por el gran tamaño del árbol de dependencias creado.

5.3.3. Aplicación del modelo en la realidad

El proceso de pruebas permite establecer que el modelo utilizado se puede utilizar en casos reales así:

- En instancias pequeñas en las cuales no tengan un gran número de datos de entrada, es decir, de 3 a 10 nodos.
- En casos donde no se haga uso de transbordos.
- El prototipo construido en este proyecto ayuda a tener una aproximación al comportamiento de un aplicación mas completa para la asignación de rutas de transporte público usando programación por restricciones.

5.3.4. Recomendaciones

- En general, para obtener rutas que se adecuen a las necesidades de los usuarios y de las empresas prestadoras del servicio de transporte hay que seleccionar bien los tiempos de viaje, el tiempo entre salidas y la capacidad de demanda que puedan éstas. Estas características son esenciales a la hora de determinar un buen servicio, y se deben decidir con detalle para que no se generen problemas como poca cobertura, viajes muy largos, frecuencias excesivas o demasiado cortas que no se puedan cumplir, saturación del sistema, entre otros.

- El parque automotor influye de manera considerable en la calidad del servicio prestado, pues una flota de buses con buena capacidad de pasajeros permite reducir los tiempos entre salidas lo que significa aumentar la frecuencia; siendo así posible transportar más personas en menos tiempo. Además, viajar en un vehículo en buen estado brinda comodidad y satisfacción a los usuarios. Invertir en renovar la flota de buses mejorará significativamente la prestación del servicio.

- Se debe revisar constantemente el conjunto de rutas existentes para mejorarlas o añadir nuevas rutas que abarquen las partes de la población que no sean satisfechas en el momento. En lo posible estas rutas deben tener tiempos y sitios de inicio y final que sean claros y disponibles para el público, recorrer caminos directos, evitar detenerse en demasiadas ocasiones y no excederse en su longitud. Además la ruta debe atender a una demanda mínima que permita su operación, de lo contrario no será viable y representará pérdidas para la empresa que preste el servicio.

- Además de los aspectos mencionados anteriormente, se deben tener en cuenta las condiciones de las empresas operadoras, las regulaciones de ley vigentes en la zona, la malla vial y el nivel socioeconómico de los sectores de la población. Estos puntos se evidenciaron en la etapa de recolección de investigación, pero no hacen parte de lo estudiado en este trabajo, por lo que se tratarán con mayor detalle en el capítulo de conclusiones.

Capítulo 6

Conclusiones y trabajos futuros

6.1. Conclusiones

1. El modelo de satisfacción de restricciones elegido para este proyecto tiene en cuenta muchas de las variables involucradas en el proceso de diseño de rutas de transporte público masivo, sin embargo presenta dificultad para representar un periodo de tiempo de evaluación dado y también para modelar conjuntos de más de una ruta. Se encontraron limitantes en la cantidad de datos suministrada por la secretaría de Tránsito de Tuluá, en la falta de claridad del modelo sobre cómo implementar el tiempo como una variable continua, en la poca capacidad de procesamiento evidenciada en el capítulo 5, y en la complejidad de la implementación de rutas mediante el paradigma orientado a objetos, en lo que al lenguaje Mozart/Oz se refiere.

2. Debido a la naturaleza del municipio de Tuluá, no se tomaron en cuenta aspectos importantes de un sistema de transporte más grande, como son los transbordos y variación en la flota disponible. Estos aspectos se deben implementar al estudiar una zona más grande o un sistema de transporte más complejo que el del municipio estudiado.

3. Se debe tener en cuenta que la validez de las soluciones arrojadas por el aplicativo están sujetas a los parámetros ingresados por el usuario en la interfaz, por lo tanto se sugiere tener conocimiento del área a evaluar para obtener resultados cercanos a la realidad.

4. Se pudo establecer que la solución implementada en este proyecto queda sujeta a algunas limitaciones computacionales como por ejemplo el tamaño de la entrada, si es muy grande el aplicativo puede fallar y no mostrarse soluciones a una instancia.

5. Según la encuesta recogida en la secretaría de Tránsito del municipio de Tuluá la cual se anexa en el CD que se entrega con este documento, se pueden inferir particularidades que evidencian cómo los ciudadanos usan el sistema de transporte:

   - El sistema de transporte público masivo es fuertemente utilizado en barrios de estrato medio-bajo y poco utilizado en barrios de estrato alto, dado que las personas que residen allí tienen mayor poder adquisitivo y principalmente se movilizan en taxis o vehículos particulares.
   - La gran mayoría de personas usa el sistema de transporte público masivo para ir al centro del municipio y volver a sus casas; existen registros que evidencian viajes que recorren otros puntos de la ciudad pero son una minoría, por lo que solo se obtiene una representación parcial de la demanda a lo largo de la ciudad.
   - Como se estableció previamente, las personas viajan desde y hacia el centro con mucha mayor frecuencia que en cualquier otro punto; sin embargo la frecuencia de viajes hacia el centro es más grande que la frecuencia de viajes con origen en el centro. Este hecho podría revelar algún comportamiento importante de la población respecto a las rutas que pasan por esta zona. Se recomienda que se haga un análisis de los motivos de esta situación los cuales podrían ser tiempos de esperas mas largos, mayor congestión, rutas mas largas desde el centro a sus lugares de residencia, entre otras.

6. Además de los aspectos relevantes a este proyecto, se sugiere mejorar las condiciones para la prestación del servicio de transporte público masivo en la ciudad de Tuluá; estas condiciones van desde mejorar los vehículos del parque automotor y la calidad de los conductores a disminuir los precios del servicio. Cabe destacar que mejorando las condiciones del servicio se pueden mejorar los parámetros operativos, por ejemplo al adquirir vehículos en buen funcionamiento y con mayor capacidad se puede aumentar las frecuencias de las rutas.

7. El prototipo desarrollado presenta limitaciones para ser utilizado en una zona amplia; sin embargo éste presenta los elementos esenciales de una red de transporte público, por lo cual se considera que este prototipo y el modelo utilizado para construirlo pueden servir como una base para otros estudios orientados a lograr soluciones de TND con mayor detalle.

6.2. Trabajos futuros

1. Estudiar otros modelos y/o restricciones adicionales que permitan obtener soluciones más acordes a la realidad o que optimicen los recursos computacionales requeridos.

2. Usar otra herramienta de programación por restricciones más potente y/o orientada a objetos, como es el caso de Gecode (*C++*).

3. Aplicar el paradigma de programación por restricciones junto a una herramienta de programación concurrente, esto para aumentar la capacidad de ejecución del aplicativo.

4. El proyecto usa formatos XML con datos que se basan en encuestas tomadas de la Secretaría de Tránsito del municipio, pero que no son realmente útiles para determinar la asignación de la demanda general de la zona. Se sugiere implementar otros formatos o métodos de entrada, o apoyarse en un sistema de información geográfico (**SIG**) para obtener los datos de los costos y los valores de demanda de las zonas a evaluar en el aplicativo.

Apéndice A

Anexos

A.1. Diagrama de flujo de aplicación

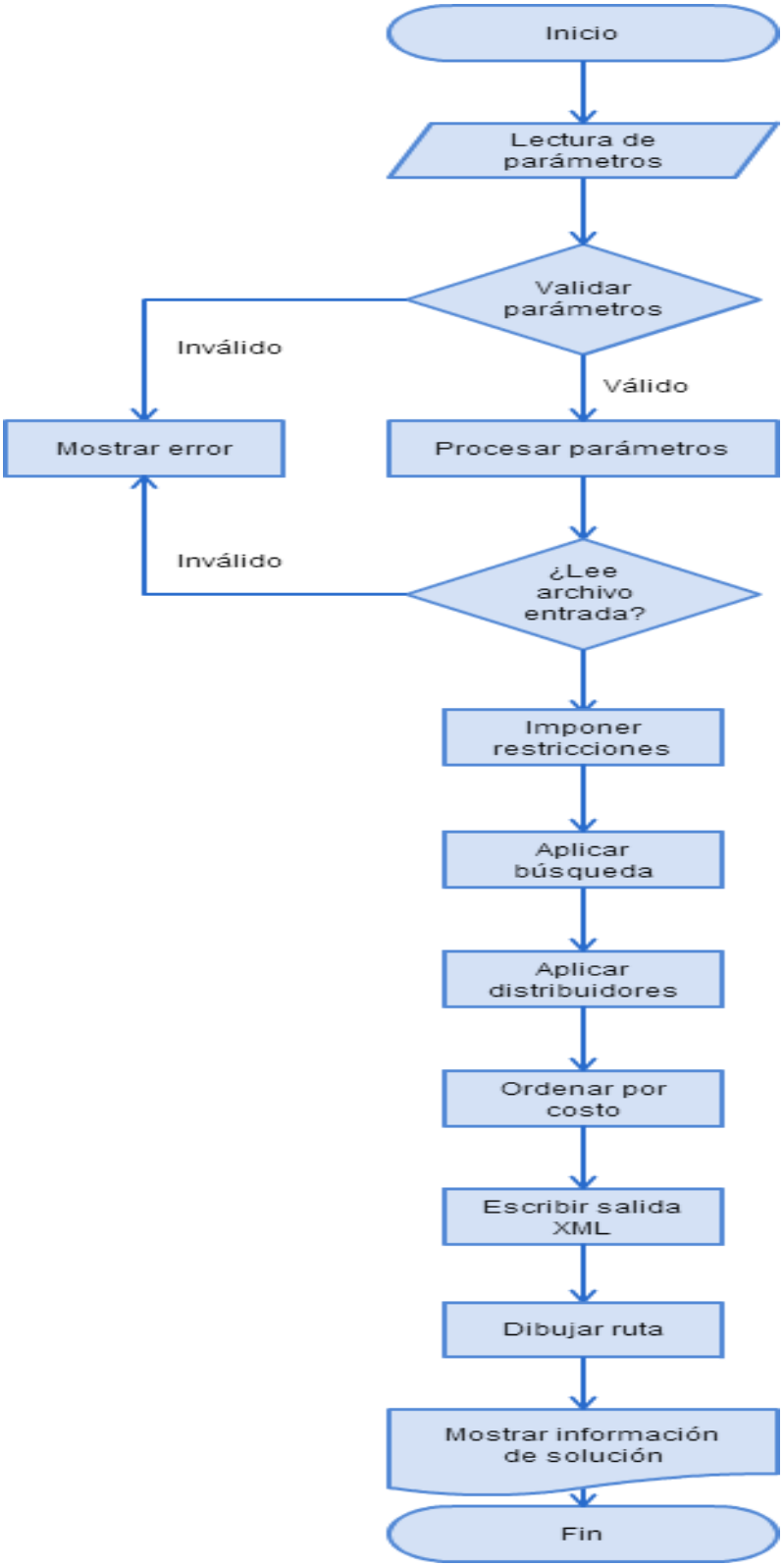


FIGURA A.1: Diagrama de flujo de la aplicación

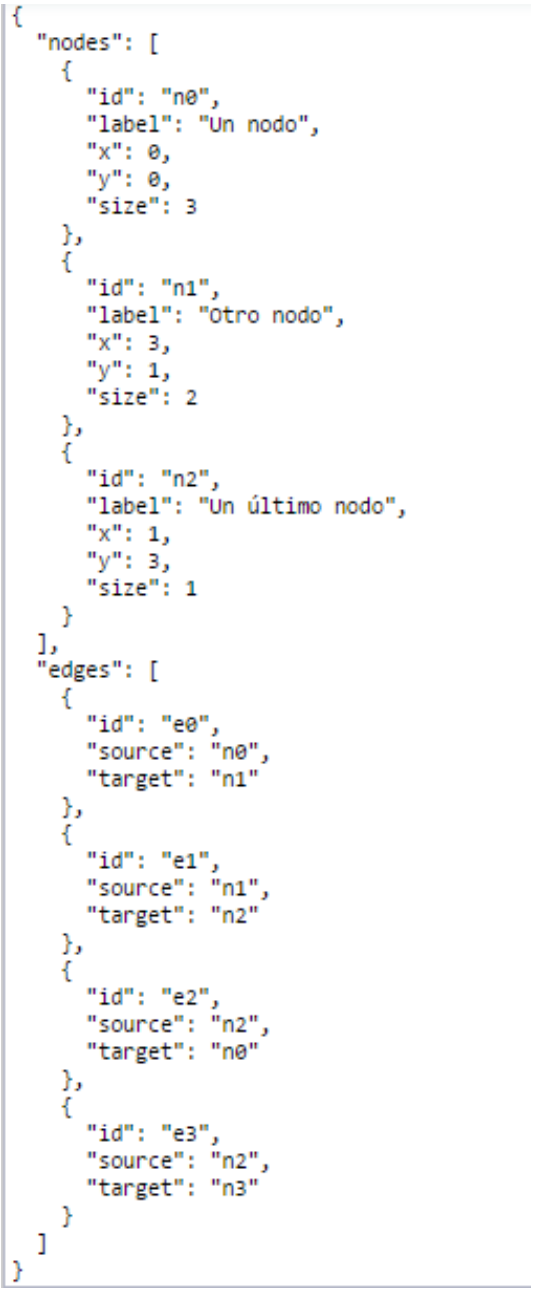
Fuente: Elaboración propia

A.2. Formato de entrada XML

```
<instance>
  <presentation nbSolutions="?" format="XCSP 2.1"/>
  <dict>
    <entry key="cost">
      <list>
        <i>0</i>
        <i>2</i>
        <i>3</i>
        <i>0</i>
      </list>
    </entry>
    <entry key="demand">
      <list>
        <i>0</i>
        <i>10</i>
        <i>30</i>
        <i>0</i>
      </list>
    </entry>
    <entry key="nodes">
      <i>2</i>
    </entry>
    <entry key="minRouteCost">
      <i>15</i>
    </entry>
    <entry key="maxRouteCost">
      <i>25</i>
    </entry>
    <entry key="minRidership">
      <i>8</i>
    </entry>
    <entry key="minH">
      <i>5</i>
    </entry>
    <entry key="maxH">
      <i>10</i>
    </entry>
    <entry key="actualFL">
      <i>2</i>
    </entry>
    <entry key="CTU">
      <i>15</i>
    </entry>
  </dict>
</instance>
```

Fuente: Elaboración propia

A.3. Formato JSON de instancia de grafo



```
{
  "nodes": [
    {
      "id": "n0",
      "label": "Un nodo",
      "x": 0,
      "y": 0,
      "size": 3
    },
    {
      "id": "n1",
      "label": "Otro nodo",
      "x": 3,
      "y": 1,
      "size": 2
    },
    {
      "id": "n2",
      "label": "Un último nodo",
      "x": 1,
      "y": 3,
      "size": 1
    }
  ],
  "edges": [
    {
      "id": "e0",
      "source": "n0",
      "target": "n1"
    },
    {
      "id": "e1",
      "source": "n1",
      "target": "n2"
    },
    {
      "id": "e2",
      "source": "n2",
      "target": "n0"
    },
    {
      "id": "e3",
      "source": "n2",
      "target": "n3"
    }
  ]
}
```

FIGURA A.2: Archivo JSON que representa el grafo

Fuente: Elaboración propia

A.4. Formato de salida XML

```
<?xml version="1.0" encoding="UTF-8"?>
<Solutions authorXML="Julian Felipe Bonilla,Andres Felipe Lince">
        <head solution="Optima">
                <spacesFailed>13</spacesFailed>
                <spacesCreated>1</spacesCreated>
                <spacesSucceeded>4</spacesSucceeded>
                <variablesCreated>28</variablesCreated>
                <propagatorsCreated>129</propagatorsCreated>
                <memoryUsed>558676</memoryUsed>
                <totalTime>128</totalTime>
        </head>
        <solution>
                <timeCost>28</timeCost>
                <edges>
                        <edge ID="0"> 0 </edge>
                        <edge ID="1"> 0 </edge>
                        <edge ID="2"> 0 </edge>
                        <edge ID="3"> 0 </edge>
                        <edge ID="4"> 1 </edge>
                        <edge ID="5"> 0 </edge>
                        <edge ID="6"> 0 </edge>
                        <edge ID="7"> 0 </edge>
                        <edge ID="8"> 1 </edge>
                        <edge ID="9"> 0 </edge>
                </edges>
                <headway>9</headway>
                <demand>29</demand>
        </solution>
</Solutions>
```

Fuente: Elaboración propia

A.5. Manual de usuario

Requerimientos de software

La estación de trabajo donde se desee usar el prototipo de software deberá satisfacer los siguientes requerimientos de software:

- Servidor web (ej: Apache).
- PHP 5.2.3 o superior.
- Máquina virtual de Mozart 1.4.0.
- El aplicativo está diseñado para ser un servicio web por lo que debe ejecutarse en un navegador, preferiblemente Google Chrome o Mozilla Firefox en su última versión.

Con estos requerimientos de software satisfechos se espera que el aplicativo funcione correctamente. Para ejecutar la aplicación es necesario que la carpeta .ªplicacion"se encuentre alojada en su servidor de aplicaciones. Una vez ubicada se puede acceder a ella por el navegador en la ruta *"nombreservidor/aplicacion"*

Por ejemplo si se usa AppServ como entorno de desarrollo, se debe guardar la carpeta en C:\AppServ\www. . Ejemplo: *"localhost/aplicacion"*

Al ingresar a la aplicación usted se encontrará con un formulario que le permitirá suministrar todos los parámetros necesarios para ajustar la solución que retornará la aplicación (Figura A.3). Ingrese los valores en los campos como los crea convenientes de acuerdo a su experiencia, elija la estrategia de distribución deseada y cargue los archivos JSON y XML. Recuerde que debe tener estos archivos definidos previamente y que todos los campos deben ser numéricos. Cuando todo esté listo, haga clic en el botón Ejecutar.

La aplicación se bloqueará por el tiempo de ejecución que usted especifique. Al terminar podrá ver la ruta solución obtenida en forma de grafo el cual usted podrá agrandar, alejar y mover a su gusto. (Figura A.4).

Posteriormente podrá encontrar la información referente a la ejecución y a la solución en una tabla de dos pestañas (Figura A.5).

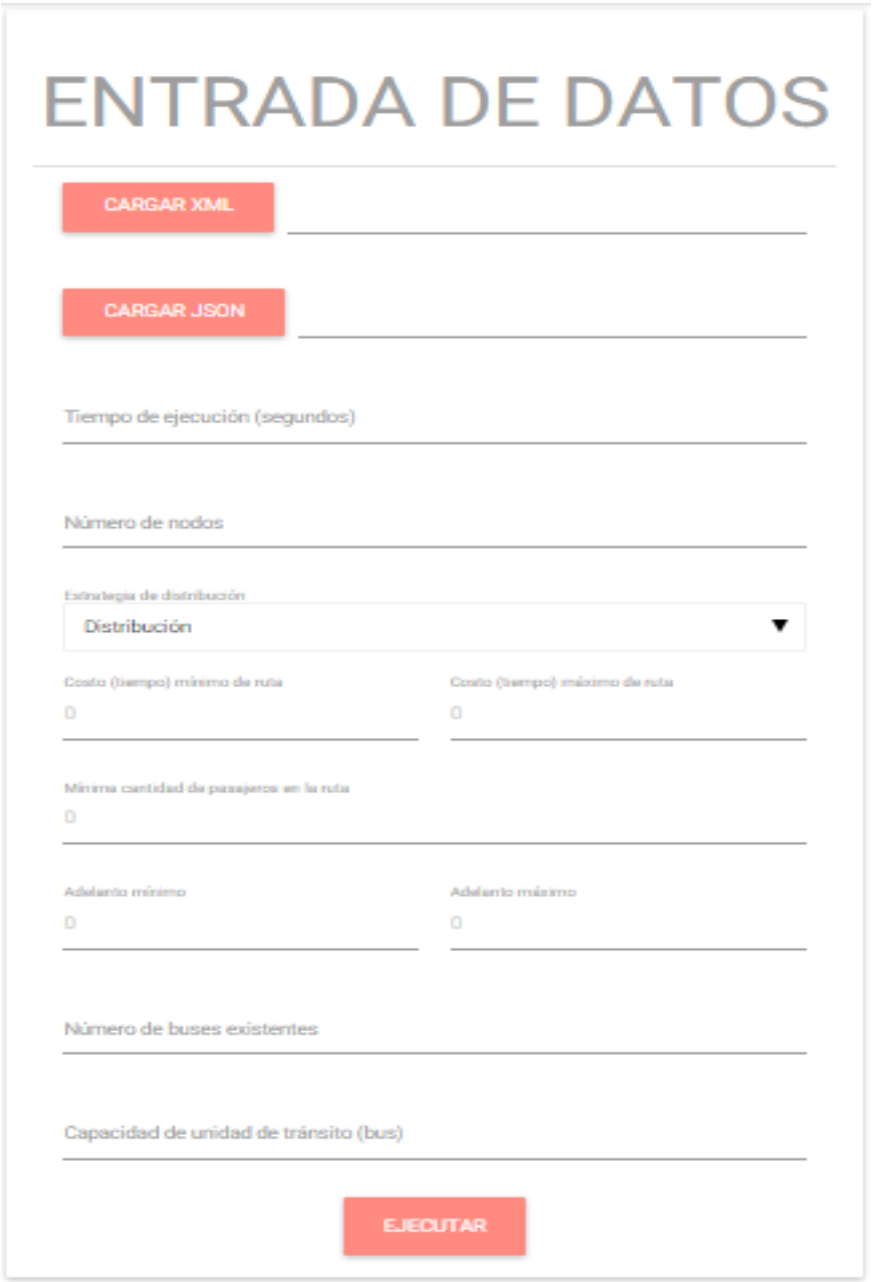


FIGURA A.3: Formulario de la aplicación

Fuente: Elaboración propia

FIGURA A.4: Ruta solución

Fuente: Elaboración propia

(A) Datos de ejecución

(B) Datos de la solución

FIGURA A.5: Tablas de resultados

Fuente: Elaboración propia

Bibliografía

[1] L. F. Pasmín, *Comunicacion personal.*

[2] A. Barra, L. Carvalho, N. Teypaz, V.-D. Cung, and R. Balassiano, "Solving the transit network design problem with constraint programming.," Junio 2007.

[3] M. de Transporte de Colombia, "Transporte automotor - preguntas frecuentes."

[4] S. de Movilidad de Bogotá, "Formulación del plan maestro de movilidad para bogotá d.c."

[5] D. C. R. Lozano, "Análisis de los factores que han limitado la implementación de los decretos 112 a 116 de 2003 de reestructuración del sistema de transporte público colectivo en bogotá."

[6] R. Farahani, E. Miandoabchi, W. Szeto, and H. Rashidi., "Review on urban transportation network design problems.," *European Journal of Operational Research*, pp. 281–302, 2013.

[7] P. J. Álvarez, C. G. Calderón, and G. G. Calderón, "Route optimization of urban public transportation.," Septiembre 2007.

[8] V. Guihaire and J. Hao, "Transit network design and scheduling: A global review.," Diciembre 2008.

[9] A. Mauttone, "Optimización de recorridos y frecuencias en sistemas de transporte público urbano colectivo.," 2005.

[10] "Wikibooks - fundamentals of transportation.."

[11] C. MacKechnie, "How transit routes and schedules get planned.."

[12] R. Barták, "On-line guide to constraint programming.," 1998.

[13] F. Barber, "Problemas de satisfacción de restricciones.,"

[14] C. Schulte and G. Smolka, "Finite domain constraint programming in oz.," Febrero 2000.

[15] B. Yu, Z. Yang, C. Cheng, and C. Liu, "Optimizing bus transit network with parallelant colony algorithm.,"

[16] J. F. Díaz, C. A. Delgado, M. A. Cruz, and L. F. Vargas, "Metodología para crear aplicaciones de programación por restricciones como servicio web. definición e implementación con diversos casos.,"

[17] O. C. of the Third International Competition of CSP Solvers, "Xml representation of constraint networks format xcsp 2.1," 2008.

[18] C. A. Delgado, "Diseño e implementación de una aplicación prototipo para la gestión del espectro radioeléctrico usando programación por restricciones.," 2012.

[19] "Pmoinformatica.com - la oficina de proyectos de informática.."