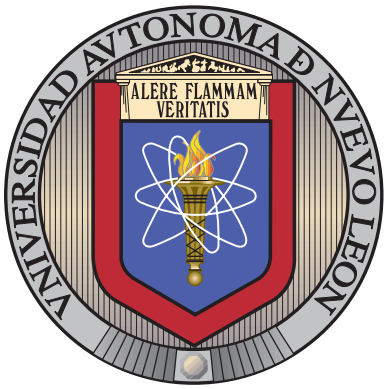
Universidad Autónoma de Nuevo León

Facultad de Ingeniería Mecánica y Eléctrica

División de Estudios de Posgrado



Planificación Dinámica de Rutas del Transporte Público a Partir de los Requerimientos del Usuario

por

Fernando Elizalde Ramírez

en opción al grado de

Maestría en Ciencias

en Ingeniería de Sistemas

San Nicolás de los Garza, Nuevo León Junio 2013

Universidad Autónoma de Nuevo León

Facultad de Ingeniería Mecánica y Eléctrica

División de Estudios de Posgrado

Planificación Dinámica de Rutas del Transporte Público a Partir de los Requerimientos del Usuario

por

Fernando Elizalde Ramírez

en opción al grado de

Maestría en Ciencias

en Ingeniería de Sistemas

San Nicolás de los Garza, Nuevo León Junio 2013

Universidad Autónoma de Nuevo León

Facultad de Ingeniería Mecánica y Eléctrica

División de Estudios de Posgrado

Los miembros del Comité de Tesis recomendamos que la Tesis «Planificación Dinámica de Rutas del Transporte Público a Partir de los Requerimientos del Usuario», realizada por el alumno Fernando Elizalde Ramírez, con número de matrícula 1576998, sea aceptada para su defensa como opción al grado de Maestría en Ciencias en Ingeniería de Sistemas.

El Comité de Tesis

Dr. Romeo Sánchez Nigenda

Director

Dra. Yasmín A. Ríos Solís

Revisora

Dr. Hugo Jair Escalante Balderas

Revisor

Vo. Bo.

Dr. Moisés Hinojosa Rivera

División de Estudios de Posgrado

San Nicolás de los Garza, Nuevo León, Junio 2013

A mi Abuelita Andrea, mi Madre,

Tíos y Hermana.

Índice general

Índice de figuras

Índice de tablas

Agradecimientos

Ante todo quiero agradecerle a Dios por todo lo que me ha dado, por todas esas oportunidades, personas que me ha puesto en el camino para poder alcanzar mis metas, por darme salud, estabilidad, amigos, sabiduría y por esa familia a la cual pertenezco.

A mi abuelita Andrea y mi mamá Rosalba por todo el amor que me han dado, por cuidarme, por hacerme sentir importante, a mis tíos Raúl y Jacinto por todo su apoyo incondicional que me han brindado no solo en esta travesía si no desde siempre, por haberme enseñado a ver las cosas desde un punto donde no debo afectar los demás como dijera el primero de ellos, aprender ver más allá de las barañas, y a todos ellos en conjunto por haberme criado, por apoyarme en mis decisiones y haber hecho de mi lo que soy hoy en día. A mi hermana aunque vivamos peleando sabemos mutuamente que allí estamos para lo que sea. También quiero agradecerle a mis demás tíos y primos que siempre me han apoyado.

Agradezco a todo el apoyo brindado por la Dra. Yasmín Ríos Solís para que yo pudiera entrar a la maestriía y durante todo el desarrollo de ésta, al Dr. Romeo Sánchez Nigenda por haber aceptado ser mi asesor en esta tesis de investigación por su paciencia, por su apoyo intelectual y personal, durante los estudios y en la vida misma, por compartirme su conocimiento, a los profesores del PISIS.

Me gustaría decirle lo agradecido a Doña Yolanda por haberme recibido en su casa, por mantenerme a raya, procurar que yo estuviera cómodo y sano, a mis primas Laura, Cristina y Rosa por haberme soportado 2 años y a mi tía Alicia.

A todos esos viejos y grandes amigos que han estado conmigo en esta aventura, que me han visto contento, enojado, tirado por la calle de la amargura, por apoyarme y ayudarme en mis decisiones y en cada momento en mi tesis, Agustín, Joaquín Azueta, Mario Rivera, Linda, Germain, Tephy, Omar, Beto, Ing. Atilano, Iris Abril, Elizabeth Briand, entre muchos otros de ellos de Cerro Azul y alrededores, gracias. Qué decir de mis nuevos amigos y compañeros de la maestría con los cuales hemos compartido grandes vivencias y recibido apoyo incondicional en cada uno de ellos, bajo cualquier circunstancia, los que están Vicky, Alexis, Cristina, Nancy, Luis, Andres, Paulina, y de los que ya no están Alex, Diana y Nely, gracias hasta por haberme aguantado en mis penas dijera Nancy. Y a una persona importante en este proceso a quien quiero agradecerle enormemente, aun siendo que la conocí ya casí a finales de la mestría, ella es Carolina Medellín Moreno, por todo su apoyo intelectual y personal que me ha brindado, pero sobre todo por su amistad, y a cada uno de los chicos pisianos.

Quiero agradecer de modo muy especial a la Facultad de Ingeniería Mecánica y Eléctrica, a la Universidad Autónoma de Nuevo León por haberme recibido, y dado un lugar para poder llevar acabo mis estudios, y de distracción, por la beca de pago de colegiaturas, al CONACyT por haberme brindado la beca mensual con la cual pude cubrir mis gastos de manutención en ésta ciudad, y a los contribuyentes.

Resumen

Fernando Elizalde Ramírez.

Candidato para el grado de Maestro en Ciencias
con especialidad en Ingeniería de Sistemas.

Universidad Autónoma de Nuevo León.
Facultad de Ingeniería Mecánica y Eléctrica.

Título del estudio: Planificación Dinámica de Rutas del Transporte Público a Partir de los Requerimientos del Usuario

Planificación Dinámica de Rutas del Transporte Público a Partir de los Requerimientos del Usuario

Número de páginas: 158.

Objetivos y método de estudio: *Objetivo General*

Resolver el problema de la generación de planes de viaje para el usuario utilizando rutas de autobuses del transporte público, considerando para la planificación de viajes los requerimientos o preferencias del usuario. Se propone el uso de modelos de planificación que nos permitan capturar tanto la topología de la red de transporte como las métricas y requerimientos de los consultantes.

En paralelo a este objetivo, se desarrolla un análisis del estado del arte de los

actuales algoritmos de planificación y la factibilidad de desarrollar un modelo de planificación de acuerdo a estos algoritmos, resaltando sus debilidades y áreas de oportunidad para ésta y futuras investigaciones mediante los resultados obtenidos.

Específicos.

- Estudio de los factores que comprenden el dominio del transporte público.
- Análisis de las preferencias o requerimientos del usuario en el uso del transporte público.
- Análisis para poder entender qué características del problema son modelables, y el grado de impacto en el problema original por aquellas que no se pudieron modelar.
- Diseño y desarrollo de funciones multiobjetivo para representar los múltiples criterios de selección de los usuarios de transporte público.
- Generación de un modelo de planificación que represente los factores y preferencias antes analizados para su resolución.
- Desarrollo de un algoritmo de Dijkstra que solucione una versión limitada del problema original. El fin es obtener un benchmark que nos permita comparar la factibilidad de las técnicas de planificación en modelos simples y relajados.
- Creación de un generador de instancias para la experimentación y evaluación.
- Resolución y validación de rutas de viaje usando el modelo de planificación generado y algoritmos autónomos de planificación de Inteligencia Artificial.
- Análisis de los resultados para identificar aquellos factores críticos del modelo que incrementan la complejidad de resolución de los algoritmos de planificación actuales.

Contribuciones y conclusiones: Contribuciones:

- Modelo de planificación de rutas de viaje del dominio de transporte público.
- Problema multiobjetivo (diversas métricas a optimizar de acuerdo a lo que requiere el usuario).
- Generación de planes en base a las preferencias del usuario.

Conclusiones:

- Facilidad práctica de las técnicas de planificación para el modelado del problema de transporte usado en la generación de planes de viaje para el usuario de acuerdo a sus necesidades.
- Insuficiencia de los planificadores actuales al momento de computar un plan con base en el modelo desarrollado debido al alto tiempo de cómputo para mostrar una solución cuando se hace uso de las preferencias como restricciones.
- Baja memoria por parte de los planificadores, lo cual impide al soporte de una red completa de transporte público.
- Facilidad para el desarrollo de un modelo donde se combinen diversos tipos de transporte.
- Las técnicas usadas permiten la descomposición de un problema en subproblemas.
- Necesidad de hacer uso de dos modelos distintos, uno para cuando existen restricciones u otra cuando no existen estas.
- La calidad del plan mejora conforme se incrementa el número de restricciones, pero de igual modo se incrementan los tiempos de resolución.
- Resultados con mucha variabilidad entre problemas similares.

- Mediante la experimentación se tiene que los planificadores LPG y SGPLAN tienen la suficiente capacidad para soportar una red de transporte de ciudades medianas o grandes de México.

Firma del asesor: _____

Dr. Romeo Sánchez Nigenda

Capítulo 1

Introducción

1.1 Descripción del problema

Uno de los problemas más complejos y estudiados en el dominio del transporte es el de la generación de rutas de viaje entre un punto de origen s a un punto destino t, tomando en cuenta métricas de optimización, como el tiempo estimado de viaje, costo de transporte, distancia recorrida, étc., y las restricciones propias de la red.

Aunque ha habido grandes avances en la modelación y solución de este tipo de problemas [58] resolviendo mapas de viaje continentales [6], poco se ha avanzado para generar planes de viaje en el dominio del transporte público tomando en cuenta los requerimientos del usuario. El problema radica en que el dominio del transporte público incluye propiedades no antes relevantes o presentes en el problema de rutas en general.

Por ejemplo, en el dominio del transporte público, los usuarios están interesados en obtener rutas de viaje que se acomoden a sus preferencias particulares, no necesariamente relacionadas a las métricas de optimización de la empresa que provee el servicio. Más importante aún, es el hecho que el cómputo de las rutas de viaje tiene que considerar la síntesis de soluciones con respecto al espacio y al tiempo. Es decir, una solución satisfactoria entre dos puntos s y t en las primeras horas del día, puede ser una solución completamente ineficiente para las primeras horas de la noche. Además de lo anterior se tiene una relación a que el cómputo de los planes

de viaje se ven afectados por el lugar geográfico donde se encuentran los principales actores del problema (es decir el espacio donde se encuentran localizados autobuses y usuarios) y por el efecto de sincronización implícito que hay en el problema (es decir, la ventana de tiempo que tendrían que cubrir dichos actores para concordar en algún punto en el espacio).

Entonces cuando planeamos, estamos evaluando desde puntos de origen diferentes para cada actor, involucrando puntos intermedios donde dichos actores deben sincronizarse, para llegar a un punto destino (objetivo).

Podemos distinguir en nuestro problema de transporte público principalmente cuatro clases de propiedades, aquellas relacionadas al:

a. ambiente,

b. topología de la red,

c. al usuario final y

d. objetivos del problema.

Las propiedades relacionadas con el ambiente se refieren al dinamismo en tiempo real que afectan las operaciones de las unidades del servicio público, como ejemplo tenemos: rutas en constante cambio en relación con el tiempo, es decir están sujetas a retrasos debido al tráfico en ese momento, descompostura de la unidad, percances con alguna otra unidad u objetos, bloqueos, condiciones ambientales, entre muchos otros factores propios del medio, por lo tanto el usuario depende del sistema.

El problema que introduce el dinamismo en el ambiente tiene como particularidad que un plan que inicialmente era válido podría dejar de serlo debido a los cambios que el ambiente introduce. Lo cual es el problema que tenemos aquí.

Existen dos formas básicas de lidiar con este problema dependiendo también qué tan dinámico es el ambiente.

1) Replanificación incremental

2) Replanificación completa

En replanificación incremental, se conserva el plan afectado, y se le van haciendo modificaciones (inserción de nuevas actividades, remoción de otras, modificación en sus tiempos de inicio, étc.) incrementalmente. La idea es que si resulta muy costoso planificar, entonces es mejor utilizar el plan en turno y se le va modificando de acuerdo a los cambios que surjan en el ambiente.

La segunda forma general de replanificación, es mandar el plan actual a la basura y tomar un *snapshot* actual del mundo es decir, identificar cuál es el estado actual después de que se produjo la disrupción por el ambiente. Entonces se toma al estado actual como el estado inicial y se vuelve a planificar completamente (se genera un nuevo plan del nuevo estado inicial al final). Obviamente si el proceso de planificación es muy costoso (e.g., en términos de tiempo, o recursos) no es posible hacer esto.

Estas propiedades se encuentran también presentes en el problema tradicional, cuando uno se encuentra manejando con apoyo de un GPS se puede encontrar esta clase de eventos y se pueden ofrecer rutas alternas para alcanzar el objetivo debido a la existencia de total libertad para desplazarse dentro de la red de carreteras. Lo anterior no puede llevarse a cabo directamente en nuestro problema, debido a qué las rutas del transporte público están predefinidas [44]; no pueden modificarse libremente a las necesidades del pasajero, imponiendo restricciones adicionales al proceso de síntesis de soluciones [44].

Dentro de las propiedades relacionadas a la topología de una red de transporte público podemos encontrar el número de rutas, sus recorridos y la ubicación de las paradas. Además de información de las frecuencias de paso u horarios de servicios, distancias cubiertas y costos asociados al uso del transporte.

Es importante mencionar que aunque ya existen horarios determinados de lle-

gada del autobús a una parada, o del tiempo entre autobuses, estos nunca se respetan, las paradas mismas no se respetan en la mayoría de los sistemas existentes (en México). A pesar de que esto afecta directamente al momento de computar un plan de viaje, no lo tomaremos en consideración en éste trabajo, dejándolo como trabajo futuro.

Las propiedades referentes al usuario se subdividen en dos categorías, en la primera se consideran la situación actual y los objetivos del usuario; es decir, el punto de inicio del recorrido y el destino del mismo.

La segunda parte de estas propiedades son las preferencias del usuario [7] [46], muchas de éstas relacionadas a métricas heredadas de la topología de la red (costo económico de transporte, tiempo de recorrido, distancia a recorrer, étc.) y otras a sus gustos personales (distancia a caminar, número de transferencias, tiempo de espera, étc.). El problema se convierte en multiobjetivo.

Una de las propiedades más importantes que se deriva de las anteriormente mencionadas es la planeación con respecto al espacio y tiempo. Esta propiedad es tal vez la más importante en el dominio del transporte y la que lo distingue del problema general de cómputo de rutas de viaje. Planificar con respecto a espacio y tiempo implica considerar múltiplos espacios geográficos donde los recursos (autobuses) se localizan con respecto al tiempo de planeación del usuario (ya sea tiempo de llegada o arribo en un destino). Esto implica un nivel de sincronización entre los actores del problema para generar viajes que permitan estimar tiempos de arribo partiendo de los lugares geográficos donde se encuentran distribuidos los autobuses y las estimaciones de tiempos de recorridos necesarios para que los recursos lleguen a donde se necesitan. Es decir, queremos llegar a la parada de tal manera que tengamos el mínimo tiempo de espera para abordar el autobús. El problema es que no queremos salir inmediatamente, sino que buscamos generar un plan de viaje a futuro, maximizando el grado de sincronización del usuario y una unidad de transporte en un punto de intersección (i.e., parada) disminuyendo de esta manera el tiempo de espera del usuario en tal punto de intersección.

Por lo tanto independientemente del ambiente y la topología, también los objetivos son distintos. Estos involucran indirectamente la sincronización de partes, una de las cuales no tenemos control sobre ella (el transporte), propiedad no presente antes en el problema general.

A diferencia de los algoritmos actuales de cómputo de rutas de viaje que se basan mayormente de los Algoritmos de Dijkstra y A*, nosotros proponemos el diseño y uso de modelos de planificación considerando las propiedades de una red de transporte público y las necesidades y/o preferencias del usuario. Técnicas de planificación han sido utilizadas anteriormente en [21], [38], [39], [9], [2] entre otros pero no consideran las preferencias de los usuarios y ninguna restricción salvo la definida por las rutas establecidas. Creemos que la aplicación de técnicas de planificación a nuestro problema en particular tendrá un efecto positivo en la expresividad de los modelos a generar, así como también en el tamaño y complejidad de los problemas a resolver. Esperando obtener planes de muy buena calidad en tiempos de cómputo cortos.

Para el desarrollo de estos modelos de planificación utilizaremos el lenguaje de planificación PDDL-2.1 [22], debido a que es un lenguaje estandarizado para éste tipo de propósitos.

1.2 Justificación del problema.

En un medio ambiente donde no se tolera la pérdida de tiempo y la mayoría de los habitantes de una ciudad usa el transporte público en sus actividades diarias [51], se le debe brindar una atención especial hacia sus necesidades al momento de computar un plan de viaje.

Para lograr una mayor satisfacción del usuario, diversas empresas han implementado sistemas de monitoreo y control de sus unidades, pero ésta información solo es observada por los controladores, además, solo se obtienen resultados aproximados,

ya que el envío de información no es de forma continua.

La información solo fluye en una sola dirección, hacia la empresa transportista. Es decir, no existe información para el usuario que le permita planificar mejor sus viajes. Lo que se propone es que nuestros modelos tomen una fotografía actual del sistema (red de transporte), y acorde a las necesidades del usuario se generen planes de viaje que les permita travesar la red más eficientemente.

En la actualidad no existen trabajos donde se considere este problema combinando las diversas preferencias del usuario [46], esto quizá se deba a la complejidad del problema abordado, y en el tiempo necesario para encontrar una ruta de viaje ya que el resultado debe ser arrojado lo más pronto posible hacia el usuario.

Nuestros modelos se centran en capturar las preferencias o información relacionada al usuario al momento de querer computar un plan de viaje, además de la topología y los elementos que conforma el sistema mencionado.

El desarrollo de la presente tesis pretende avanzar el estado del arte en el cómputo de rutas de viaje en redes de transporte público. Entre nuestras aportaciones tenemos: a) El desarrollo de un modelo computacional de redes de transporte público usando técnicas de planificación. b) El enriquecimiento de los modelos de planificación para considerar diferentes requerimientos de los usuarios de las redes de transporte. A nuestro entender, no existen trabajos que solucionen el dominio del transporte público con técnicas de planificación considerando las preferencias de los usuarios al momento de planear su viaje, c) un análisis exhaustivo soportado con datos experimentales, que cubrirá la solución de los modelos generados con algoritmos de planificación actuales con el fin de descubrir las ventajas y limitaciones de dichas tecnologías, y así encontrar los puntos críticos para mejorarlos en investigaciones futuras.

Lo que pretendemos es corroborar si las técnicas de planificación actuales pueden ser utilizadas para resolver nuestro tipo de problemas, y alumbrar en la dirección correcta a la que deberían apuntarse esfuerzos de futuras investigaciones.

1.3 Objetivos.

General

Resolver el problema de la generación de planes de viaje para el usuario utilizando rutas de autobuses del transporte público, considerando para la planificación de viajes los requerimientos o preferencias del usuario. Se propone el uso de modelos de planificación que nos permitan capturar tanto la topología de la red de transporte como las métricas y requerimientos de los consultantes.

En paralelo a este objetivo, se desarrollará un análisis del estado del arte de los actuales algoritmos de planificación y la factibilidad de modelar nuestro modelo de acuerdo a estos algoritmos, resaltando sus debilidades y áreas de oportunidad para esta y futuras investigaciones mediante los resultados obtenidos.

Específicos.

- Estudio de los factores que comprenden el dominio del transporte público.
- Análisis de las preferencias o requerimientos del usuario en el uso del transporte público.
- Análisis para poder entender que características del problema son modelables, y el grado de impacto en el problema original por aquellas que no se pudieron modelar.
- Diseño y desarrollo de funciones multiobjetivo para representar los múltiples criterios de selección de los usuarios de transporte público.
- Generación de un modelo de planificación que represente los factores y preferencias antes analizados para su resolución.
- Desarrollo de un algoritmo de Dijkstra que solucione una versión limitada del problema original. El fin es obtener un benchmark que nos permita comparar la factibilidad de las técnicas de planificación en modelos simples y relajados.

- Creación de un generador de instancias para la experimentación y evaluación.
- Resolución y validación de rutas de viaje usando el modelo de planificación generado y algoritmos autónomos de planificación de Inteligencia Artificial.
- Análisis de los resultados para identificar aquellos factores críticos del modelo que incrementan la complejidad de resolución de los algoritmos de planificación actuales.

1.4 Hipótesis

Modelar el problema de generación de rutas de viaje en el dominio del transporte público usando técnicas de planificación producirá una mayor flexibilidad y expresividad para representar las propiedades del mismo, incluyendo las diferentes métricas de optimización; y permitirá la resolución de instancias grandes. Parte de la investigación será determinar si esto es posible y si el modelo resultante se puede resolver con técnicas actuales, o cúales serían las modificaciones que se necesitarían para que dichas técnicas pudieran representar y resolver nuestro problema.

1.5 Aportaciones científicas

El problema planteado de generación de planes de viaje de transporte público muy pocas veces se ha estudiado. Aunque en la actualidad ha cobrado un gran auge debido a la necesidad de mejorar el transporte público y de brindar mayores herramientas a los usuarios, y por otro lado, incrementar el uso de este servicio de transporte, ya sea por cuestiones ecológicas y/o para reducir la carga vehicular de las carreteras, mejorando así los tiempos de viaje [37].

El usuario mediante esta inplementación tendría una herramienta confiable para poder generar planes de viaje hacia los diversos puntos que desea alcanzar

dentro de una ciudad, de acuerdo a lo que él requiere, teniendo un mayor control de su tiempo y de su economía durante el día.

No solo el usuario se vería beneficiado con nuestro trabajo, si no también las empresas transportistas, ya que mediante este sistema una empresa puede estimar la demanda esperada a cierta hora o día, además de identificar los puntos de mayor demanda, permitiendo a la empresa una mejor planeación hacia una mayor satisfacción de la demanda, si el modelo fuera dinámico se le podría brindar tiempos de recorrido aproximados y de demanda durante el día para sus rutas.

Los modelos resultantes de este proyecto pudieran ser utilizados haciéndoles ciertas modificaciones, en problemas de logística, cadena de suministros, ruteo de vehículos y otros dominios afines. Es decir, áreas que consideren la necesidad de transportar algún bien a un destino final, satisfaciendo demandas, restricciones y optimizando costos o preferencias que se tengan.

Se puede relacionar con problemas de ruteo tipo estrella, donde las variables y requerimientos son similares a un problema de lógistica, con la variante que son viajes de ida y vuelta entre la matriz a un punto en específico, y se utiliza más que nada para la generación de rutas para el transporte de material peligroso.

Otra área de potencial aplicación sería la de planeación turística donde se desea reducir el costo de viaje, pero al mismo tiempo visitar el mayor número de puntos de interés en el menor tiempo posible. Nuestros modelos permitirían indicar al turista que acciones llevar a cabo para lograr cubrir todos los puntos de su interés y cuál es el costo de éste. Minimizando estos últimos.

Podemos utilizar nuestro modelo para la planificación de la producción dentro de una empresa manufacturera o de procesos, ya que qué se requiere producir diversos materiales en diversas máquinas, lo cual implica llevar materia prima o producto terminado entre las estaciones de trabajo. Teniéndose en este tipo de problemas limitantes de capacidad, tiempos de procesamiento y de recorrido.

Existen algunos sistemas de generación de rutas de viajes usando el transporte

público donde se manejan algunas preferencias objetivos de los usuarios [60], sin embargo en todos los casos solo consideran un solo objetivo a optimizar al momento de trazar el plan y solo utilizan algoritmos tradicionales.

Teniendo en consideración lo anterior, nosotros estaríamos contribuyendo con los siguientes puntos:

Modelo de planificación de rutas de viaje del dominio de transporte público.

Como bien mencionamos anteriormente, nuestra primera contribución es tratar de resolver este tipo de problema del transporte público mediante modelos de planificación, ya que no existen otros modelos de PDDL (planificación) que aborden este tema, buscando con nuestra investigación generar los primeros modelos eficientes para resolver este tipo de problema. El modelo de planificación a desarrollar se verá enriquecido por las necesidades o preferencias de los usuarios al momento de realizar un viaje, manejando estas últimas como restricciones o funciones a optimizar, lo cual le da un mayor grado de complejidad al mismo.

Lo anterior nos permitirá hacer un análisis de los factores de mayor importancia en la solución del plan de viaje para determinar cuáles son los factores que aumentan la complejidad en resolver este tipo de problemas usando modelos de planificación, con el fin de extender o diseñar algoritmos que los puedan resolver más eficientemente.

Problema multiobjetivo.

Como bien se ha hecho hincapié, los algoritmos actuales solo optimizan una métrica, ya sea costos, tiempos y/o distancia. Sin embargo, el usuario podría requerir optimizar más de un solo objetivo o preferencia, llevando a hacer un uso combinado de estas preferencias.

Generación de planes en base a las preferencias del usuario.

Como bien se mostrará más adelante, los generadores de rutas de viaje actuales, ya sea mediante planificación o algoritmos clásicos, están libres de restricciones. Es

decir, sin limitantes a la hora de realizar el computo de la ruta de viaje, además que en las rutas generadas se puede ir por cualquier punto, y no tienen la limitancia de rutas predefinidas como es el caso del transporte público.

Los algoritmos tradicionales solo optimizan una sola métrica (tiempo o distancia de recorrido, étc.). Donde el usuario tiene menos restricciones y un control completo de sus acciones. Es por eso que el mismo problema, en el ámbito del transporte público, cambia drásticamente.

Pero el usuario al ser quien hace uso de este servicio de transporte, tiene sus propias necesidades o preferencias individuales, las cuales representan restricciones para el dominio al momento de generar un plan, por ejemplo tenemos: la distancia máxima a recorrer hasta la parada, el gasto económico capaz de aceptar, étc..

Asimismo atacaremos otras preferencias que tiene el usuario al querer hacer uso del transporte público, o para que su viaje sea más placentero. De igual modo se podrá hacer combinaciones entre estas preferencias según los deseos del consultante. Estas preferencias se enlistan a continuación y fueron tomadas de [46], y de una encuesta realizada por una empresa de transporte público de la Cd. de Monterrey.

- Menor costo.
- Menor número de transferencias.
- Evitar caminar demasiado.
- Reducir tiempo de espera.
- Combinación de las métricas tiempo costo para la generación de un plan de viaje.
- Cómputo de planes de viaje considerando espacio y tiempo (simulación de movimientos independientes).

1.6 Metodología

En nuestro trabajo de investigación iniciaremos con la realización de un estudio y análisis del problema a abordar, entendiendo todas sus propiedades y elementos que intervienen. Una vez definido el problema, se analizarán problemas abordados en otros trabajos de investigación para conocer los elementos y técnicas utilizadas para dar solución al problema presentado, ya sea mediante algoritmos tradicionales o de planificación. Esto para tomar ideas para el desarrollo de nuestro modelo.

Aparte de estas propiedades, también se realizará un análisis de las preferencias o necesidades del usuario al momento de planear su viaje usando el sistema de transporte público.

Se llevará a cabo un estudio sobre el marco de planificación entendiendo sus conceptos y su modo de aplicación, llevándonos a estudiar el lenguaje de planificación PDDL-2.1, sus requerimientos y necesidades, y la forma de expresar un modelo de planificación con este lenguaje.

Una vez realizados estos análisis se seleccionarán a todas aquellas propiedades y preferencias que queremos representar en nuestro modelo. Para ello se verá la factibilidad de modelar estas propiedades en lenguaje PDDL-2.1.

Diseñaremos modelos de planificación que representen nuestro problema, para poder así hacer un diseño de experimentos, y a partir de este poder probar si las técnicas de planificación son eficientes para el modelado del problema de cómputo de planes de viaje a través de una red de transporte público y determinar las limitantes de los solvers a usar.

En el diseño de experimentos se generará un modelo alternativo de planificación para poder comparar su rendimiento contra un algoritmo básico de rutas de viaje (Algoritmo de Dijkstra) el cual se programara para este fin, y de este modo ver la competitividad de los modelos de planificación ya que si el modelo básico no puede resolver este tipo de problemas un modelo completo tampoco podrá.

Para todos los experimentos a realizar se tomarán diversos tamaños de redes, número de rutas y paradas y tratar de resolverlos para analizar la eficiencia de estas técnicas que estamos aplicando. Se seleccionarán sistemas de planificación para la resolución de nuestros modelos, de acuerdo a sus requerimientos y propiedades.

Una vez hecho lo anterior se procederá a la realización de la experimentación con los distintos modelos desarrollados, para poder llevar a cabo una evaluación y análisis del rendimiento y de la calidad de la solución, comparando los resultados obtenidos del modelo de planificación (diversos planificadores) contra el Algoritmo de Dijskstra. A partir de lo anterior se buscará que factores afectan al rendimiento de los sistemas utilizados.

1.7 Estructura de la tesis

Al comienzo de nuestra tesis describiremos el problema y su importancia para dar a conocer lo que pretendemos hacer, así mismo presentamos los objetivos general y específicos del problema, la justificación del por qué decidimos abordar este proyecto de investigación, así como la hipótesis del mismo, lo cual muestra el grado de relevancia de éste trabajo de investigación.

También en este primer capítulo se presenta las aportaciones científicas que se esperan alcanzar al termino de nuestra investigación, además de la relevancia del problema, y ya para finalizar esta sección, se muestra la metodología a seguir durante el desarrollo de la tesis de investigación.

El segundo capítulo presenta el estado del arte de los algoritmos actuales para generar rutas de viaje, así como también de las diversas aplicaciones existentes y disponibles en la actualidad.

De igual modo abordaremos el planteamiento del problema donde se describirá más a detalle nuestro trabajo. Introduciremos a fondo los conceptos necesarios para el entendimiento de esta investigación, como lo son, planificación, planificado-

res, lenguaje PDDL-2.1, así como conceptos propios del transporte y de transporte público.

Una vez presentado el marco teórico, se procederá identificar los diversos factores y variables que representan nuestro problema, junto con esto, seleccionar las diversas preferencias que tienen los usuarios del transporte público al momento de abordar una unidad pública para realizar su traslado o al momento de realizar una consulta interactiva (tiempo de viaje, de abordaje, comodidad, étc.). Todo esto se podrá ver en el tercer capítulo.

El capítulo 3 también mostrará el desarrollo del modelo de planificación de rutas del transporte público, donde se presentan las restricciones en base a las preferencias de los usuarios utilizadas para la creación del dominio de planificación, así como su relevancia en el modelado del dominio explicando cómo afectan en la generación de planes de viaje.

En el cuarto capítulo se presentará el diseño de experimentos necesario para la evaluación y análisis de los modelos y sistemas de planificación utilizados.

En este mismo capítulo se pondrá un mayor énfasis en evaluar la eficiencia computacional de los planificadores, ya que queda muy claro que el usuario necesita obtener resultados de una forma rápida. Sumado a esto, se realizará un análisis exhaustivo de los resultados obtenidos por los sistemas de planificación con el fin de evaluar la calidad de los planes de viaje generados, y así identificar áreas de oportunidad para mejorar las heurísticas de los mismos cuando las soluciones no sean satisfactorias.

En el quinto capítulo mostraremos los resultados obtenidos en esta investigación, así como una explicación comprensiva de los resultados, y para finalizar se presentará el trabajo a futuro a realizar para mejorar los sistemas de planificación y el modelo de planificación, así también nuevos implementos heurísticos y probabilísticos para brindar planes más confiables a los usuarios del transporte público.

Capítulo 2

Marco Teórico

2.1 Descripción del problema

El problema de generación de rutas de viaje se puede representar utilizando dos grafos. El primer grafo representa la red de autobuses del sistema *G(V,E)*, con V vértices correspondiendo a las paradas de autobuses y E arcos correspondiendo a las rutas que cruzan los vértices del grafo que dependen del número de rutas R. El conjunto de rutas R es definido, es decir, no se pueden modificar a placer del usuario. Cada a en E tiene pesos correspondientes al tiempo de duración *T(a)* entre cada V para todos los arcos a. Estos tiempos son no-negativos.

El segundo grafo representa la red de caminos *W(K,L)* por los cuales el usuario p puede desplazarse libremente caminando o en alguún otro medio ajeno a los autobuses del transporte público, con K vértices que representan las esquinas de una cuadra y ℓ arcos b, todos ellos con longitud $\ell(b)$ no-negativa. Tanto los arcos a y b son bidireccionales.

El problema radica en que dado un vértice de cualquiera de las dos redes, se pretenda llegar a otro vértice en las mismas dos redes, es decir, el plan de viaje desde el nodo de inicio s al nodo final t. Para el problema consideramos un conjunto PR de preferencias del usuario, las cuales funcionan como restricciones al momento de computar un plan o como preferencias a optimizar. Estas PR son ingresadas directamente por el consultante.

Estas dos redes deben ser presentadas en un modelo de planificación en formato PDDL-2.1 adhiriendo a éste modelo las preferencias que tiene el usuario al hacer uso de este servicio, siendo parte de ellas funciones a optimizar *f(s,t)* para un plan del nodo s hasta el nodo t y aquellas que restringen al modelo de planificación.

Realizado el modelo se ejecutaran diversos experimentos con distintas características o variables del modelo con la finalidad de encontrar las limitantes de estas técnicas para este tipo de problemas. Esta forma de cómputo de rutas de viaje utilizando planificación no han sido explotadas y en lo qué que se ha encontrado en la literatura no se consideran las preferencias de los usuarios, así como también en los sistemas que utilizan métodos o algoritmos tradicionales para el computo de una ruta de viaje, esto se presentará más adelante.

2.2 Transporte urbano

En las grandes y medianas ciudades el uso del transporte público tiene gran relevancia en las actividades diarias de sus moradores. En las ciudades medianas de la República Mexicana del 50 al 60 % de los traslados se hacen mediante el uso del transporte público, mientras en las ciudades de mayor tamaño el porcentaje es superior al 80 % [47]. Como ejemplo tenemos a la Ciudad de México [59] en la cual el 78.5 % de los viajes dentro de la ciudad en el 2007 se realizaron utilizando el transporte público, y el 44.55 % de estos viajes se llevaron a cabo en microbuses y/o autobuses [42]. En la figura 2.1 mostramos las preferencias sobre el transporte público por parte de los usuarios, donde se demuestra lo anteriormente mencionado para el caso del transporte público de la Ciudad de México.

2.2.1 Características del servicio de transporte urbano.

Como bien se menciona en el primer capítulo de este trabajo, nosotros centramos nuestros esfuerzos en la generación de modelos de planificación para el transporte público. En esta primera etapa de trabajo nos enfocamos a los autobuses, porque

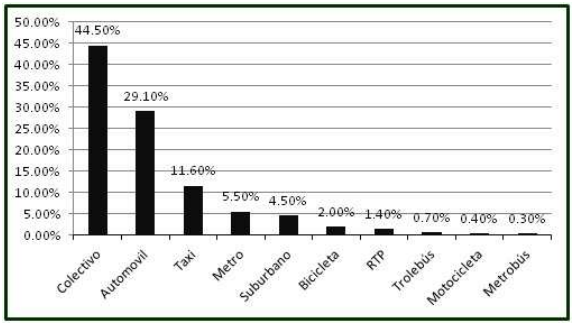


Figura 2.1: Porcentaje de distribución de uso del transporte en la zona metropolitana del Valle de México 2007 [59].

como transporte público es el que mayor demanda tiene [59].

El objetivo principal del transporte urbano es satisfacer la demanda de potenciales clientes ofreciendo un servicio satisfactorio, para esto se estima el número de unidades para satisfacer la demanda diaria de usuarios. Estas rutas deben ser accesibles a los usuarios, las paradas deben estar ubicadas en zonas seguras, es lo que se espera y/o requieren para el buen funcionamiento del sistema, pero debido a diversas situaciones, cambios en el dominio del transporte, medio ambiente, entre otras variables, esto difícilmente se cumple de una forma óptima. Por ejemplo, pueden ocurrir accidentes que incrementen el tiempo de paso de las unidades sobre un determinado punto afectando a la red de transporte en general.

Uno de los principales problemas en México en el transporte público es la sincronización de las unidades, es decir, no existen márgenes de llegada estables a un cruce de varias rutas (coordinación). Esto se debe a que el servicio en su mayoría es privado [46] y cada empresario ve por sus intereses, además que dentro de las mismas rutas existe competencia entre los operadores [16], lo cual desestabiliza más al sistema. Al final todo opera para obtener mayores ganancias mientras la satisfacción del usuario pasa a segundo término.

De las principales problemáticas que se tiene en el modelado de las redes de transporte en México es que en la mayor parte de los sistemas existentes cada esquina puede ser usada como una parada a pesar de la existencia de las oficiales, haciendo

caso omiso de éstas. Es por ello que el número de paradas reales varía de acuerdo a lo registrado y dependen más que nada de la longitud de las rutas de autobuses. Por todo esto no se conoce con precisión la distancia de separación entre paradas. En definición tenemos que la distancia entre estos puntos de abordaje recomendable en zona urbana es entre 300 a 500 metros, mientras que en zonas suburbanas es de 800 metros [48].

La capacidad de las unidades en la Ciudad de Monterrey es de 45 personas sentadas, pero sumando las que pueden ir de pie la capacidad se incrementa a 70, aunque como usuarios del sistema sabemos que suele ser rebasada en horas pico. Esta parte de la capacidad es importante debido a que es una preferencia del usuario, ya que cabe la posibilidad que se desee encontrar asientos disponibles al momento de abordar el transporte, esta cantidad de asientos disponibles puede ser conocida gracias al sistema GPS con que cuentan las unidades.

Si nuestros modelos son rápidos de resolver, tal vez podríamos en un futuro implementar un marco que fuera dinámico para amortiguar un poco los problemas que generan una demanda variable en el cómputo o selección de las rutas de viaje.

2.3 Estado del arte

Debido a la naturaleza de nuestro problema, la revisión del estado del arte se dividió en dos categorías, en la primera de ellas presentamos trabajos relacionados a los algoritmos tradicionales de la ruta más corta. Por otro lado se recopilo artículos y tesis que atacan problemas iguales o similares que mostramos utilizando planificación ya sea con sistemas multi-agente [38][39] , con PDDL (Planning Domain Definition Language) [23] o PDL (Perl Data Language) [9].

Generación de planes de viaje utilizando Algoritmos tradicionales.

Resultados clásicos.

La mayoría de las técnicas actuales de cómputo de rutas de viaje se basan en

los algoritmos de Dijkstra [19] [35] [62] y A* [4] [41], modificados para considerar problemas de gran escala como puede verse en [58] [33]. Los algoritmos tradicionales presentados son aplicables al cómputo de viaje donde se excluye el uso del transporte público. Para este se cuenta con la restricción de rutas predefinidas, lo cual a diferencia de los trabajos mostrados en esta sección, no se tiene todo el grafo de caminos para alcanzar el objetivo final del usuario, además de que no cuentan con las preferencias del usuario las cuales nosotros ingresamos en nuestro trabajo.

El algoritmo de Dijkstra [19] [35] [62] consiste en ir explorando todos los caminos más cortos que parten del vértice origen y que llevan a todos los demás vértices, cada vértice es etiquetado si es el de menor peso acumulado durante ese escaneo; cuando se obtiene el camino más corto desde el vértice origen al resto de vértices que componen el grafo, el algoritmo se detiene, o cuando se alcance el nodo destino, para esto se requiere de un arreglo donde se va guardando los pesos acumulados de cada vértice etiquetado [19].

Este algoritmo nos garantiza la ruta más corta entre un origen y un destino, aunque para poder arrojar la ruta más corta, necesita explorar toda o gran parte del área de búsqueda.

Para mejorar su eficiencia se desarrolló para este mismo algoritmo una búsqueda bidireccional [58] [17], la cual hace una búsqueda de la ruta más corta a partir del vértice origen hacia el destino y viceversa. Cuando ambas búsquedas se interceptan, el algoritmo se detiene ya que se ha encontrado la ruta más corta entre estos dos puntos. El área de búsqueda de este algoritmo es menor que el de Dijkstra [19], [35], [62], disminuyendo con esto el costo computacional para encontrar la solución. Estas estrategias son inflexibles o ineficientes en cara de los requerimientos complejos como con la red o las consultas relativas a múltiples destinos u orígenes [34]. Su aplicación práctica no es posible para el transporte público por que no conoce el nodo de destino en la red dependiente del tiempo, ya que se debe de tratar los planes por separado [34]. Sumado a lo anterior se debe de agregar la parte de las preferencias, las cuales requieren de un espacio de memoria para poder llevar un control de estas,

debido al uso de vectores correspondientes a cada una.

Por otro lado tenemos el algoritmo A* [61] [4] [41] el cual es muy similar al algoritmo de Dijkstra [19] [35] [62] descrito anteriormente, pero este se compone de dos partes, el valor actual acumulado para ese vértice y de un valor potencial para alcanzar el objetivo a partir del vértice actual (heurística, se basa en una estimación geográfica que se pudiera representar y calcular adecuadamente) [33]. A* mantiene dos estructuras de datos auxiliares, que podemos denominar abiertos, implementado como una cola de prioridad, y cerrados, donde se guarda la información de los nodos que ya han sido visitados.

En cada paso del algoritmo, se expande el nodo que esté primero en abiertos (en cola de prioridad), y en caso de que no sea un nodo objetivo, calcula los pesos de todos sus hijos, los inserta en abiertos, y pasa el nodo evaluado a cerrados. En general este algoritmo garantiza la optimalidad del resultado arrojado [41], siempre y cuando la función heurística nunca sobreestime el valor de la distancia real entre el nodo y el destino. Para mejorar este algoritmo A* se hace uso de límites inferiores, es decir, se limita el costo de la parte heurística (por ejemplo, distancia euclidiana) [33].

Su limitación en el transporte público se centra en el manejo y control de las preferencias de los usuarios, ya que se necesitaría vectores donde almacenar estas y comparar con el movimiento a realizar, afectando a la parte heurística de este algoritmo.

Explotación Jerárquica.

En los trabajos por Sanders and Schultes [58] y Funke at et [6], se utilizan técnicas desarrolladas para grafos planos, una de estas técnicas es el separador basado en el método multi-nivel, la idea es la partición del grafo en pequeños para remover un conjunto de nodos de separación. Estos nodos de separación junto con los arcos representan caminos precalculados entre estos y constituyen un siguiente nivel.

El diseño en que la mayor parte de los artículos leídos, se basan para reducir el

espacio de búsqueda en la jerarquización de carreteras [57], [56], [58], lo cual consiste en la jerarquía de arcos y nodos de una red de carreteras. Esta jerarquía se basa en el grado de importancia o del peso específico de los posibles caminos a seguir, es decir se le da mayor prioridad de búsqueda a una avenida que a una calle pequeña, este método se alterna entre dos procedimientos:

Contracción: Remueve todos los nodos de grado menor.

Reducción de arcos: Define los vecindarios para cada nodo, cuyo límite puede ser circular o cuadrado, si el arco se encuentra fuera de ese rango es eliminado.

El algoritmo de consulta es muy similar a la búsqueda del Dijkstra Bidireccional con la diferencia de que ciertos arcos no son necesarios expandir cuando la búsqueda es suficientemente lejana del destino. El problema con el método es que al reducir la búsqueda se limita las rutas a recorrer por parte del usuario, debido a que le dará prioridad a desplazarse por las rutas de mayor peso y eliminando las de menor, las cuales podrían ser de menor costo, tendiendo así a repercutir en distancias muy grandes a recorrer en relación a alguna eliminada, y para nuestro caso, quizá no se logre computar ni un plan factible en base a las preferencias del usuario respecto a la longitud recorrida caminando.

Gutman [34], propone un algoritmo de enrutamiento basado en alcance, donde dice que una ruta arbitraria puede ser formada por un número pequeño de sub-rutas previamente computadas. La partición permite un intercambio entre precalculos y tiempos de consulta procesados.

Para ello propone un concepto general de niveles de arco. Considera un arco e en medio del camino más corto, con respecto al tiempo de viaje entre dos nodos que tienen una distancia d de separación por ejemplo una distancia euclidiana. Gutman define los niveles con respecto a la distancia euclidiana, pero define que se puede utilizar cualquier otra métrica.

También presenta algoritmos sencillos que calculan cotas superiores para el nivel del arco y debido al uso de distancias euclidianas, su enfoque permite utilizar

diversas variantes del algoritmo de Dijkstra, esto para encontrar rutas más cortas exactas sobre una red de carreteras moderada.

Esta propuesta de Gutman [34] ofrece las siguientes ventajas: Garantiza optimalidad, el tiempo de cálculo del camino más corto es comparable con el enfoque industrial, puede ser combinado con otros algoritmos, el requerimiento de almacenaje no se ve afectado por los datos de precalculo, puede manejar dinámicamente cambios en el grafo de forma rápida. Reduce grandemente el cálculo para múltiples orígenes y destinos. Se ve limitado al uso de las preferencias a manejar o deseadas por parte del usuario ya que considera que este tiene su propia forma de desplazarse.

Sanders and Shultes [56] combinan nodos de tránsito [6], con jerarquía de carreteras [57] [56] [58]. Esta combinación alcanza preprocesamientos y procesamientos extremadamente rápidos al momento de generar una solución, al mismo tiempo se incrementa el costo computacional debido al uso de algoritmos complejos y al alto consumo de memoria. También hace mención de que una búsqueda bidireccional no puede ser utilizada para transporte público. Ésta ineficiencia creemos que se debe a la incapacidad de trabajar con rutas predefinidas, ya que sería bastante costoso que se encontraran las dos direcciones de la búsqueda, además que si se desea considerar el costo esta técnica lo imposibilita.

Golderberg [33], combina los niveles de las aristas con un esquema de comprensión y utiliza límites inferiores, basado en las distancias precalculadas para algunos puntos de referencia, para permitir una búsqueda más dirigida a un objetivo. Tiene un alto consumo de memoria ya que necesita registrar para todos los nodos en la red las distancias a todas las etiquetas [5].

Otra de las implementaciones que se encontró en la literatura es el tránsito de nodos [6], donde se precálcula tablas de distancia para nodos importantes, así también para conexiones relevantes entre los restantes nodos, para esto último requiere de búsquedas locales en su mayoría con un algoritmo de Dijkstra simple.

Los tiempos de solución de los algoritmos basados en tránsito de nodos son muy

buenos para todos los trabajos presentados y en si se basan en tablas de búsqueda, es decir almacenan rutas previamente calculadas para usarlas en un futuro. Sin embargo, estos algoritmos solo se utilizan para cómputo de rutas de viaje en mapas continentales, no para el transporte público. Aunque estas ideas son válidas para este tipo de aplicaciones, solo serviría para calcular la ruta más corta, lo cual es tan solo una de las preferencias que el usuario desea al momento de decidir hacer uso de esta forma de transporte.

Teniendo que la complejidad de este método se debe al precomputarizar diferentes tablas, dadas diferentes métricas, y después del cómputo de las tablas se presente otro problema que es cómo llevar a cabo la combinación de métricas para satisfacer lo requerido por el usuario (por ejemplo, planes que involucren menos tiempo y menos costo).

Chao-Lin Liu [44] ataca el problema de planes de viaje para el transporte público asumiendo que no se pueden modificar las rutas del transporte a criterio del usuario, debido a que son definidas. Él propone el uso de matrices para solucionar este problema, ya que de este modo se pueden capturar las restricciones correspondientes al sistema, las matrices propuestas son:

Matrices de planificación de rutas.

- Matrices de adyacencia.
- Matrices de conectividad.

Planificación con matrices de transición.

- Matrices de planificación.
- Planificación bajo rutas restringidas.
- Integración con algoritmos de ruta más corta.

Donde concluye, en base a los resultados que obtuvo, que el método de matrices de transición es el más viable para la generación de planes de viaje para el usuario, utilizando el transporte público (solo autobuses), pero no se muestran resultados para un mayor entendimiento de lo propuesto, utiliza el algoritmo de Dijkstra para computar la solución.

El algoritmo solo toma en cuenta las rutas de autobuses para generar el recorrido a realizar por el usuario, más sin embargo no considera la parte de donde el usuario tenga que desplazarse caminando para alcanzar un punto en la red y esto es importante ya que incurre directamente en los costos de recorrido. Aunque su idea de trabajar con matrices puede ser muy buena para nuestro problema ya que primero se podría encontrar posibles rutas y ya después determinar cuáles de ellas cumplen con los requisitos del usuario y complementar con la parte donde se debe de caminar. Este algoritmo permite minimizar el número de transferencias.

Un ejemplo de un sistema aplicado al transporte público es el de ruta directa [20] aplicado a varias ciudades de México, donde permite al usuario ver el recorrido de las diversas rutas del sistema de transporte, que rutas pasan por un punto determinado y como alcanzar un destino a partir de un punto de origen, ofreciendo varias opciones al usuario para realizar su viaje usando el sistema de transporte urbano (solo autobuses) si es que existen.

El sistema es relativamente bueno, pero muestra únicamente que rutas pasan por los puntos involucrados en el viaje. Hace un simple cómputo de las rutas que se traslapan, y convergen en el punto de origen, en el destino así como en algún punto intermedio. Obviamente, esto no es un plan de viaje porque no está optimizando ninguna métrica, ni siquiera la distancia a recorrer o el tiempo. Esto se puede ver en la figura 2.2.

Generación de planes de viaje utilizando Algoritmos de Planificación.

Pare éste punto se encontró muy poca información al respecto, creemos suponer que esto es debido a que la aplicación de técnicas de planificación en el tema del

Figura 2.2: Resultado mostrado por Ruta Directa [20].

transporte público es un área de investigación reciente, ya sea en la estandarización de lenguajes de planificación, así como también, de los planificadores, estos últimos permiten dar una solución al problema como bien se describió anteriormente.

Entre los trabajos encontrados bajo este esquema tenemos el propuesto por Hrnír [38] [39], donde propone la utilización de agentes para la generación de planes de viajes, esto para el sistema de transporte del Reino Unido. En este trabajo combinan autobuses de pasajeros del transporte público, sistemas de trenes y recorridos caminando, ellos modelan su problema para un grupo de personas con distintos puntos de origen o diversos destinos, pero siempre viajando en conjunto en algún momento, en su solución cada agente representa un plan de viaje, donde estos planes no se interponen entre sí.

Su algoritmo se compone de tres fases, las cuales son [38] [39]:

Fase inicial: Una ruta de viaje es encontrada para cada agente usando un plan relajado.

Fase BR: Mejora el plan inicial para un agente utilizando el algoritmo BRP (best-response planning), esto se logra utilizando las conexiones con otros agentes.

Fase timetabling: Los recorridos optimizados compartidos se comparan con los horarios utilizando un solo agente planificador que asume el dominio completo.

En este trabajo se tiene como función objetivo la minimización del tiempo de recorrido, y como trabaja por grupos de n personas, engloban el costo económico de viaje como un solo pago proporcional al número de usuarios.

Lo presentado en este trabajo es válido para planificar viajes de grupos de usuarios que parten desde un mismo punto de origen hacia un mismo destino, nosotros podemos ampliar nuestro modelo para varios usuarios en donde se combinen diversos puntos de origen y de destino. El tiempo computacional de solución reportado tiende a ser alto para la generación de estos planes.

La principal diferencia encontrada con este trabajo es que ellos no consideran las preferencias de los usuarios como restricciones al momento de computar una ruta de viaje, además que solo optimizan la métrica de tiempo de duración de recorrido, mientras nosotros proponemos el uso de algunas otras más que el usuario busca o desea.

Por otro lado se presenta TIMIPLAN [21] un trabajo de logística multitransporte, cuyo dominio podría adaptarse a lo que estamos desarrollando debido al uso del transporte para llevar objetos entre dos puntos minimizando costos. Lo que ellos proponen es descomponer el problema en dos sub-problemas, donde el primero es un problema de asignación, donde se toma para cada ruta un camión (es) y contenedor (es) con el menor costo estimado (sincronización de pasajero-autobús en la parada). El segundo sub-problema es la planificación de la logística, para alcanzar cada servicio individual minimizando los costos (traslado del pasajero entre dos puntos). Para poder solucionar esto se propone la utilización de planificadores y del lenguaje PDDL. La complicación encontrada por estos desarrolladores es que no soporta grafos grandes.

En [9], se introduce el problema de generar planes de viaje de visitas turísticas a centros de interés que el usuario desea visitar (como bares, construcciones, áreas verdes, étc.). Se considera el tiempo disponible del visitante, y se generan los planes de acuerdo a los puntos que son más visitados, según el rubro deseado, optimizando

el tiempo de recorrido, también brinda información del costo económico del plan.

Ellos hacen uso de agentes para solucionar este problema, así como también del lenguaje PDL [15]. Este tipo de técnicas pudieran ser utilizadas o son complementarias de lo que nosotros proponemos, ya que busca el mejor plan posible de acuerdo a las necesidades y/o preferencias de los turistas, con la diferencia que ellos solo consideran puntos con mayor grado de visitas y no puntos intermedios.

Por último tenemos un trabajo realizado por [23], en cuya tesis habla de la utilización del planificador SHOP2 (Simple Hierarchical Ordered Planner 2) [52], y del lenguaje HTN-PDDL (Hierarchical Task Network) [52] para la generación de planes de viaje utilizando el transporte público de la ciudad de Madrid, España, el cual incluye rutas de autobuses y el sistema del metro. Ellos modelan las acciones necesarias para este dominio y solo tienen una función objetivo, la cual es minimizar el tiempo de recorrido. Su implementación muestra las rutas gráficamente, sin embargo ellos argumentan que los resultados obtenidos no son los deseados debido a que el tiempo computacional requerido para la generación de los planes de viaje son muy altos y no consideran las diversas preferencias de los usuarios, además que el modelo que proponen no soporta todo el ambiente del sistema que desean representar, es decir, empiezan hablando de diversos medios de transporte, pero al final solo tuvieron soporte para representar las líneas del metro de la Ciudad de Madrid.

Por ello nos atrevemos a decir que nuestro modelo tiene ventajas sobre el propuesto en [23], ya que estamos considerando un mayor número de elementos que conforman una red de transporte, (paradas, autobuses, carreteras), además ellos solo presentan un modelo sin restricciones donde nuestros resultados obtenidos de la experimentación con nuestro modelo básico se observa que somos competitivos, solo que nosotros utilizamos mayor número de elementos, y sobre todo de restricciones o preferencias por parte del usuario para el computo del plan.

2.4 Similitudes, diferencias y oportunidades con respecto al estado del arte

Tomando en cuenta lo presentando en el punto anterior, todos los algoritmos y el nuestro tratan de brindar una mejor solución al usuario en el cómputo de una ruta de viaje, donde se fotografía toda una red de caminos los cuales se representan mediante arcos y los nodos indican puntos de interés o esquinas de las diversas cuadras. Sin embargo, nuestro modelo se diferencia de los demás debido a las técnicas de solución adoptadas, ya que estas permiten la incersión de acciones, las cuales son esenciales en el desarrollo de un plan, es decir, en los sistemas actuales solo indican una ruta, pero no así los movimientos explícitos para llevar a cabo el plan. Además nuestro modelo permite distinguir quíenes hacen dichos movimientos, es decir, permite la acción de múltiples actores, sumado a esto podemos conocer el costo por llevar a cabo dichas acciones.

En los trabajos realizados con anterioridad no se consideran preferencias y/o requerimientos de los usuarios, salvo el presentado en [9] (solo algunas pero para viajes grupales), sumado a esto los algoritmos existentes son válidos cuando el usuario tiene su propio medio de desplazamiento dentro de la ciudad. Se tiene para estos modelos un número grande de posibles rutas a seguir para poder alcanzar el objetivo, esto no sucede en el transporte público debido a la restricción de rutas definidas.

Si tomamos en cuenta los sistemas de planificación mencionados en el estado del arte, existen algunos que son válidos para coordinar un plan para grupos de personas, en el caso que manejamos se podría generar planes para diversas personas, tomando para la realización de cada uno de ellos las preferencias personales de los usuarios, teniendo así planes diferentes para este conjunto de usuarios sin tener que coordinarlos.

Es por ello que tenemos grandes oportunidades de aportación con esta investigación que va desde identificar qué factores limitan a los planificadores, como el

ingreso de nuevas variables a optimizar y de restricciones en el cómputo de planes de viaje, en base a lo que el usuario desea, ya que ellos son quienes hacen uso del sistema de transporte público.

Se podría tomar la idea presentada en [9] de considerar puntos de interés o de mayor concurrencia, teniendo entre ellos planes precalculados almacenados en tablas de búsqueda reduciendo con su uso el tiempo de espera por parte del usuario al momento de realizar la consulta.

2.5 Modo de solución

En nuestro proyecto de generación de planes de viaje para el usuario a partir de sus preferencias usaremos técnicas de planificación para la obtención de dichos planes. Para esto llevamos a cabo un análisis del sistema de transporte con la finalidad de identificar las propiedades y variables que lo conforman. Tomaremos aquellas componentes controlables a nuestros intereses.

En el diseño del modelo consideramos las propiedades y/o elementos que nos es permisible representar del sistema de transporte en PDDL-2.1. Un plan se basa en acciones para llevarlo a cabo, por ello en la parte de diseño se debe considerar qué acciones se requieren y cuáles podemos usar de acuerdo a las propiedades y variables necesarias para su realización, lo cual nos encamina al desarrollo de un modelo de planificación.

Se generó varios modelos de planificación donde se representan las diversas preferencias del usuario (con una o varias métricas) esto para dar solución al problema de planificación de rutas de transporte público para el usuario. Debemos modelar la red del transporte público a partir de los elementos encontrados factibles.

Una vez realizado el modelo de planifación se procede a generar una solución mediante el uso de planificadores, los cuales serán seleccionados de acuerdo a su compatibilidad con el tipo de problema que estamos abordando, es decir, en base a

los requerimientos de nuestro modelo.

Finalmente, se desarrollará y se presentará una implementación del Algoritmo de Dijsktra modificado, esto para hacer comparativas (debido a que es un método exacto) con un modelo básico de planificación, donde se considera solo las acciones de movimientos entre nodos (tiempos y distancias), esto para minimizar el tiempo de recorrido en base a las preferencias de distancia a caminar. Todo esto será presentado en el siguiente capítulo.

2.6 Planificación

Nuestro trabajo genera planes de viaje para el usuario en base a sus necesidades o preferencias. Pero para poder abordar el problema debemos conocer a qué nos referimos con planificación, a continuación se presenta su definición de acuerdo al contexto de la Inteligencia Artificial [54]:

"La planificación es el proceso de búsqueda y articulación de una secuencia de pasos que permitan alcanzar un objetivo a partir de un estado inicial en un espacio de acciones que para poder llevarse a cabo se requiere de condiciones para causar un efecto en el sistema."

Para poder generar estas soluciones (planes) se requiere del uso de planificadores, los cuales se presentan en puntos posteriores.

Ideas clave para la planificación [54]:

La primera de las ideas clave para la planificación es la de la "apertura" de la representación de estados, metas y acciones. Los estados y metas se representan mediante conjuntos de oraciones; las acciones, mediante descripciones lógicas de condiciones previas y efectos. Esto permite al planificador establecer conexiones directas entre estados y acciones.

La segunda clave fundamental consiste en que el planificador es libre de añadir

acciones al plan, en vez de hacerlo de forma gradual empezando por el estado inicial. Al tomar primero decisiones "obvias" o "importantes" el planificador puede disminuir el factor de ramificación correspondiente a elecciones futuras y disminuir la necesidad de retroceder a decisiones arbitrarias.

La tercera y última clave de la planificación consiste en que la mayoría de las partes del mundo son independientes de otras partes.

La consecuencia de esto, es que se puede separar problemas, para resolverlos de una manera más fácil. En nuestro caso, podríamos separar nuestro problema en dos: a) el problema de encontrar un plan para el usuario para llegar de su origen al lugar donde va a tomar el camión, b) y el plan estimado de los autobuses.

Los problemas de planificación se representan mediante oraciones lógicas que describen las tres partes principales de un problema [54]:

Estado inicial: Una condición lógica arbitraria acerca de una situación S_0. Como ejemplo tenemos el problema donde se está en casa y no se tiene ni leche, ni plátanos y ni un taladro, esto queda denotado de la siguiente manera:

En(casa, So)$\wedge \neg$ *Hay(Leche,So)*$\wedge \neg$ *Hay(Plátanos,So)*$\wedge \neg$ *Hay(Taladro,So).*

Estado meta: Una consulta lógica para encontrar situaciones adecuadas. En el ejemplo de las compras, la consulta sería:

$\exists$ *s En(casa, s)*$\wedge$ *Hay(Leche,s)*$\wedge$ *Hay(Plátanos,s)*$\wedge$ *Hay(Taladro,s).*

Operadores: Conjunto de la especificación de acciones.

Los estados [54] se representan mediante conjunciones de literales básicas en las que no hay funciones, es decir, predicados aplicados a signos constantes, posiblemente negados, la especificación de un estado no necesariamente tiene que ser completa. Una especificación así, corresponde a un conjunto de estados completos posibles para el que el agente desearía obtener un plan satisfactorio. Por otra parte, en muchos sistemas de planificación se adopta la convención de que si en la descripción del

estado no se menciona una determinada literal positiva entonces se toma como falsa.

2.6.1 Plan

Ya se ha hablado de planificación, de sus ideas claves, de sus estados, pero todo esto nos lleva a la obtención de un plan, el cual es la solución obtenida y que se presenta al consultante, a continuación mostraremos su definición esto de forma coloquial:

"Un plan es una secuencia de pasos a partir de un estado inicial que permite alcanzar un estado final."

Sin embargo, a un plan se le define formalmente como una estructura de datos constituida por los cuatro elementos siguientes [54]:

- Un conjunto de acciones A del plan. Cada uno de éstos es uno de los operadores del problema.
- Un conjunto de restricciones para el ordenamiento de las acciones. Las restricciones de ordenamiento tienen la forma $a_i \prec a_j$, es decir que la acción a_i debe producirse en algún momento antes de la acción a_j.
- Un conjunto de restricciones para enlace de variables. Cada restricción de variable tiene la forma $v{=}x$, en donde v es una variable en alguna de las acciones y x es una constante u otra variable.
- Un conjunto de vínculos causales, los cuales sirven para llevar un registro del(os) objetivo(s) de las acciones del plan.

El plan inicial, sencillamente describe el plan no completo. Consta de dos acciones, denominadas *Iniciar* y *Terminar*, regidas por la restricción de ordenamiento $Iniciar \prec Terminar$. La acción *Iniciar* no tiene condiciones previas y su efecto consiste en añadir todas las preposiciones que sean verdaderas en el estado inicial. En la

meta de la acción *Terminar* el estado meta es su condición previa y no tiene efecto alguno [54]. Se puede decir que se alcanzó una solución cuando se tiene un plan completo, consistente. Se define los términos anteriores a continuación [54].

Un plan completo es aquel en el que cada condición previa de todos las acciones se logra mediante otra acción. Una acción logra una condición si ésta es uno de los efectos producidos por la acción, y si ningún otra acción tiene posibilidad de eliminar la condición. De manera más formal, una acción a_i logra una condición previa c de la acción a_j si (1) $\prec a_j$ y $c \in$ EFECTOS(a_i); y (2) si no hay una acción a_k tal que $\neg c \in$ EFECTOS(a_k, en donde $a_i \prec a_k \prec A_j$) en alguna linealización del plan (secuencia paso a paso).

Un plan consistente es aquel en el que no hay contradicciones en las restricciones de ordenamiento o enlazamiento $a_i \prec a_j$ como $a_j \prec a_i$ se cumplen, pero no al mismo tiempo los dos. Tanto $\prec$ como $=$ son positivos, es decir que no son negados y, por ejemplo, un plan en el que $a_1 \prec a_2$, $a_2 \prec a_3$, y $a_3 \prec a_1$ es inconsistente.

2.7 Lenguaje de planificación PDDL-2.1

PDDL (Planning Domain Definition Language) es el lenguaje estándar para la codificación de modelos de planificación, el cual fue desarrollado por Drew McDermott y dado a conocer en la International Planning Competition de 1998 [1] [32].

2.7.1 Descripción del lenguaje

- PDDL. Es un lenguaje centrado en las acciones, inspirado en las formulaciones STRIPS de problemas de planificación.
- Es una estandarización de la sintaxis para expresar acciones utilizando precondiciones y post-condiciones para describir la aplicabilidad y efectos de las acciones.

El lenguaje soporta las siguientes funciones sintácticas [32]:

- Estilo STRIPS (representa los estados mediante conjunciones de literales básicas en las que no hay funciones (métricas)).
- Efectos condicionales (funciona como una sentencia if del lenguaje de programación C, esto para condicionar los efectos de una acción determinada).
- Cuantificadores universales sobre universos dinámicos (i.e., implicaciones lógicas sobre ellos.)
- Especificación de restricciones (condiciones necesarias para poder llevar a cabo una acción).
- Especificación de acciones jerárquicas compuestas de sub-acciones y sub-objetivos (división del problema general en subproblemas llamados acciones que constan de condiciones y efectos, estableciendo con esto sub-objetivos para cada uno de estos sub-problemas).
- Administración de múltiples dominios usando diferentes subconjuntos de características del lenguaje (robustez).

Para el soporte de nuestro modelo necesitamos de las propiedades anteriormente mencionadas y de otras presentadas en la versión PDDL-2.1 [22] ya que requerimos del uso de métricas, las cuales son necesarias para las funciones a optimizar, así como para las preferencias de los usuarios. Desde un punto de vista las preferencias de los usuarios son restricciones de nuestro modelo al computar un plan de viaje, a este requerimiento se le llama *Fluents* [22], el cual indica al planificador que debe de soportar cantidades númericas. Cada acción tiene un tiempo de duración en su realización definido como *durative-action* en [22] y permite darle valores númericos a las duraciones de las acciones en vez de ser instantáneas.

```
(define (domain <nombre del dominio>)
    (:requeriments
    (:predicates
    (:functions
    (:Actions <nombre de la función o valor constante>
)
```

Figura 2.3: Estructura de un dominio de planificación en PDDL-2.1 con métricas.

2.8 Dominio de planificación y archivo problema

Para poder solucionar problemas utilizando modelos de planificación requerimos de dos archivos: el dominio de planificación y el archivo problema. Donde el archivo de domino representa el conjunto de operadores permisibles, sus reglas de aplicación, así como su función de transición (efectos) de estados. Mientras que el archivo problema representa las condiciones iniciales y los objetivos finales que se utilizarán y para los que hay que planificar.

2.8.1 Dominio de planificación

El dominio de planificación es el archivo donde se describe el mundo que se desea modelar, sus características, valores, tipo de constantes, formas de transicionarlo, étc., y una parte de este mundo comprende a los operadores necesarios que son las formas de cómo podemos afectarlo y actualizarlo para la generación del plan, en base a ciertas condiciones. Un dominio de planificación donde se utilizan métricas, queda estructurado en la figura 2.3. Estas partes que conforman el dominio de planificación se definen a continuación:

(define (domain < nombre del dominio >): Define el nombre del dominio.

Requeriments: Son las propiedades que distinguen al dominio en turno, estableciendo cuáles serán las características del dominio que se pretende describir. Presentamos un ejemplo de cómo se declaran estos *requeriment*, ver figura 2.4 donde

```
(:requeriments :numeric-fluents :durative-actions)
```

Figura 2.4: Representación de *requeriment* en un dominio de planificación en PDDL-2.1.

```
(via_bus ?r ?bus ?de ?a)
```

Figura 2.5: Representación de predicates en un dominio de planificación con PDDL-2.1.

el primer *requeriment* es para indicar que se debe de soportar entidades numéricas y la segunda para las acciones durativas.

Predicates: Son aquellas propiedades de los objetos que nos interesan [45]. Es una definición abstracta de todas las proposiciones atómicas utilizadas en el dominio y que se utilizan para representar conceptos o relaciones entre los objetos. Los predicados únicamente toman valores Boleanos, es decir, pueden existir o no existir en el mundo. La etiqueta define los nombres de los predicados y el número y tipo de parámetros [8]. Como ejemplo tenemos la figura 2.5, donde *via_bus* indica el nombre del predicado mientras que *?r*, *?bus ?p1* y *?p2* hacen referencia a los objetos pasajero, unidad, origen y destino, respectivamente. Este predicado *via_bus* expresa lo siguiente el pasajero *?r* puede desplazarse en al autobús *?bus* desde *?p1* a *?p2*.

Functions: A diferencia de las proposiciones atómicas *(predicates)*, que en un mapa de interpretación sean verdaderas o falsas, las funciones se interpretan númericamente, es decir es una métrica [8], o sea, es una relación de un concepto abstracto a uno numérico. Esto se muestra en la figura 2.6, donde la primera línea muestra cómo se define una *function* en el archivo dominio, en este caso aquella relacionada con el tiempo de recorrido de un autobús del punto *?de* al punto *?a*, en la segunda línea se muestra como está escrita esa misma *function* solo que ahora se indica qué unidad es y entre qué nodos se puede desplazar seguido de un valor numérico que indica el tiempo correspondiente de desplazamiento entre los nodos *c70* a *c80*.

Actions: Las acciones son operadores que basan sus descripciones lógicas en

(tiempo_b ?bus ?de ?a)

(tiempo_b ?bus c70 c80)11)

Figura 2.6: Representación de funciones en un dominio de planificación en PDDL-2.1.

precondiciones y efectos, generando descripciones completas de los estados resultantes después de su aplicación. Las acciones se están conformadas por tres partes [8] [22]: Lista de parámetros, precondiciones y efectos.

Notación de las acciones [54]:

- La acción se identifica por el nombre y su lista de parámetros, ver figura 2.7, donde el nombre de la acción es definida como *Desplazar_bus*, mientras que los parámetros son los objetos necesarios para realizar la acción correspondiente.
- La precondición es una lista de literales que dice qué debe ser verdad antes de poder aplicar el operador, en la figura 2.7 se muestran en precondition, las precondiciones necesarias, particularmente se requiere que exista una ruta entre el punto *?de* hacia *?a* para el autobús *?bus* ubicado en *?de*.
- El efecto de un operador es una conjunción de literales (positivas o negativas) que dice de qué manera cambia la situación al aplicar el operador, esta anotación se ve en la figura 2.7, donde se indica que el autobús *?bus* ya no se encuentra en la pocisión *?de* debido a que se recorrió a *?a*.

Para el problema que estamos abordando haremos uso de acciones durativas, es decir, tienen un peso asociado de ejecución, resultando en un plan temporal. Por ejemplo, esto se presenta cuando se desea viajar entre dos puntos y hacer tal acción tiene un tiempo de duración. Además, en la construcción de un plan temporal las acciones pueden solaparse por lo qué será necesario tener en cuenta el instante en que se producen los efectos positivos y negativos de las acciones para la consecución de un plan temporal válido [26], estas acciones durativas son de dos tipos: discretas y continuas, en nuestro caso trabajaremos con acciones durativas discretas.

```
(:action Desplazar_bus
        :parameters(?bus ?de ?a)
        :precondition(and
                (autobus ?bus)
                (via_bus ?bus ?de ?a)
                (ruta_bus ?bus ?de)
        effect(and
                (not(ruta_bus ?bus ?de))
                (ruta_bus ?bus ?a))
)
```

Figura 2.7: Representación de las acciones en un dominio de planificación en PDDL-2.1.

Las *durative-action* no solo tienen efectos que se fijan hasta el final de la acción, sino también al inicio [25], esta estructura se puede ver en la figura 2.8. De donde las *duration* puede ser de tres formas según lo explica García Olaya [24]:

a. La duración puede ser dada por un valor numérico.

   (=?duration 5).

b. Puede ser calculada usando fluents.

   *:duration (=?duration (*2(ítem-weight ?w))).*

c. Puede ser vacía.

En la figura 2.8 se presenta la estructura de una acción durativa, la cual no cambia demasiado en relación a la estructura presentada anteriormente de una acción simple.

Las condiciones de las *durative-action* se dividen en tres grupos: las que son garantizadas hasta el inicio de la acción, el conjunto de condiciones invariantes durante la ejecución de la acción, y aquellas que son garantizadas hasta el final de la

```
(:durative-action <nombre de la acción>
        :parameters (<parámetros>)
        :duration (  duración de la acción)
        :condition( condiciones para llevar a cabo la acción>)
        :effect (<efectos>)
)
```

Figura 2.8: Estructura de una *durative-action* en un dominio de planificación en PDDL-2.1.

```
(:durative-action load-truck
        :parameters (?t - truck)
                (?l - location)
                (?o - cargo)
                (?c - crane)
        :duration (= ?duration 5)
        :condition (and (at start (at ?t ?l))
                (at start (at ?o ?l))
                (at start (empty ?c)
                (over all (at ?t ?l))
                (at end (holding ?c ?o))
        :effect (and (at end (in ?o ?t))
                (at start (holding ?c ?o))
                (at start (not (at ?o ?l)))
                (at end (not (holding ?c ?o)))
)
```

Figura 2.9: Ejemplo de una *durative-action* en un dominio de planificación en PDDL-2.1 [22].

acción. Por otro lado, los efectos se clasifican en dos grupos: los que son afirmados al inicio de la acción y los que lo son al final de la misma, estos pueden ser positivos y/o negativos. La duración de la acción es un valor positivo [25], [14].

En la figura 2.9 se puede ver aquellas proposiciones que se necesitan mantener al inicio del intervalo *start* (el punto donde la acción es aplicada), al final del mismo *end*, y aquellas que se requieren mantener durante todo el intervalo de acción *over all*.

En la sección de las *condition* nos dice que necesitamos al inicio de la acción que el transporte *?t* se encuentre ubicado en la posición *?l* y que la carga *?o* este en la misma posición que el transporte, y que la grúa *?c* este vacía, esto queda representado del siguiente modo respectivamente *(at start (at ?t ?l)), (at start (at ?o ?l)) y (at start (empty ?c))*. Para ello se requiere mantener durante todo el lapso de tiempo que el transporte *?t* este en *?l*, es decir *(over all (at ?t ?l))*.

Mientras que para los *effects* tenemos que hasta que se finalice la acción load-truck estará la carga *?o* en el transporte *?t*, esto queda *(at end (in ?o ?t))*. Los dos siguientes *effects* como se puede ver se llevan a cabo al inicio de la acción, el primero de ellos nos indica que la grúa *?c* sostiene a la carga *?o (at start (holding ?c ?o))*, la segunda conditions indica que la carga *?o* ya no se encuentra en ubicación *?l*, *(at start (not (at ?o ?l)))*. La sentencia *(at end (not (holding ?c ?o)))*, es para indicar en la acción que al termino de ella, la grúa *?c* no sostiene más la carga *?o*.

2.8.2 Archivo problema

El archivo problema es donde se declaran los objetos involucrados en nuestro problema de planificación, así como también los estados iniciales de estos objetos, por otro lado es en esta parte donde se asignan los pesos (costos) a las *function* correspondientes y que son declaradas en el dominio de planificación, este archivo problema está compuesto en su estructura como se muestra en la figura 2.10.

En la figura 2.11 se puede ver cómo se compone cada una de las partes del

```
(define(problema <nombre del problema>)
(:domain <nombre del dominio>)
    (:objects
    (:init
    (:goal
    (:metric <minimize or maximece> <variable a optimizar>)
)
```

Figura 2.10: Estructura de un archivo problema en un dominio de planificación con PDDL-2.1.

```
(define(problem ruta8)
     (:domain ruta-transporte)
(:objects
pas1
 ruta_1
 ruta_2
 c80 c70 c69 c68 c67 c57 c56 c55 c54 c53 c52 c51 c50
(:init
(pasajero pas1)
(autobus ruta_1) (at pas1 c80)

(via_bus pas1 ruta_1 c80 c70)
(=(tiempo_b pas1 ruta_1 c80 c70)11)
(via_bus pas1 ruta_1 c70 c69)
(=(tiempo_b pas1 ruta_1 c70 c69)10)

                .
                .
                .
(via_bus pas1 ruta_1 c51 c50)
(=(tiempo_b pas1 ruta_1 c51 c50)8)


(:goal (and (at pas1 c50)))
(:metric minimize(total-time))
)
```

Figura 2.11: Representación de un archivo problema en un dominio de planificación con PDDL-2.1.

archivo problema implementado en PDDL-2.1. Donde *problem ruta8* es el nombre del archivo, seguido del nombre del archivo dominio con el cual se resolverá, en este caso es ruta-transporte, los objetos como bien se mencionó anteriormente son las partículas de nuestro mundo, para el ejemplo de la figura 2.11 tenemos *pas1*, *ruta_1*, *c80*, *c70*, étc., los cuales en orden de aparición representan al pasajero, a la unidad de transporte y los nodos o paradas a alcanzar.

Siguiendo la figura 2.11, en los *init* se muestra el estado inicial, *(pasajero pas1)* y *(autobus ruta_1)* solo para indicar con que elementos se cuenta para ese predicado, en *(at pas1 c80)* indica la ubicación inicial del usuario, *(via_bus pas1 ruta_1 c80 c70)* y *(=(tiempo_b pas1 ruta_1 c80 c70)11)* muestran de donde a donde se puede desplazar la unidad con un pasajero, y cuanto tiempo le toma hacer ese desplazamiento, todo esto en orden de aparición.

(:goal (and (at pas1 c50))) nos dice que el objetivo es que el pasajero se encuentre en el nodo *c50*, mientras que en *(:metric minimize(total-time))* muestra la variable a optimizar *total-time* donde se indica que debe ser minimizada.

2.9 Algoritmos de planificación

Cuando el estado del mundo es accesible, el agente de planificación emplea las percepciones que da el ambiente para construir un modelo completo y correcto del estado del mundo correspondiente a ese momento. Si se le plantea una meta, el agente emplea el algoritmo de planificación para elaborar un plan de acción. Realizado éste, el agente procede a ejecutar los pasos que se indique, uno a la vez [54]. Por lo tanto, en éste apartado describiremos los algoritmos de planificación que usaremos para la creación de planes para el problema dado en ésta investigación.

2.9.1 Marco del Agente Planificador Simple [54]

En la figura 2.12 se muestra el algoritmo del agente planificador simple. Supondremos que se cuenta con una función DESCRIPCIÓN-ESTADO, la que se alimenta con una percepción y produce la descripción del estado inicial en el formato que requiere el planificador. Con la función HACER-CONSULTA-META, que sirve para preguntar a la base de conocimiento cuál es la siguiente meta. Note que el agente deberá ser capaz de manejar casos en los que la meta planteada no es factible, y también el caso en el que el plan completo está vacío, dado que la meta ya se cumple desde el estado inicial. El agente interactúa con el ambiente en forma mínima; se vale de las percepciones para definir el estado inicial y, basado en éste, la meta inicial; después de esto, se limita a dar los pasos marcados por el plan que él mismo elaboró.

2.10 Planificadores

Un planificador es un sistema que a partir de modelos de planificación formalmente validados encuentra una solución (plan) a un problema.

Definición de planificador [54]:

"Es un algoritmo de propósito especial, el cual utiliza un lenguaje de planificación formal con una sintaxis, semántica y teoría de la demostración bien definidas, el cual nos permite generar planes."

La optimilidad de un planificador depende del algoritmo mismo y de la heurística a utilizar. Es decir, existen planificadores que garantizan optimilidad dados estos dos criterios. Nosotros consideraremos funciones heurísticas no óptimas ya que estas son más eficientes en obtener una solución satisfactoria debido a la necesidad de pronta respuesta a los usuarios.

```
función AGENTE-PLANIFICADOR-SENCILLO (percepción) responde con una acción
    estático: BC, una base de conocimiento (incluye la descripción correspondiente de las acciones)
        p, un plan, originalmente NoHayPlan
        t, un contador, originalmente 0, para marcar tiempo
    variables locales: M, una meta actual,
                una descripción del estado actual

    DECIR (BC, HACER-OPERACIÓN-DE-RECEPCIÓN (percepción, t))
    Actual ← DESCRIPCIÓN-DE-ESTADO (BC, t)
    Si p = NoHayPlan entonces
        G ← PREGUNTAR (BC, HACER-CONSULTA-META(t))
        p ← PLANIFICADOR-IDEAL (actual, G, BC)
    si p = NoHayPlan o si p está vacío entonces acción ← Noop
    entonces
        acción ← PRIMERO(p)
        p ← RESTO (p)
    DECIR(BC, HACER-ORACIÓN-ACCIÓN) (acción, t)
    t ← t + 1
    responder con acción
```

Figura 2.12: Un agente planificador sencillo. Este agente empieza por proponer la meta por alcanzar y luego procede a construir un plan que le permita obtenerla a partir de su estado actual. Una vez que cuenta con un plan, lo ejecuta hasta terminar con lo previsto por dicho plan y procede entonces a ocuparse de una nueva meta.

2.11 Tipos de Planificadores

En la literatura existe un gran número de planificadores de los cuales hemos seleccionado solo tres para la realización de nuestra investigación. Estos planificadores fueron escogidos en base a la características de nuestro problema, ellos deben soportar *Durative-Actions, Numeric Fluents, Strips, y Equality.* estás características necesarias para la solución de nuestro problema se describen en las líneas siguientes, lo cuales son llamados requerimientos. Estos planificadores son:

- LPG-TD [31].
- SGPLAN [40].
- PLAN [53].

Dentro de los requerimientos adicionales que nuestros modelos necesitan se encuentran los siguientes: a) *durative-actions*, este requerimiento permite la modelación de acciones con duraciones discretas [22], b) *numeric-fluents* que permite la manipulación de valores numéricos en PDDL-2.1 [24], [22]. Estas se definen en el dominio del modelo de planificación y se pueden utilizar como condiciones y efectos, los valores correspondientes a una *function* se asignan en el archivo problema.

En el caso de *durative-action* se requiere para modelar los tiempos de desplazamiento de usuarios y autobuses, y para calcular los *delays* o tiempos de retraso en la sincronización de los mismos, es decir cuánto cuesta llevar a cabo esa acción. Y los *fluents* a aparte de asignar valores a los posibles movimientos, nos sirven para controlar las diversas preferencias que fungen de restricciones en el modelo de planificación.

También hacemos uso de STRIPS que es el subconjunto más básico de PDDL, que consiste en "supuestos" [43]. Permite realizar búsquedas en el espacio de planes, agregando acciones al plan cuando son necesarias, es decir cuando se cumplen las condiciones necesarias y solo especifican un orden parcial.

Otro requerimiento necesario es el *equality* que nos permite utilizar el símbolo de igualdad "=" [43], del cual hacemos uso en el archivo problema para asignar pesos en las *function*.

Además de soportar estos requerimientos tan necesarios en el desarolló de nuestro modelo, estos planificadores fueron los ganadores de las últimas competencias internacionales de planificación, y el propósito de escogerlos radica también en que se quiere descubrir cuáles son las limitantes, o factores críticos que impiden que funcionen correctamente en nuestros problemas.

Tratamos de descubrir que tan bien escalan los planificadores en las instancias de nuestros problemas. Si estos no pueden resolver instancias sencillas, entonces los demás algoritmos de planificación no tendrían o tendrían muy poca oportunidad de resolver nuestros problemas. Utilizamos planificadores de uso general debido a la inexistencia de alguno apropiado a nuestro modelo de planificación.

2.11.1 LPG-TD

LPG-TD [31] es un poderoso planificador que tomó parte del 4th International Planning Competition, 2004. LPG-TD es una extensión del planificador LPG para manejar las nuevas características de PDDL-2.1 [22]: *"Timed initial literals"* y *"Derived predicates"*. La generación del plan se basa en una búsqueda local estocástica [27], el principal esquema de búsqueda de este planificador es el Walkplan [29].

La representación de planes temporales [55] con acciones durativas son basadas en grafos de acción. Dado un grafo de planificación ϑ para un problema de planificación es posible asumir que el nodo objetivo de ϑ en el último nivel representa la precondición de una acción especial a_{end}, el cual es la última acción de cualquier plan válido, mientras que los nodos de inicio en el primer nivel representan los efectos de una acción a_{start}, el cual es la primera acción de un plan valido. Para poder realizar un plan, el planificador debe de soportar acciones durativas para el cual es necesario el desarrolló a un grafo de acción durativa temporal *(TDA-graph)*.

Un grafo de acción durativa temporal [55] de ϑ es un triple$< A, \tau, \Omega >$ donde A es un grafo de acción durativa con propagación; τ es una acción de valor real para el factor, no-op (permiten el paso de una proposición que no se cumplió en el nivel i al nivel *i+1*) y nodos de acción de A; Ω es un conjunto de restricciones ordenadas entre los nodos de acción de A.

Ordenamiento de restricciones [55]

El ordenamiento de restricciones en TA-Graphs [28] es de dos tipos: restricciones entre acciones que están implícitamente ordenados por la estructura causal del plan ($\prec_{C-constraints}$), y restricciones impuestas por el planificador para tratar con acciones mutuamente exclusivas ($\prec_{E-constraints}$). $a \prec_c$ pertenece a Ω si y solo si a es usada para alcanzar un nodo de condición de b en A, mientras $a \prec_E$ b pertenece a Ω solo si a y b son mutuamente exclusivos en A. Si a y b son exclusivos, el planificador impone tanto $a \prec_E b$ o $b \prec_E a$. LPG elige $a \prec_E b$ si el nivel de a precede el nivel de b, $b \prec_E a$ otro caso.

En dominios implementados en PDDL-2.1 con acciones durativas, LPG usa TDA-graphs, si $\Omega \models a \prec b$, es posible que a y d se superponen; de esta manera, nosotros podemos distinguir otros tipos de restricciones en conformidad con las posibles relaciones causales ($\prec_C$) y con las mutex entre condiciones y efectos ($\prec_E$). Dos acciones ordenadas a y b pueden sobreponerse en 4 modos diferentes. Algunos modos representan un orden diferente de restricción; por tanto, se tiene los siguiente [55]:

a. b no puede iniciar antes de que finalice a.

b. b no puede finalizar antes que finalice a.

c. b no puede iniciar antes de que comience a.

d. b no puede finalizar antes de que comience a.

Ellos denotan con *Time(x)* el valor temporal asignado por τ para un nodo x.

En el TA-graphs, LPG calcula el valor temporal de una acción b para simplificar se examina el valor máximo sobre el valor temporal de una acción a en A que precede a b conforme a Ω:

$$Time(x) \max_{a \prec b \in \Omega} \{Time(a), 0\} + dur(b) + \varepsilon$$

Los valores temporales de los nodos de acción TA-Graph son usados para calcular el valor temporal de TDA-graph de nodos hechos y no-op. Si un nodo hecho f es soportado por varias acciones, LPG considera los valores temporales de la acción que soporta f tempranamente.

En TDA-graph, LPG calcula los valores temporales de un nodo de acción hecho f para simplificar la examinación del valor mínimo sobre los valores temporales de la acción a en A que soporta f:

$$Time(f) = \min_{a \in \Lambda(f)} \{Time(a)\},$$

donde $\Lambda(f)$ es el conjunto de nodos de acción TA-graph que soporta el nodo hecho f. En TDA-graph, LPG distingue los casos en los cuales f es soportado hasta el comienzo o hasta el fin de una acción, y por tanto el valor temporal de un nodo hecho f es calculado de acuerdo a la siguiente definición de *Time(f)*:

$$Time(f) = \min \left\{ \min_{a \in \Lambda(f)} \{Time(a)\}, \min_{a \in \Lambda_S(f)} \{Time(a) - dur(a)\} \right\}$$

donde $\Lambda_E(f)$ y $\Lambda_S(f)$ son los conjuntos de los nodos de acción en TDA-graph que soportan f al final y al comienzo, respectivamente. En problemas de planifiación para los cuales es importante minimizar el makespan del plan, LPG usa estos valores temporales para guiar la búsqueda hacia una dirección que mejora la calidad del plan.

LPG-TD tiene la particularidad de generar planes optimizando la longitud del plan, es decir el número de acciones necesarias para poder llevar a cabo el plan arrojado, makespan, y el costo de ejecución, asegurando de este modo planes de muy buena calidad pero no así el resultado óptimo [29] [30], y en el caso del tiempo de ejecución, se reporta que este planificador es bastante eficiente en un gran número de problemas presentados en el ICAPS 2002 [2] y en el del 2004 [3] manteniendo tiempos

bajos al generar una primera solución, la cuál es la mejor en base a la longitud del plan [30].

Por lo detallado anteriormente podemos llegar a creer que los planes obtenidos a partir de este planificador LPG-TD deben de ser de muy buena calidad y su longitud o número de acciones del plan deberán de ser lo menor posible, debido a su capacidad de crear planes combinando la calidad y longitud de la solución, lo cual es ideal para el usuario.

2.11.2 SGPLAN

El segundo planificador que consideraremos en nuestro estudio es SGPLAN. Este planificador tiene la particularidad de particionar un problema de planificación en sub-problemas, ordena cada uno de estos de acuerdo a una resolución secuencial de cada uno de estos sub-objetivos, y a partir de esto encuentra un plan factible para cada factor objetivo [10] [11] [12].

El propósito de realizar estas particiones es que conducen a subproblemas significativamente más fáciles de resolver y que a la vez son similares al problema original, pudiendo ser eficientemente solucionado por el mismo planificador con algunas modificaciones en su función objetivo [13].

Las condiciones pueden ser descompuestas dentro de una forma particionada donde cada problema es asociado con una condición local necesaria. Las condiciones particionadas reducen grandemente el espacio de búsqueda, lo cual lleva a obtener planes en un tiempo de ejecución muy bajo, esto se verá en la sección de experimentos y resultados. Las condiciones están definidas en una función Lagrangiana que consiste de la suma de los objetivos y el peso de las restricciones por un multiplicador de Lagrange.

Arquitectura del planificador[11]:

En la figura 2.13, se muestra la arquitectura de SGPLAN, en el nivel global

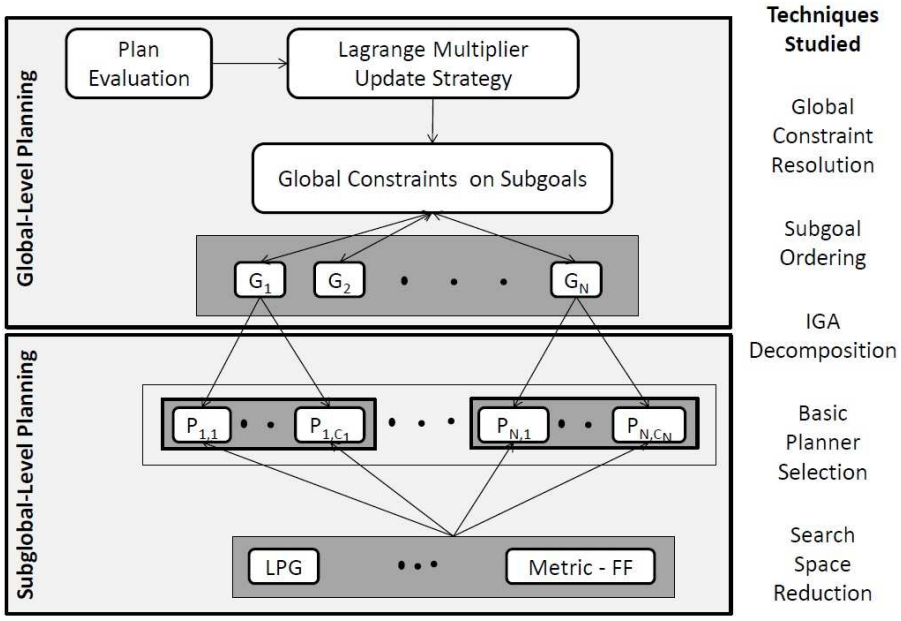


Figura 2.13: Arquitectura del planificador SGPLAN[11].

se selecciona un orden adecuado para el planificador particionando al problema en sub-objetivos, esto introduciendo restricciones globales artificiales con la finalidad de hacer cumplir que la solución de un sub-objetivo no invalide a uno anterior, por otro lado resuelve las restricciones violadas.

En el nivel local se realiza una descomposición jerárquica para pocas acciones irrelevantes antes de llamar o elegir un planificador. En el segundo nivel se lleva a cabo un análisis de reducción del espacio de búsqueda [11].

Planificando a nivel global [11]:

Las heurísticas para el ordenamiento parcial de sub-objetivos son las siguientes:

Heurísticas para el ordenamiento parcial de sub-objetivos:

a) Ordenamiento razonable: En esta heurística el factor objetivo A esta ordenado antes de B en la lista de subojetivos, pero antes de esto se debió de obtener un plan que alcance a A, no se puede obtener B sin invalidar primero a A. Entonces la búsqueda para alcanzar primero A se desperdicia y es más eficiente

para alcanzar a B antes de A.

b) Ordenamiento de irrelevancias: Se basa en el cálculo del número de acciones irrelevantes de cada factor objetivo, ordena A antes de B si A tiene menos acciones irrelevantes.

c) Ordenamiento de precondiciones: Para A y B con el mismo número de acciones irrelevantes que no pueden se ordenadas por un ordenamiento razonable, se ordena A antes de B si $\eta_p(A)<\eta_p(B)$. Donde $\eta_p(A)$ es el minimo número de precondiciones de las acciones de apoyo.

$$\eta_p(A) = min\ \eta_{pre}(A),\ a \ni S(a)$$

donde $S(A)$ es el conjunto de todas las acciones que soporta el factor objetivo A, y η_{pre} es el número de precondiciones de la acción a.

Cada restricción global en SGPLAN es una restricción binaria que indica si existe o no conflictos al solucionar subobjetivos.

Para lo anterior se hace uso de la resolución de restricciones globales que definen a las funciones Lagrangianas que consisten en la suma de los objetivos y peso de las restricciones por multiplicadores de Lagrange. Donde se realiza periódicos decrementos de los multiplicadores, así como ascensos.

Seguido de esto se utiliza una planificación de nivel sub-objetivos [11] donde se descompone el problema y se crean a partir de una agenda intermedia de subobjetivos. Permitiendo con ello la reducción del espacio de búsqueda [11] mediante la eliminación de acciones irrelevantes.

Este planificador como se mencionó anteriormente tiene la particularidad de generar planes en tiempos muy cortos, además de esto la longitud de los planes arrojados se conforman de pocas acciones, es decir la longitud de la solución es corta, esto se verá más adelante.

Como SGPLAN usa subgoal partitioning podría particionar nuestro problema original, en dos subproblemas, el subproblema que mueve caminando al usuario hacia la parada más prometedora según la métrica, y una segunda partición que se encarga de mover el autobús hasta la parada del usuario, esperando con esto reducir el tiempo de ejecución.

2.11.3 PLAN-TFD (Temporal Fast Downward)

Este planificador es presentado por primera vez en el ICAPS-2008, ganando uno de los primeros lugares debido a su desempeño al momento de resolver diversos problemas de planificación.

Este planificador trabaja en tres etapas:

- Translation.
- Knowledge compilation.
- Search

La etapa de traducción *translation* consiste en la compilación de características ADL (Action description language [49][50]) y objetos fluents, permite el movimiento de las características numéricas de los operadores a los axiomas. También posibilita el aterrizamiento de los operadores y axiomas, y la conversión de los resultados representados para elegir variables de estado con multi-valor.

Mientras que en la etapa *Knowledge compilation* tiene como propósito construir estructuras de datos que utilicen la heurística aditiva y que faciliten la eficiencia de la expansión de estados durante la búsqueda. Las responsabilidades de este paso de pre-procesamiento son el cálculo de los grafos causales de la tarea de planificación que codifican dependencias entre diferentes variables de estado, calcular el grafo de transición de dominio (DTG) para cada variable de estado que codifican como los operadores afectan la variable y así como también el cálculo del generador

sucesor, una estructura de datos que soporta eficientemente el cálculo en una sobre aproximación del conjunto de operadores aplicables para un estado.

El grafo causal codifica dependencias entre diferentes variables de estado. Para cada variable de estado un grafo de transición de dominio es calculado para codificar bajo cuales circunstancias la variable puede cambiar su valor. Este planificador debe de distinguir diferentes formas de DTGs para las diferentes tipos de variables en una tarea temporal multi-valor.

En *Search* el planificador utiliza una búsqueda greedy best-first con enfoque mejorado con evaluaciones heurísticas diferidas.

1. Espacio de búsqueda: Además de los valores de las variables de estado, los estados con marca de tiempo en el espacio de búsqueda contienen una marca de tiempo real así como información acerca de los efectos programados y las condiciones de las acciones ejecutadas actualmente.

2. Heurística: La búsqueda es guiada por una variante de context-enhanced heurística adictiva adaptada para manejar variables numéricas. La heurística es una generalización de la heurística adictiva y del grafo causal usado en Fast Downward [36].

   Esta heurística usa técnicas de búsqueda basadas en el algoritmo de Dijkstra y toma en cuenta el contexto actual de las variables de estado relevantes. La idea básica de la extensión de las variables de estado numéricas es para construir el grafo en cual el Algoritmo de Dijkstra es realizado sobre la marcha durante el cálculo heurístico.

Capítulo 3

Formulación del problema

En el presente capítulo se expondrá una descripción del problema que estamos atacando, los supuestos involucrados, así como todos los elementos y métricas necesarias para el desarrollo del dominio de planificación y del archivo problema. También se abordará en uno de estos puntos las preferencias de los usuarios, y como representaremos estas en el modelo.

Los generadores de planes de viaje en su inmensa mayoría arrojan una ruta de viaje que es válida para cuando el usuario tiene su propia forma de trasladarse, pero la mayoría de los habitantes de una ciudad se desplaza utilizando el transporte público. Sumando a esto, las distancias a recorrer son muy largas en la mayoría de los casos, he aquí donde se decide generar rutas de viaje utilizando el transporte público. Nosotros estamos limitando el problema a rutas de autobuses, aunque bien podría funcionar para otros tipos de medios o combinaciones entre ellos.

De los trabajos encontrados en la literatura se tiene que la mayor parte de estos generadores de rutas trabajan en base a algoritmos tradicionales para encontrar el camino más corto, como lo son el algoritmo de Dijkstra o A*, con modificaciones para reducir el espacio de búsqueda y por lo tanto el tiempo de ejecución. Por otro lado, solo se encontró uno donde lo modelan con planificación solo que al final terminaron desechando esa idea debido a que solo fueron capaces de soportar muy pocos elementos correspondientes al sistema de transporte.

El punto es que no se ha atacado este problema mediante el uso de modelos de

planificación de forma seria y es lo que nosotros pretendemos lograr, considerando las diversas acciones necesarias en el sistema, así como las variables que podemos controlar y representar en este modelo.

Sin embargo, no solo tratamos de hacer planes considerando únicamente la topología de la red, si no también usando aquellas preferencias que el usuario tiene al momento de realizar su plan. Al considerar preferencias de los usuarios nuestros modelos son mucho más expresivos que las soluciones actuales, pero tendiendo también a aumentar su complejidad de representación y resolución. Por el contrario, los trabajos relacionados existentes no consideran restricciones al momento de calcular el plan, y solo se basan en un solo objetivo a optimizar, ya sea tiempo o distancia total del recorrido.

Para poder realizar esto, necesitamos conocer que elementos conforman la red de transporte público, y hacer un análisis de las preferencias de los usuarios por su grado de importancia, y una vez seleccionadas estás, se debe proceder a identificar cuáles se pueden representar en nuestro modelo, y cómo agregarlas en el dominio de planificación. Seguido a esto se especifica a que acción pertenece cada una de estas restricciones, o elementos de la topología del sistema, esto es crucial para la realización del plan ya qué determina que pasos realizar en función de lo anterior.

Como bien se mencionó, la importancia de realizar este trabajo radica en que no se han encontrado investigaciones que muestren resultados satisfactorios utilizando técnicas de planificación, las cuales en teoría deberían de ser mejores para la generación de dichos planes de viaje, permitiendo modelar de una forma más simple la realidad del sistema, subdividida en todas esas acciones necesarias para poder realizar dicho plan, considerando las preferencias de los usuarios.

Trabajaremos con la generación de planes donde se considere los tiempos de espera por parte de los usuarios, esto tiene gran peso en la hora de resolver el problema, ya que el tiempo de espera puede ser muy alto, aunque el de recorrido sea pequeño, entonces el viaje tiene un tiempo de ejecución alto, por lo tanto se

presentará un modelo de dominio de planificación donde se tome en cuenta el tiempo de espera por parte del usuario.

Otro reto de nuestro proyecto de investigación no solo se centra en la generación del modelo de planificación, si no también en descubrir, mediante la experimentación, qué factores afectan directamente la capacidad de resolución de los planificadores, ya sea en uso de memoria o en tiempos de ejecución.

Se busca también estudiar la posibilidad de crear sub-modelos para los diferentes tipos de métricas, con la finalidad de observar los cambios en los tiempos de ejecución conforme este número de métricas se incrementan, las cuales son de interés para el usuario que realiza la consulta al momento de computar un plan de viaje.

Siguiendo con los factores, sean el número y tamaños de las rutas, cantidad de unidades de transporte, número de nodos en el grafo, se analizarán en su grado de afectación a la hora de ejecutar el planificador, esto para dar paso a investigaciones futuras a trabajar directamente sobre los algoritmos de los planificadores, mejorándolos o apegandolos más a nuestro dominio de planificación, de modo que soporten todo lo implementado en el modelo de transporte público, mejorando así su rendimiento, tanto en memoria, tiempo de ejecución y calidad del plan.

En resumen el problema abordado consiste en: estudiar las preferencias de los usuarios al momento de realizar un viaje en una unidad de transporte público y seleccionar las más importantes, seguido de esto representar tanto la topología de la red de calles y de rutas de transporte junto con las preferencias de los usuarios en un modelo de planificación, todo esto para generar planes de viaje para el usuario mediante el uso de estas técnicas de inteligencia artificial, ofreciendo un mayor número de restricciones y opciones a optimizar, teniendo así un plan generado más completo que los que se pueden obtener en la actualidad. Por otro lado se buscará mediante un diseño de experimentos las limitantes de los planificadores utilizados en la solución de nuestro dominio, esto para hacerles mejoras, e incrementar su rendimiento, tanto en la calidad del plan, como en el tiempo de ejecución.

3.1 Planteamiento del problema

El problema que estamos tratando, como bien se ha recalcado a lo largo de esta tesis, es la generación de planes de viaje para el usuario usando para ello la red de transporte público, utilizando técnicas de planificación. A este modelo se le agregan las preferencias del usuario ya sea como función a optimizar *f(s,t)* para el plan de viaje, o como restricciones *PR*.

Anteriormente se dijo que este problema requiere del uso de los grafos *G(V,E)* y *W(V,E)* los cuales respectivamente pertenecen a la red de transporte, donde se definen las rutas de autobuses R y a la red de caminos por donde el usuario p puede desplazarse libremente. Tenemos que los arcos a y b (estos arcos representán las rutas de los grafos correspondientes a las rutas de autobuses y a las rutas caminando respectivamente) son bidireccionales con valores positivos para cada uno de los grafos utilizados, respectivamente.

Hasta este punto solo se tienen dos funciones objetivos *f(s,t)* (donde s es el punto de origen y t el objetivo) a optimizar las cuales corresponden al tiempo de duración de un plan *T(s,t)* y al costo económico del mismo para p, *C(s,t)*, las demás *f(s,t)* serán tomadas de acuerdo a las preferencias a optimizar por parte de p, estas preferencias fueron tomadas de algunas consultas realizadas y se mostrarán mas adelante en este capítulo.

Para cuando *f(s,t)* sea *T(s,t)* se optimizará la suma del tiempo recorrido en una unidad de transporte *TR*, siendo este tiempo *Tb*, más el tiempo en el cual p se encuentra caminando y se representa como *Tc*. Este último tiempo se obtiene al dividir $\ell(b)$ (distancia recorrida caminando) entre una constante, esta constante tiene el valor de 1.39 el cual es la velocidad en m/s que una persona alcanza caminando en promedio.

Sin embargo cuando p realiza un plan en base a lo anterior tiene como preferencia más qué simples *f(s,t)* a optimizar, entonces p puede ingresar ciertas restricciones

las cuales limita el plan a generar de acuerdo a lo que él desea. Por ejemplo, el usuario desea caminar una distancia máxima durante la duración del plan *Dmax(s,t)*, entonces el ingresa esta distancia y nuestro modelo de planificación entregará un plan a p, si es qué este existe. Teniendo así que estas restricciones son válidas para las diversas *f(s,t)* a optimizar.

Por lo tanto nuestro modelo debe ser capaz de representar los elementos de *G(V,E)* y *W(V,E)* así como las *f(s,t)* deseadas por el usuario al momento de planear su viaje, para esto se hará uso de las diversas preferencias que restringen al plan.

3.2 Supuestos del problema

En el diseño de nuestro dominio de planificación estamos suponiendo un estado ideal, donde no hay cambios en el medio, por lo tanto los tiempos de recorrido no varían, al igual que las rutas de transporte. Cada ruta se agrega una sola unidad por dirección y se encuentran ubicadas en extremos contrarios, suponemos que sus ubicaciones no generarán perturbaciones al momento de computar un plan, debido a que el peso por mover unidades es igual a cero cuando no llevan ni un pasajero (no existe tiempos de espera). Todas las rutas de transporte público tienen la misma cantidad de paradas para un problema dado.

Se considera que todas las unidades de transporte tienen espacio disponible, no existen horarios que restrinjan la frecuencia de paso, aunque se piensa en algún futuro implementar esto último.

Otra de las suposiciones que hacemos es que cada nodo representa la esquina de una cuadra, para las rutas de autobuses tenemos que cada esquina correspondiente a su recorrido se toma como una parada en nuestro modelo de planificación.

Para nuestros fines solo se genera un plan, ya que solo se trabaja para un solo usuario, sin embargo, el mismo problema se pudiera agrandar incluyendo más de un estado inicial y objetivos. Esto hace que el problema sea interesante, ya que esto no

lo puede llevar a cabo los algoritmos existentes basados en técnicas clásicas.

3.3 Preferencias del usuario a considerar

Anteriormente se mencionó que uno de los objetivos primordiales de este trabajo de investigación es la implementación de un sistema que genere planes de viaje de acuerdo a las necesidades del usuario, y que a la vez sea fácil de consultar. En el presente punto mostraremos un conjunto de preferencias que previamente fueron seleccionadas de [46] y de una encuesta realizada por una empresa de transporte público de la Ciudad de Monterrey, N.L., en la tabla 3.1 se muestran estas preferencias divididas en tres categorías: tiempo, comodidad y capacidad.

Tiempo
Frecuencia de paso
Tiempo de recorrido
Información de horarios
Tiempo de duración del viaje (incluye tiempo de espera)
Tiempo de recorrido (tiempo caminando + tiempo en autobús)
Comodidad
Asientos libres
Número de transferencias
No aglomeraciones
Viaje placentero
Limpieza de las unidades
Capacidad del usuario
Costo económico
Longitud a recorrer caminando

Tabla 3.1: Preferencias de los usuarios al momento de querer hacer uso del transporte público.

Como se puede ver en la tabla 3.1 algunas de estas preferencias no se pueden controlar numéricamente, siendo así eliminadas junto con algunas otras más. Por

Figura 3.1: Preferencias de los usuarios al utilizar el transporte público.

otro lado se ha decidido agregar algunas preferencias en base a la experiencia del uso del transporte público y de los comentarios recolectados en la experiencia diaria entre los usuarios, la figura 3.1 muestra las preferencias primarias, de allí se desprenden algunas secundarias, todas estas se explican más adelante.

Menor costo.

Debido a las tendencias económicas actuales, salarios bajos e incremento de los costos en todos los rubros, los usuarios del sistema de transporte público siempre buscarán o querrán optimizar su recurso económico percibiendo un mayor beneficio como consecuencia, y esto también aplica cuando se requiere hacer uso de éste servicio, ya que desean que este sea lo más económico posible, lo cual es una función a minimizar al generar un plan de viaje para el usuario.

Por otro lado, ¿Qué pasa si el usuario tiene una cantidad destinada o límite para gastar en transporte?, he aquí una restricción de preferencia por el usuario que debe de ser tomada en cuenta, debido a su importancia. Esta restricción es tomada directamente del usuario al realizar su consulta. Por lo tanto desde la parte económica, se tiene una función objetivo a minimizar y una restricción de lo disponible a gastar durante su traslado.

Minimización de transferencias.

El número de transferencias no solo influye en el costo económico, sino también en los tiempos de recorrido ya que cada transferencia conlleva un tiempo de

espera. Además, la importancia de minimizar el número de transferencias radica en la incomodidad para los usuarios de realizar transferencias y más que nada porque muchos de estos usuarios pueden no estar en condiciones para llevarlas a cabo (mujeres embarazadas, usuarios con niños, personas de la tercera edad, o que padezcan de una discapacidad).

Ruta más corta o menor tiempo de viaje.

Para este caso, todos los trabajos encontrados de sistemas de rutas de viaje manejan la minimización tanto del tiempo o de la distancia recorrida, teniéndose buenos resultados en las aplicaciones existentes online (maps, traffic, yahoo, waza, gps, bing, étc.), pero estos sistemas no toman en cuenta la ubicación de las unidades de transporte público, ya que son diseñados para problemas de rutas generales, en el caso de Google Traffic solo se indica las rutas que siguen los autobuses pero desconoce las ubicaciones de las unidades los cuales repercuten en el tiempo de espera de los usuarios el cual es parte del tiempo de viaje.

Existen sistemas como ruta directa [20] donde solo se marca la ruta a seguir por las unidades al momento de consultar un recorrido, lo cual no permite optimizar alguna métrica, es más ni siquiera computa un plan tal cual.

La importancia de considerar la ubicación de los autobuses consiste en minimizar el tiempo de espera por parte del usuario en la parada de autobuses, ya que de este modo se puede sincronizar con el usuario una unidad que en conjunto minimicen el tiempo de recorrido y el tiempo de espera. De este modo se asigna una unidad de transporte al usuario, y él se quita la preocupación de estar pensando que unidad tomar y si se va tardar mucho en pasar.

Longitud caminada.

La finalidad es minimizar tiempos de recorrido, lo cual abarca el tiempo que el usuario caminará dentro del plan de viaje generado. Pero este tiempo y/o distancia de recorrido debe ser la menor posible por comodidad del usuario, es de aquí que nace la necesidad de considerar cuanto es lo máximo que el usuario desea caminar hasta

alcanzar la parada, y por otro lado también se tiene, cual es la distancia máxima que está dispuesto a caminar durante todo el traslado, así cómo una métrica a optimizar.

Combinación de métricas. Costo-Tiempo de recorrido.

Como bien se sabe, estás son las métricas a minimizar más importantes y en conjunto tienen una gran relación, es por eso que se decidió manejarlo como una preferencia a optimizar. Donde el usuario ingresa pesos ponderados tanto para el tiempo como para el costo económico del plan, pudiendo así generar un plan de viaje donde se optimice ambas métricas.

En la figura 3.2 se muestran las preferencias que se modelaron en nuestro dominio de planificación, en el segundo nivel se presentan los tipos de medida a optimizar, en el tercer nivel se observan en color gris las funciones objetivos, mientras que las que están en color blanco son las preferencias del usuario (restricciones para el modelo). Para el caso de la preferencia máxima distancia a caminar se subdivide en dos partes, la primera para alcanzar la parada y la segunda para el total del recorrido.

Comodidad.

Los usuarios siempre buscarán estar cómodos por diversas razones. Como lo pueden ser: embarazos, discapacidad, tercera edad, demasiada carga, viajes largos, cansancio, entre otros, es por eso la selección de esta preferencia donde se pretende asegurarle al usuario un asiento disponible, o que al menos la unidad de transporte no esté muy saturada, para esto se requieren de pronósticos. Para el caso de nuestro trabajo esta preferencia no es tomada en cuenta, debido a la falta de modelos de pronósticos. Sería implementada más adelante.

Combinación de tipos de transporte.

Más que una preferencia es una opción que se tiene debido a la diversidad de tipos de transporte existentes en las grandes ciudades (metro, tren ligero, autobuses, microbuses, taxis, combis, étc.). Esto podría modelarse sin ningún problema al in-

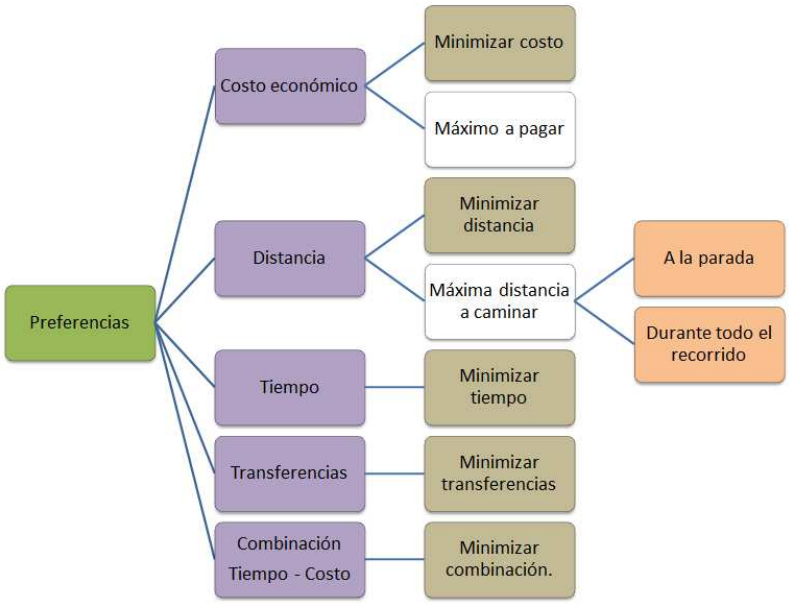


Figura 3.2: Preferencias de los usuarios tomadas en cuenta en el dominio de planificación.

cluir predicados adicionales así como acciones para los otros tipos de transporte, sin embargo, este punto no es considerado en nuestra experimentación. Pero para nuestros propósitos otro tipo de transporte es muy sencillo de incorporar y no aumenta la complejidad del problema ya que se considera como otra ruta más. Para su validación incluímos una versión modificada de nuestro modelo original que considera el sistema del metro el cual, tales modificaciones se muestran en las figuras A.1 y 3.17. Todas las preferencias mostradas son válidas para cualquier función objetivo y combinándose entre ellas.

3.4 Modelación del dominio

En este subcapítulo se mostrarán todos los elementos necesarios para la formulación del dominio de planificación, y las variables a utilizar, explicando él por qué son necesarias, así como las acciones requeridas y su formulación en base a las condiciones y efectos que ellas involucran. Una vez presentado lo anterior mostramos nuestro dominio formulado para el transporte público.

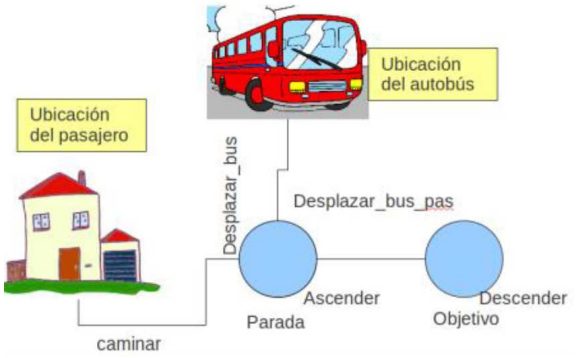


Figura 3.3: Acciones necesarias en el dominio de planificación del transporte público.

3.4.1 Acciones a utilizar

Los dominios de planificación se constituyen por la descripción de acciones que modifican el entorno del problema para generar un plan. Para nuestro enfoque, hemos considerado cinco acciones, las cuales se enlistan a continuación y se representan gráficamente en la figura 3.3.

- Caminar.
- Ascender_pasajero.
- Descender_pasajero.
- Desplazar_bus.
- Desplazar_bus_pas.

Descripción de las acciones.

Las acciones anteriormente mencionadas aparte de indicarnos los pasos a realizar para poder generar un plan de viaje, también nos permiten controlar las diversas métricas necesarias a optimizar.

Acción Caminar

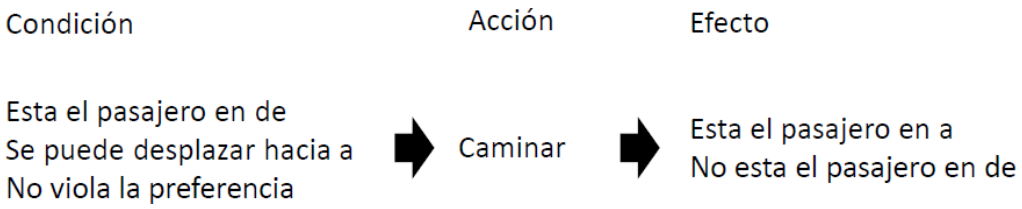


Figura 3.4: Diagrama de estados correspondiente a la acción caminar.

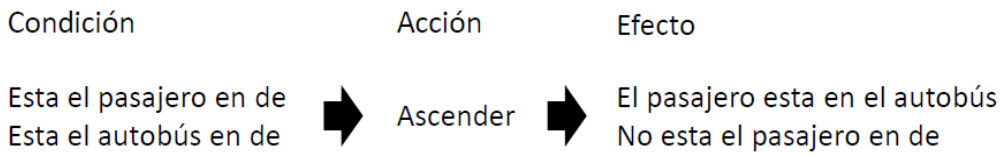


Figura 3.5: Diagrama de estados correspondiente a la acción ascender.

Permite desplazar al usuario por sus propios medios a través de la red de caminos existentes (ver figura 3.3), esto para alcanzar las paradas de autobuses o los objetivos. Esta acción nos permite llevar un control de las distancias y tiempos de recorrido para su optimización. Para más detalle ver la figura 3.4, donde se muestra esta acción en un diagrama de estados donde se observa las condiciones necesarias y los efectos ocasionados al momento de ejecutarse para su posterior modelación.

Acción Ascender

Esta acción permite ascender al pasajero a una unidad de transporte. La única condición necesaria es que tanto el pasajero como la unidad estén en la misma ubicación. La explicación anterior es para la forma generalizada de esta acción, pero para acercar esto más a la realidad y conociendo la ubicación del autobús, se puede restringir que solo se podrá ascender a la unidad si su tiempo de llegada a la parada sea mayor que la del pasajero a ese mismo punto, lo mismo sucede si el pasajero se ubica en un autobús y requiere hacer una transferencia.

Esta acción es de suma importancia ya que nos permite controlar el costo económico del plan, el número de transferencias, combinación tiempo-costo. Y hablando del costo económico, en esta acción *Ascender* estará ubicada la restricción del límite del costo del plan que el usuario desea pagar. Para un mayor entendimiento

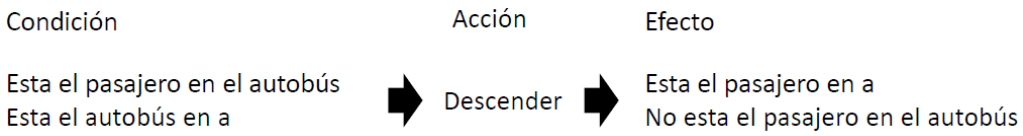


Figura 3.6: Diagrama de estados correspondiente a la acción descender.

ver la figura 3.5, donde se muestra esto en un diagrama de estados donde claramente se pueden observar las condiciones y efectos mencionados para esta acción.

Acción Descender

La realización de ésta acción, tal como se puede ver en la figura 3.6 se requiere tener conocimiento de la ubicación de la unidad de transporte y que el pasajero se encuentre dentro de ella, si esto se cumple tenemos como efecto que el usuario ya no se encuentra en la unidad pero si en la ubicación actual de esta última.

Acción Desplazar_bus:

Nace con la necesidad de simular el movimiento de unidades de transporte hacia las paradas (ver figura 3.3). En un modelo general no genera cambios en las diversas métricas a considerar, sin embargo, si se quiere minimizar tiempos en base a los tiempos de espera, es necesario utilizar una métrica que almacene el tiempo de recorrido de la unidad hasta la parada de autobuses, esto para sincronizar con el tiempo de recorrido caminando por parte del usuario y reducir el desfase entre estos, para ello se requiere de una restricción que indique que el tiempo de recorrido del autobús a la parada debe ser mayor al del usuario hacia ese mismo tiempo.

Para un mayor entendimiento de sus condiciones y efectos, ver la figura 3.7, donde se requiere como condiciones conocer la ubicación actual de la unidad y a donde se puede desplazar. Los efectos correspondientes es que la unidad se encuentra en otro punto de los posibles a alcanzar, negando su ubicación anterior.

Acción Desplazar_bus_pas

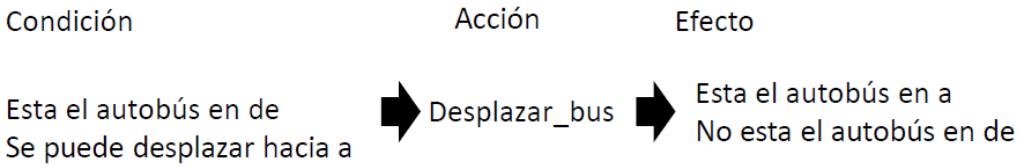


Figura 3.7: Diagrama de estados correspondiente a la acción desplazar_bus.

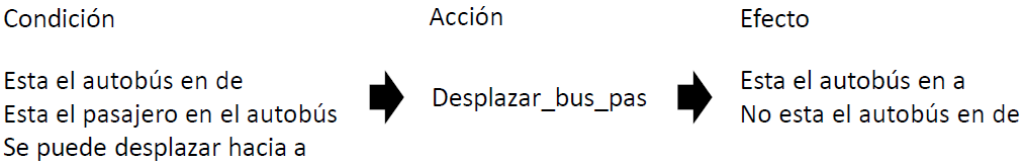


Figura 3.8: Diagrama de estados correspondiente a la acción desplazar_bus_pas.

Se ejecuta cuando el usuario ha ascendido a la unidad de transporte desplazándolo en su interior hacia el objetivo establecido por parte del pasajero, ver figura 3.3. Esta acción no necesariamente tiene que llevar al usuario exactamente al objetivo, si no también a un punto cercano donde pueda alcanzar caminando su meta o alguna parada de autobuses sin violar las restricciones de longitud, si es que fueron dadas por el usuario. Ésta acción permite la minimización del tiempo de recorrido.

Las condiciones y efectos se representan en la figura 3.8 donde se requiere conocer la ubicación de la unidad hacia donde se puede desplazar y que el usuario haya abordado esa unidad, teniendo como efecto el desplazamiento a un nodo que se pueda alcanzar desde su ubicación actual.

Sin embargo mediante la experimentación se descubrió que en muchas ocasiones el plan arrojado indica que se debe de ascender un autobús y en un recorrido mínimo descender de él. Esto es válido considerando que se minimizaba tiempos, siendo esto más rápido que ir caminando. Sin embargo, económicamente no es factible, así que se puede colocar en esta acción una variable de mínima distancia recorrida en el autobús, que restringe la *Acción Descender*.

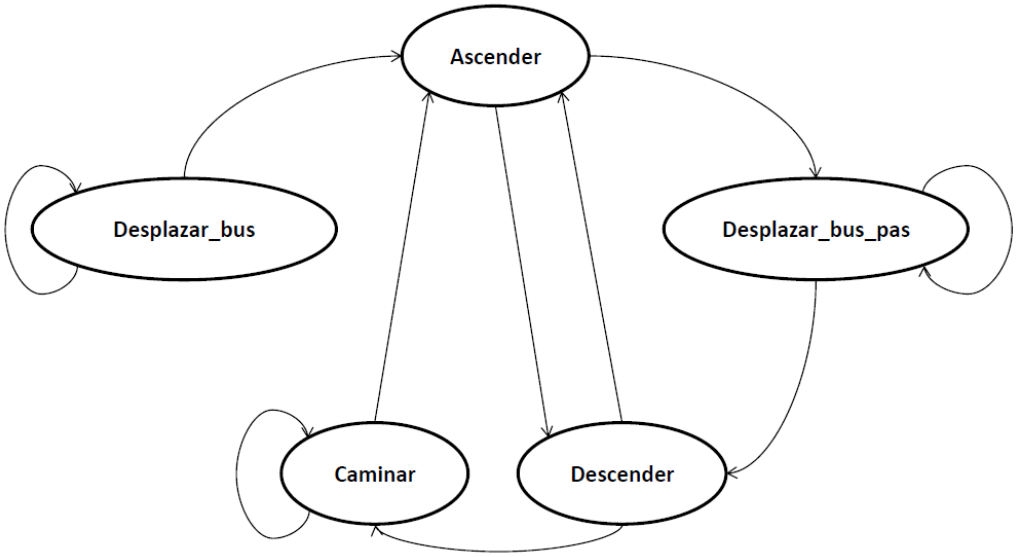


Figura 3.9: Diagrama de transición de las acciones correspondiente al modelo de planificación de transporte púlico.

Diagrama de transición de las acciones del modelo de planificación

En la figura 3.9 se puede ver las transiciones que se pueden realizar en nuestro modelo de planificación en base a las acciones implementadas. Como bien se muestra, tenemos que la acción *Desplazar_bus* solo puede transicionar a la acción *Ascender* y sobre ella misma ya que se puede desplazar la unidad por varios puntos antes de llegar al usuario.

A partir de la acción *Ascender* se puede pasar a las acciones *Descender* y *Desplazar_bus_pas* y esta acción se puede mantener hasta que se requiera dejar la unidad de transporte, haciendo uso para esto de la acción *Descender*.

De la acción *Descender* se puede alcanzar la acción *Ascender* esto es posible si en el punto donde el pasajero desciende puede abordar otra unidad de transporte. Desde esta misma acción se puede ir a la acción *Caminar*.

Siguiendo con el diagrama mostrado en la figura 3.9 se observa que a partir de la acción *Caminar* se puede transicionar a la acción *Ascender* o mantenerse en ella

hasta alcanzar el destino final o para abordar una unidad de transporte.

3.4.2 Requerimientos

Tenemos dos requerimientos necesarios para el soporte de nuestro modelo, el primero de ellos es *:numeric-fluents*, esto para el soporte de las function presentes.

Por otro lado necesitamos indicar que trabajaremos con *durative-action* debido a la naturaleza de nuestro dominio, para esto debemos indicar que necesitaremos del requerimiento *:durative-action*.

Los requerimientos antes mencionados son los necesarios para el correcto funcionamiento de nuestro dominio de planificación.

3.4.3 Requerimientos de las acciones

En este apartado se muestra en la tabla 3.2 las acciones con sus respectivos predicados y *function* para su correcto modelado. Cabe mencionar que estas acciones son del tipo durative-action, tal como se describirá más adelante, estás acciones conllevan un costo por su realización en este caso son los tiempos de recorrido.

El valor 1.39 en la duration *(/(longitud_cam ?pas ?de ?a)1.39)* para action-durative *Caminar*, es la velocidad promedio de una persona caminando, esto lo tomamos debido la imposibilidad de tomar tiempos de recorrido para una persona en específico y se decidió calcular este tiempo en base a la distancia entre dos nodos *(longitud_cam ?pas ?de ?a)*, de igual modo nos ayuda a reducir el tamaño del archivo problema al no tener function para los tiempos de desplazamiento caminando.

3.4.4 Predicados

En éste punto, formularemos y describiremos los predicados correspondientes a cada una de las acciones necesarias en nuestro modelo de planificación, teniendo

Durative-action	Parameters	Predicates	Function	Duration
Caminar	?pas ?de ?a	(pasajero ?pas) (ir_cam ?pas ?de ?a) (at ?pas ?de ?a)	(longitud_c) (long_max_cam) (longitud_cam ?pas ?de ?a) (dist_max_c) (dist_maxima) (longitud_caminada)	(/(longitud_cam ?pas ?de ?a)1.39)
Ascender	?pas ?bus ?de	(pasajero ?pas) (autobus ?bus) (at ?pas ?de) (rut_bus ?bus ?de)	(pasajero_in_bus) (longitud_c) (num_ascensos) (costo ?bus) (costo-total) (costo_max)	0
Desplazar_bus_pas	?bus ?de ?a	(autobus ?bus) (rut_bus ?bus ?de) (via_bus ?bus ?de)	(tiempo-total) (pasajero_in_bus)	(tiempo_b ?bus ?de ?a)
Desplazar_bus	?bus ?de ?a	(autobus ?bus) (rut_bus ?bus ?de) (via_bus ?bus ?de ?a)	(pasajero_in_bus)	0
Descender	?pas ?de ?a	(pasajero ?pas) (autobus ?bus) (in ?pas ?de ?a) (rut_bus ?bus ?a)		0

Tabla 3.2: Componentes de las acciones del modelo de planificación para el dominio de transporte público.

en cuenta que siempre deben ser positivos cuando se usan como condiciones dentro de la acción. En la tabla 3.2 se puede visualizar los predicados necesarios para cada acción a utilizar.

Los predicados *(pasajero ?pas)* y *(autobus ?bus)* indican que se requiere de un pasajero y de una unidad de transporte explicita para poder llevar a cabo todas las acciones. Para conocer la ubicación actual de la unidad de transporte utilizamos el predicado *(rut_bus ?bus ?de)*, donde el prefijo *?bus* indica qué unidad de transporte es y *?de* indica la ubicación de la unidad móvil.

Siguiendo con las unidades de transporte, a éstas se les debe indicar a donde puede desplazarse, para esto hacemos uso de *(rut_bus ?bus ?de ?a)*, donde *?bus*

indica qué autobús se está desplazando, *?de* representa la ubicación actual y *?a* el nodo donde se puede desplazarse desde *?de*.

Para conocer la ubicación del pasajero definimos el predicado *(at ?pas ?de)* donde *?de* del mismo modo que en *(rut_bus ?bus ?de)* se utiliza para conocer la ubicación actual en este caso del usuario *pas1*, mientras que la ruta que puede tomar un pasajero caminando a partir de *?de* es definida de la siguiente manera *(via_pas ?pas ?de ?a)* donde *?pas* indica el objeto pasajero, mientras *?de* de donde puede partir con destino a *?a*.

Estos predicados definidos anteriormente nos indican a donde se puede desplazar ya sea el pasajero o la unidad de transporte, así como también controla sus ubicaciones actuales al momento de correr el dominio de planificación para buscar un plan. Las ubicaciones de inicio ya sea para el usuario o las unidades se describirán a más detalle en el punto 3.5.2.

Para indicar que el pasajero se encuentra en una unidad de transporte, hemos definidos el predicado *(in ?pas ?bus)* donde *?pas* indica que pasajero está involucrado en ese predicado y *?bus* hace referencia a la unidad de transporte en la cual el pasajero se encuentra.

3.4.5 Function

Anteriormente se definió funtion, pero solo para recordar se puede decir que son aquellas funciones conformadas por objetos o como una simple variable que nos permiten el manejo de cantidades numéricas. A continuación definiremos las function necesarias para la implementación del dominio de planificación, así como su descripción correspondiente. En la tabla 3.2 se puede visualizar las function necesarias para cada acción a utilizar.

Primeramente definiremos aquellas function relacionadas directamente con los predicados mencionados en el punto anterior, para ser más precisos aquellos que indican de donde a donde se pueden realizar los movimientos. La primera de éstas es

(longitud_cam ?pas ?de ?a), la cual se utiliza para indicar la distancia entre el nodo *?de* al nodo *?a* para el pasajero *?pas*, esto se aplica cuando el pasajero se desplaza caminando entre dos puntos. Por otro lado tenemos *(tiempo_b ?bus ?de ?a)*, la cual contiene el tiempo de recorrido desde *?de* hacia *?a* para el autobús *?bus*.

Estas function actúan directamente sobre la métrica definida como *(longitud_c)* la cual acumula la distancia recorrida caminando por parte del usuario hasta la parada o al objetivo deseado a alcanzar en el plan a desarrollar, esta function se ve restringida por *(long_max_cam)*, la cual es la distancia máxima permisible a caminar hasta alcanzar una parada de autobuses, por lo tanto, concluimos que *(long_max_cam)* es una preferencia del usuario el cual determina qué valor tomará.

Para la distancia máxima a caminar por parte del usuario durante todo el plan tenemos dos function, la primera de ellas *(dist_max_c)* restringe la longitud acumulada en *(dist_maxima)*. Es decir *(dist_max_c)* guarda la información para esta preferencia de parte del usuario la cual es la limitante para poder generar un plan en base a lo longitud caminada, mientras que *(dist_maxima)* va guardando la distancia recorrida caminando.

Para esto de la longitud caminada la tenemos como métrica a optimizar al momento de generar el plan y es representada por la function *(longitud_caminada)*, la cual almacena la distancia recorrida por el usuario y que debe ser mínima para la generación del plan.

Una de las métricas a optimizar es el costo, para esto usaremos la function *(costo-total)* la cual acumula el costo económico del plan para su minimización. Esta métrica se va actualizando con los contenidos de *(costo ?bus)* cada vez que el usuario utilice una unidad.

Al hablarse de la parte económica muchas veces existe una cantidad limite a gastar en el viaje por parte del usuario, es por ello que implementamos la function *(costo_max)* el cual toma la cantidad deseada por el usuario a gastar, la cual restringe a *(costo-total)*.

El costo total *(costo-total)* se puede decir que es directamente proporcional al número de transferencias hechas durante el recorrido y también al costo de cada transferencia dada por las métricas anteriores. Esto es verdad únicamente si asumimos que el costo de los autobuses es igual para todas las unidades. Teniendo así que el *(costo-total) = (costo ?bus) * (num_ascensos)*.

Además que es muy incómodo para el usuario abordar un gran número de unidades de transporte, debido a diversos factores fuera de lo económico, como puede ser, embarazo, discapacidad, lesión, viaja con un bebé, tercera edad, cansancio, por ello se busca controlar este número de transferencias y la function correspondiente a ello es *(num_ascensos)*.

Para el caso donde se desea optimizar una combinación tiempo-costo se han implementado dos function, las cuales son valores ponderados asignados a las function *(p_costo)* y *(p_tiempo)*, de donde *(p_costo)* es el valor ponderado por peso de importancia para el costo económico, mientras que *(p_tiempo)* es el peso ponderado para el tiempo de recorrido.

La function *(pasajero_in_bus)* la utilizamos para identificar si el usuario ha ascendido a una unidad de transporte, esto para usar las acciones correspondientes al desplazamiento de las unidades debido a que los planificadores tienen problemas para trabajar con negaciones en sus condiciones.

Si *(pasajero_in_bus) > 0*, indica que el usuario se encuentra en el autobús *?bus*. Permitiendo así la aplicación de la acción *Desplazar_bus_pas*. En caso contrario, si *(pasajero_in_bus) = 0*, se puede hacer uso de la acción *Desplazar_bus*.

3.4.6 Implementación de las acciones

Una vez identificadas y descritas las acciones necesarias para nuestro dominio especificando que estas son durative-action, y partiendo de la tabla 3.2, donde se muestra que predicados, function y duration para cada durative-action especificada, se modelaran las diferentes acciones correspondientes al dominio de planificación que

proponemos, dando una explicación del porque utilizar cada una de estas componentes.

Acción Caminar

Como bien se mencionó esta acción permite el desplazamiento del pasajero caminando entre dos puntos, al ser del tipo durative-action es necesario asignarle un costo de recorrido, en este caso es el tiempo de recorrido que hace el usuario caminando y se define del modo siguiente *(=?duration (/(longitud_cam ?pas ?de ?a)1.39))*, donde 1.39 es la velocidad promedio de una persona al caminar y *(longitud_cam ?pas ?de ?a)* proporciona la distancia recorrida desde *?de* hasta *?a* por parte del pasajero *?pas*.

Es conocido que las action-durative están conformadas por condiciones y efectos, para el caso de las condiciones se es necesario conocer la ubicación actual del pasajero al inicio de la durative-action Caminar, cuyo predicado está representado por *(at ?pas ?de)*, de igual modo se le debe indicar a donde se puede desplazar, esto es dado por *(ir_cam ?pas ?de ?a)*, para ello se requiere de un pasajero *(pasajero ?pas)*.

He aquí donde presentamos nuestra primera preferencia por parte del usuario, donde él restringe la distancia máxima a caminar hasta una parada o al objetivo sin interrupciones, esto queda expresado de la siguiente manera *(longitud_c)<(long_max_cam)*. Por otro lado se muestra en esta parte de la acción las function correspondiente a la preferencia donde el usuario limita la distancia máxima a caminar durante todo el plan, donde *(dist_max_c)* acumula la distancia a recorrer por el usuario en el plan y *(dist_maxima)* contiene el valor límite asignado por el usuario, teniendo de este modo que *(dist_max_c)* deber ser menor que *(dist_maxima)*.

Los efectos realizados es que al finalizar la acción el pasajero se encuentra en *?a* es decir *(at ?pas ?a)*, y se incrementa la *(longitud_c)* con *(longitud_cam ?pas ?de ?a)*, lo mismo sucede con el tiempo total, es decir, también es incrementado de acuerdo

```
(:durative-action Caminar
        :parameters(?pas ?de ?a)
        :duration(=?duration (/(longitud_cam ?pas ?de ?a)1.39))
        :condition(and
                (at start(pasajero ?pas))
                (at end(<(longitud_c)(long_max_cam)))
                (at start(<(dist_max_c)(dist_maxima)))
                (at start(ir_cam ?pas ?de ?a))
                (at start(at ?pas ?de))

                )
        :effect(and
                (at start(not(at ?pas ?de)))
                (at end(at ?pas ?a))
                (at end(increase(dist_max_c)(longitud_cam ?pas ?de ?a)))
                (at end(increase(longitud_c)(longitud_cam ?pas ?de ?a)))
                (at end(increase(longitud_caminada)(longitud_cam ?pas ?de ?a)))
                )
)
```

Figura 3.10: Representación de la acción Caminar en nuestro modelo de planificación.

a la acción de cada movimiento realizado caminando. La durative-action Caminar queda definida como se muestra en la figura 3.10.

Dentro de los efectos también tenemos la function *(longitud_caminada)* la cual acumula la longitud recorrida caminando por parte del usuario durante todo el plan, siendo esta function una métrica a optimizar y para el caso de la longitud máxima a caminar en el plan la function *(dist_max_c)* es incrementada por *(longitud_cam ?pas ?de ?a)*.

Acción Ascender

La acción *Ascender*, ver figura 3.11, para poder llevarla a cabo se tiene como condiciones tener un pasajero *(pasajero ?pas)* y un autobús *(autobus ?bus)*, los cuales se requieren que estén en el mismo punto para que el usuario pueda hacer el ascenso, para esto necesitamos los predicados al inicio de la acción *(at ?pas ?de)* y *(rut_bus ?bus ?de)*. Todo lo anterior son las condiciones necesarias implementar para esta acción.

```
(:durative-action Ascender
        :parameters(?pas ?bus ?de)
        :duration (=?duration 0)
        :condition(and
                (at start(pasajero ?pas))
                (at start(autobus ?bus))
                (at start(rut_bus ?bus ?de))
                (at start(at ?pas ?de))
                (at start(<(costo-total)(costo_max)))
                )
            :effect(and
                (at start(not(at ?pas ?de)))
                (at end(in ?pas ?bus))
                (at end(decrease(longitud_c)(longitud_c)))
                (at end(increase(costo-total)(costo ?bus)))
                (at end(increase(num_ascensos)1))
                (at end(increase (pasajero_in_bus)100))
                )
    )
```

Figura 3.11: Representación de la acción Ascender en nuestro modelo de planificación.

El efecto principal es ascender al pasajero a una unidad de transporte, el predicado *(in ?pas ?bus)* carga con tal efecto, donde indica que el pasajero *?pas* está en el autobús *?bus*, negando entonces que esté al finalizar la acción en *?de* como se muestra en *(not(at ?pas ?de))*. En esta parte se manipulan varias function entre ellas se encuentra aquella que nos permite un control del número de ascensos *(num_ascensos)*, la cual es una métrica a optimizar si el usuario así lo desea. Por otro lado tenemos afectaciones para la function relacionada con el costo total es decir *(costo-total)*, la cual se incrementa de acuerdo al costo económico por abordar un autobús, *(costo ?bus)*.

En la figura 3.11 aparece la function *(costo_max)* la cual restringe a *(costo_total)* con la cantidad máxima a pagar por parte del usuario al momento de realizar su

viaje.

El incremento de *(pasajero_in_bus)* es solo para indicar si el usuario se encuentra en la unidad de transporte, ya que mediante la experimentación se vio que la aplicación de *(in ?pas ?bus)* no permitía la diferenciación en las acciones *Desplazar_bus* y *Desplazar_bus_pas*.

Mientras que la métrica *(longitud_c)* es decrementada por sí misma en esta acción, debido a que esta function solo nos permite restringir la distancia recorrida por el usuario a una parada o al objetivo sin ascensos, por lo tanto al momento de ascender *(longitud_c)* deberá tomar un valor de cero.

Acción Desplazar_bus_pas

El costo por llevar a cabo esta acción "duration" está dado por el tiempo de desplazamiento desde *?de* hacia *?a*, y es expresada por *(=?duration (tiempo_b ?bus ?de ?a))*.

La presente acción permite el movimiento de una unidad de transporte siempre y cuando contenga un pasajero, es decir, que *(pasajero_in_bus)* sea mayor a cero, por otro lado necesitamos una unidad de transporte público, cayendo en la necesidad de hacer uso del predicado *(autobus ?bus)*, para poder desplazar la unidad de transporte con el pasajero en su interior, necesitamos conocer su posición actual al inicio de la acción, la cual es expresada por *(rut_bus ?bus ?de)*, otra condición que debe presentarse es a donde se puede desplazar la unidad de transporte representado por *(via_bus ?bus ?de ?a)*. Esto se puede ver en la figura 3.12.

Los *effects* por llevar a cabo esta acción son los siguientes: primeramente tenemos que la unidad no se encontrara en *?de* es decir *(not(rut_bus ?bus ?de))*, pero ahora se encontrara en *?a* debido a que se desplazó hasta allí al finalizar la acción, *(rut_bus ?bus ?a)*.

```
(:durative-action Desplazar_bus_pas
        :parameters(?pas ?bus ?de ?a)
        :duration(=?duration (tiempo_b ?bus ?de ?a))
        :condition(and
                (at start(pasajero ?pas))
                (at start(autobus ?bus))
                (at start(in ?pas ?bus))
                (over all(in ?pas ?bus))
                (at start(>(pasajero_in_bus)0))
                (at start(rut_bus ?bus ?de))
                (at start(via_bus ?bus ?de ?a))

                )
        :effect(and
                (at start(not(rut_bus ?bus ?de)))

                (at end(rut_bus ?bus ?a))
                )
)
```

Figura 3.12: Representación de la acción Desplazar_bus_pas en nuestro modelo de planificación.

Acción Desplazar_bus

Permite el desplazamiento del autobús sin pasajero, es decir cuando *(pasajero_in_bus)* < 100, fuera de ésta condición las demás son similares que para *Desplazar_bus_pas*, es decir se requiere de una unidad *(autobus ?bus)*, de la ubicación actual de la unidad *(rut_bus ?bus ?de)*, y a donde se puede desplazar a partir de *?de*, *(via_bus ?bus ?de ?a)*.

De igual modo los effects son similares a los de *Desplazar_bus_pas*, con la diferencia que no se tiene la function *(tiempo-total)*, hay que indicar que ya no se está en *?de* si no en *?a*, y a donde se encontrará el autobús al finalizar la acción, es decir, *(rut_bus ?bus ?a)*. Esto se muestra en la figura 3.13.

```
(:durative-action Desplazar_bus
        :parameters(?bus ?de ?a)
        :duration (=?duration 0)
        :condition(and
            (at start(autobus ?bus))
            (at start(rut_bus ?bus ?de))
            (at start(via_bus ?bus ?de ?a))
            (at start(<(pasajero_in_bus)100))
            )
        :effect(and
            (at start(not(rut_bus ?bus ?de)))
            (at end(rut_bus ?bus ?a))
            )
    )
```

Figura 3.13: Representación de la acción Desplazar_bus en nuestro modelo de planificación.

```
(:durative-action Descender
        :parameters (?pas ?bus ?a)
        :duration (=?duration 0)
        :condition(and
            (at start(pasajero ?pas))
            (at start(autobus ?bus))
            (at start(in ?pas ?bus))
            (at start(rut_bus ?bus ?a))
            )
        :effect(and
            (at start(not (in ?pas ?bus)))
            (at start(not (rut_bus ?bus ?a)))
            (at end(at ?pas ?a))
            (at end(decrease (pasajero_in_bus)(pasajero_in_bus)))
            )
    )
```

Figura 3.14: Representación de la acción Descender en nuestro modelo de planificación.

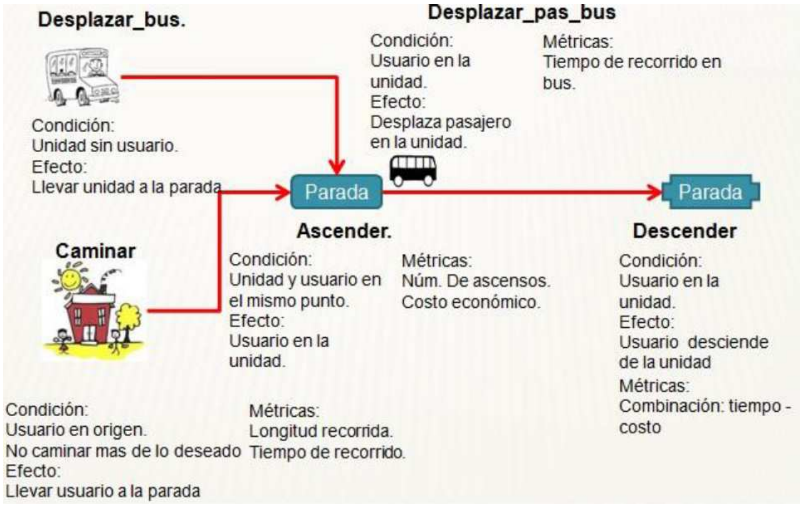


Figura 3.15: Representación de las acciones y sus respectivas conditions y effects, así como de las métricas más importantes a controlar.

Acción Descender

Si ya se está en la unidad de transporte el usuario caerá en la necesidad de descender, es de aquí la implementación de la presenta acción, que tiene carácter de ser *durative-action* pero su costo por llevarse a cabo es nulo, como *condition* tenemos que el usuario debe de estar en el algún autobús, *(in ?pas ?bus)*, por lo tanto necesitamos saber que pasajero y en que autobús se encuentra esté, los cuales son definidos por *(pasajero ?pas)* y *(autobus ?bus)*. Al tratarse de descender es necesario saber en qué lugar descenderá el usuario, para ello hacemos uso del predicado que nos indica la ubicación actual del transporte, la cual es *(rut_bus ?bus ?a)*.

Al cumplirse lo anterior se tiene que el pasajero ya no se encontrará en la unidad de transporte, denotado como *(not(in ?pas ?bus))*, la unidad ya no será tomada en cuenta, entonces se niega su actual ubicación *(not(rut_bus ?bus ?a))*, pero se le asigna una nueva ubicación al pasajero en *(at ?pas ?a)* y *(pasajero_in_bus)* es decrementado a cero al realizarse esta acción.

Todas estas *durative-action* son representadas gráficamente para una mejor comprensión, donde se puede ver claramente cómo se relacionan cada una de ellas,

figura 3.15, y las condiciones y efectos principales correspondientes a cada una, las métricas presentadas son aquellas que controlan de forma directa y/o son optimizadas.

3.4.7 Dominio de planificación para el transporte público con preferencias de los usuarios

Como resultado del análisis exhaustivo del problema de generación de viajes en el dominio del transporte público, se ha diseñado y producido un modelo basado en técnicas de planificación que constituye la principal aportación de esta tesis. Dicho dominio es presentado en su totalidad en la figura 3.16.

3.5 Archivo problema

Para poder resolver problemas de planificación aparte del archivo dominio necesitamos hacer uso de un archivo problema, figura 3.17. En este archivo problema se declaran los objetos participantes en la representación de la red de transporte.

Aquí mismo se indicá los estados iniciales de los elementos que conforman nuestro problema, por ejemplo, la ubicación del pasajero y/o de los autobuses, de a donde a donde se puede desplazar una unidad de transporte, así como también el usuario caminando. En este archivo se inicializan los valores numéricos de las function involucradas en el modelo de planificación. Aquí mismo se indica el objetivo a alcanzar y la métrica a optimizar.

En pocas palabras en este archivo se representa la red del sistema de transporte con todos los elementos que la conforman, así como todas las métricas involucradas, el objetivo a alcanzar, y para nuestro caso que preferencia se desea optimizar. A continuación iremos describiendo cada una de sus partes.

```
(define (domain ruta-transporte)
    (:requirements :numeric-fluents :durative-actions)

    (:predicates
        (pasajero ?x)
        (autobus ?r)
        (ir_cam ?x ?p1 ?p2)
        (at ?x ?p)
        (rut_bus ?r ?p)
        (via_bus ?r ?p1 ?p2)
        (in ?x ?r)
        )

    (:functions
        (longitud_cam ?pas ?de ?a)
        (tiempo_b ?bus ?de ?a)
        (pasajero_in_bus)
        (longitud_c)
        (long_max_cam)
        (dist_maxima)
        (dist_max_c)
        (costo-total)
        (costo_max)
        (costo ?bus)
        (num_ascensos)
        (longitud_caminada)
        (p_costo)
        (p_tiempo)


    )

    (:durative-action Caminar
        :parameters(?pas ?de ?a)
        :duration(=?duration (/(longitud_cam ?pas ?de ?a)1.39))
        :condition(and
            (at start(pasajero ?pas))
            (at end(<(longitud_c)(long_max_cam)))
            (at start(<(dist_max_c)(dist_maxima)))
            (at start(ir_cam ?pas ?de ?a))
            (at start(at ?pas ?de))

            )
        :effect(and
            (at start(not(at ?pas ?de)))
            (at end(at ?pas ?a))
            (at end(increase(dist_max_c)(longitud_cam ?pas ?de ?a)))
            (at end(increase(longitud_c)(longitud_cam ?pas ?de ?a)))
            (at end(increase(longitud_caminada)(longitud_cam ?pas ?de ?a)))
            )
    )

(:durative-action Ascender
        :parameters(?pas ?bus ?de)
        :duration (=?duration 0)
        :condition(and
                (at start(pasajero ?pas))
                (at start(autobus ?bus))
                (at start(rut_bus ?bus ?de))
                (at start(at ?pas ?de))
                (at start(<(costo-total)(costo_max)))
                )
            :effect(and
                (at start(not(at ?pas ?de)))
                (at end(in ?pas ?bus))
                (at end(decrease(longitud_c)(longitud_c)))
                (at end(increase(costo-total)(costo ?bus)))
                (at end(increase(num_ascensos)1))
                (at end(increase (pasajero_in_bus)100))
                )
    )

    (:durative-action Desplazar_bus_pas
        :parameters(?pas ?bus ?de ?a)
        :duration(=?duration (tiempo_b ?bus ?de ?a))
        :condition(and
            (at start(pasajero ?pas))
            (at start(autobus ?bus))
            (at start(in ?pas ?bus))
            (over all(in ?pas ?bus))
            (at start(>(pasajero_in_bus)0))
            (at start(rut_bus ?bus ?de))
            (at start(via_bus ?bus ?de ?a))
            )
        :effect(and
            (at start(not(rut_bus ?bus ?de)))
            (at end(rut_bus ?bus ?a))
            )
    )

(:durative-action Desplazar_bus
        :parameters(?bus ?de ?a)
        :duration (=?duration 0)
        :condition(and
            (at start(autobus ?bus))
            (at start(rut_bus ?bus ?de))
            (at start(via_bus ?bus ?de ?a))
            (at start(<(pasajero_in_bus)100))
            )
        :effect(and
            (at start(not(rut_bus ?bus ?de)))
            (at end(rut_bus ?bus ?a))
            )
    )

(:durative-action Descender
        :parameters (?pas ?bus ?a)
        :duration (=?duration 0)
        :condition(and
            (at start(pasajero ?pas))
            (at start(autobus ?bus))
            (at start(in ?pas ?bus))
            (at start(rut_bus ?bus ?a))
            )
        :effect(and
            (at start(not (in ?pas ?bus)))
            (at start(not (rut_bus ?bus ?a)))
            (at end(at ?pas ?a))
            (at end(decrease (pasajero_in_bus)(pasajero_in_bus)))
            )
    )
)
```

Figura 3.16: Dominio del modelo de planificación correspondiente al problema de rutas de transporte público.

3.5.1 Objects

Son las partículas necesarias que representan los diversos objetos que componen la red del transporte público. A continuación presentamos un listado de estos objects.

Nodos: Los nodos, aunque estáticos, representan la conexión de la red, imprescindible para que se puedan llevar a cabo las acciones de desplazamiento entre dos puntos.

Para caso de experimentación lo estamos representando como $c\#$, donde $\#$ es el número de identificación correspondiente a ese nodo. Las paradas son representadas por nodos, solo que estas en los estados iniciales se les antepone *?bus*.

Pasajero: es la partícula o elemento en nuestro universo al cual se le va a realizar el plan de viaje, lo representamos como *"pas1"*. Se enumera por si se decide extender el modelo a más de un usuario.

Autobuses: Unidades de transporte utilizadas para la generación del plan, estas partículas son representadas del siguiente modo $ruta\_\%\_\psi$, donde $\%$ representa el número de ruta correspondiente y ψ indica el sentido o dirección de la ruta, por ejemplo *Ruta_1_1* indica que es una unidad de la ruta *1* que corre hacia la dirección *1*.

3.5.2 Init

Es una lista de todos los predicados que son verdaderos en el estado inicial del dominio, y que son necesarios para el manejo de las condiciones para poder realizar cambios en los efectos de las acciones del dominio de planificación.

Como es de esperarse el primer estado inicial que debemos definir es la ubicación actual del usuario, la cual es proporcionada por quien hace uso del sistema, esto queda definido de la siguiente manera, *(at pas1 c#)*, donde $c\#$ es la ubicación del pasajero.

Por otro lado debemos iniciar las partículas pasajero y autobús. Para la primera se hace del siguiente modo *(pasajero pas1)*, mientras que para la segunda tenemos *(autobus ruta_ %_ψ)*, donde % indica la ruta a la que pertenece la unidad y ψ indica la dirección o sentido que sigue (ida o regreso). Para cada unidad se debe especificar su ubicación inicial al momento de llevar a cabo la consulta por parte del usuario, esto se hace de la siguiente manera, *(rut_bus ruta_ %_ψ c#)*, o sea la ruta % en dirección ψ se encuentra ubicada en el nodo *c#*, donde # es el número de nodo en donde está ubicada la unidad de transporte correspondiente.

El siguiente estado inicial es aquella relacionada con la function *(costo ?bus)*, donde se indica el costo económico por abordar una unidad de transporte público específico, esto queda expresado de la siguiente forma en nuestro archivo problema, *(=(costo rut_ %_ψ)$)*, esto se lee, el costo para la *ruta_ %_ψ* es igual a *$*, donde *$* es un valor numérico referente al precio por abordaje.

Seguido de esto indicamos la ruta a seguir por cada unidad de transporte en el sistema, es decir, a donde se puede mover desde su posición actual, quedando de la siguiente manera, *(via_bus rut_ %_ψ c#1 c#2)*, es decir el autobús % con dirección ψ puede desplazarse del nodo *c#1* al nodo *c#2*, pero hacer este movimiento incurre en un costo temporal, este es asignado por recorrido entre nodo para cada unidad de transporte, siendo expresado como *(=(tiempo_b rut_ %_ψ c#1 c#2)τ)*. Donde τ representa el tiempo de duración por desplazarse del nodo *c#1* al nodo *c#2*, para la unidad % en dirección ψ, esto se hace por cada ruta de transporte público a considerar y por el número de lugares a los cuales se puede llegar en ellas.

Anteriormente se definió el estado inicial para el pasajero y como en todo traslado este debe de desplazarse caminando aunque sea un pequeño segmento, esto se representa de manera similar que el desplazamiento de la unidad, es decir, *(ir_cam pas1 c#1 a c#2)*, pero a este movimiento va asociado un peso el cual se representa del siguiente modo, *(=(longitud_cam pas1 c#1 c#2)μ)*, donde μ representa la distancia a recorrer del nodo *c#1* al *c#2* para el pasajero *pas1* caminando.

Como bien se ha visto en este apartado, hay que inicializar todas las function utilizadas en nuestro dominio. La function especificada para acumular la distancia recorrida hasta una parada de autobús o al objetivo es inicializada a cero, quedando de la siguiente manera *(=(longitud_c)0)*, esta métrica se ve restringida por la longitud máxima deseada a caminar hasta la parada cuya function se inicializa por un valor de entrada por parte del usuario, quedando: *(=(longitud_max_cam)ϕ)*, donde ϕ es el valor de esta preferencia de entrada.

Otra de las preferencias implementadas en este modelo de planificación es la correspondiente a la longitud máxima deseada a caminar durante todo el plan. Para ello hemos asignado la function *(dist_maxima)*, la cual acumula la distancia recorrida por el usuario caminando durante todo el plan de viaje, su valor de inicio es de cero, *(=(dist_maxima)0)* y *(=(dist_max_c)α)*, la cual restringe la longitud máxima a caminar en todo el recorrido por parte del usuario, donde α es la cantidad numérica de esta preferencia, que de igual modo que para el alcance a la parada, esta es ingresada por parte del usuario.

Para el caso del costo económico del recorrido, teniendo que este tiene un valor de cero al inicio, *(=(costo-total)0)*, brindamos la oportunidad al usuario de restringirlo de acuerdo al gasto deseado al hacer en su traslado, por lo tanto el pasajero ingresa esta cantidad monetaria al momento de indicar sus preferencias, quedando esto inicializado de la siguiente manera *(=(costo_max)β)*. Donde β es la cantidad máxima a gastar durante el viaje.

Todas las demás function son igualadas a cero en su estado inicial, estas se enlistan a continuación, quedando de la siguiente manera:

(=(tiempo-total)0)

(=(pasajero_in_bus)0)

(=(num_ascensos)0)

(=(tiempo-costo)0)

(=(p_costo)0)

(=(p_tiempo)0)

3.5.3 Objetivo

El objetivo, representado como goal en nuestro archivo problema, es el punto deseado alcanzar por el usuario, por lo tanto es ingresado al momento de hacerse la consulta junto con todas las preferencias deseadas para la planeación del viaje. Esto se representa del siguiente modo, donde # es el punto deseado a alcanzar por el usuario:

(:goal (at pas1 c#))

3.5.4 Métricas a optimizar

Para un modelo estándar donde no se considera el tiempo de espera, se tiene 5 funciones objetivos a optimizar, los cuales son:

- Costo económico del viaje.
- Número de transferencias.
- Tiempo de recorrido.
- Combinación tiempo costo.
- Distancia recorrida caminando.

Estas se representaron mediante function en el archivo dominio y se inicializaron en el punto anterior de este trabajo, ahora daremos paso como se define en PDDL 2.1 la función objetivo para cada una de estas métricas.

Costo económico del viaje.

metric minimize (costo-total)

Número de transferencias.

metric minimize (num_ascensos)

Tiempo de recorrido.

metric minimize (total-time)

Combinación tiempo-costo.

metric minimize (+((total-time)(p_tiempo))(*(costo-total)(p_costo)))*

Lo que se hace en esta función es la minimización de la sumatoria del producto resultante de *(total-time)* con *(p_tiempo)* y de *(costo-total)* con *(p_costo)*. Anteriormente se describió estas function.

Distancia recorrida caminando.

(:metric minimize (long_recorrida_caminando))

Total-time es una variable estandarizada de PDDL 2.1, donde se acumula todos los costos por llevar a cabo una durative-action, para nuestro dominio estas tienen relevancia en *Caminar* y en *Desplazar_bus_pas*.

3.5.5 Archivo problema

La forma del archivo problema a considerar se puede ver en la figura 3.17, donde se muestra las componentes correspondientes a este archivo y su organización.

```
(define(problem ruta8)
(:domain ruta-transporte)

(:objects
pas1  ruta_1_1 ruta_1_2 ruta_2_1 ruta_2_2
c1 c2 c3 c4 c5 c6 c7 c8 c9 c10 c11 c12 c13 c14 c15 c16 c17
c18 c19 c20 c21 c22 c23 c24 c25 c26 c27 c28 c29 c30 c31
c32 c33 c34 c35 c36 c37 c38 c39 c40 c41 c42 c43 c44 c45
c46 c47 c48 c49 c50 c51 c52 c53 c54 c55 c56 c57 c58 c59
c60 c61 c62 c63 c64 c65 c66 c67 c68 c69 c70 c71 c72 c73
c74 c75 c76 c77 c78 c79 c80 c81 c82 c83 c84 c85 c86 c87
c88 c89 c90 c91 c92 c93 c94 c95 c96 c97 c98 c99 c100)

(:init (pasajero pas1)
(autobus ruta_1_1) (autobus ruta_1_2) (autobus ruta_2_1)
(autobus ruta_2_2) (at pas1 c1) (rut_bus ruta_1_1 c31)
(rut_bus ruta_1_2 c43) (rut_bus ruta_2_1 c86) (rut_bus
ruta_2_2 c80)

(=(costo ruta_1_1)10)(=(costo ruta_1_2)10)(=(costo ruta_2
_1)10)(=(costo ruta_2_2)10)

(via_bus  ruta_1_1 c31 c21)
(=(tiempo_b  ruta_1_1 c31 c21)20)
(via_bus  ruta_1_2 c21 c31)
(=(tiempo_b  ruta_1_2 c21 c31)20)
                    .
                    .
                    .

 (via_bus  ruta_1_2 c43 c33)
 (=(tiempo_b  ruta_1_2 c43 c33)17)
                    .
                    .
                    .

 (=(tiempo_b  ruta_2_2 c87 c86)17)
 (=(tiempo_b  ruta_2_1 c70 c80)11)
 (via_bus  ruta_2_2 c80 c70)
 (=(tiempo_b  ruta_2_2 c80 c70)11)

(ir_cam pas1 c1 c2)
(=(longitud_cam pas1 c1 c2)100)
(ir_cam pas1 c2 c1)
(=(longitud_cam pas1 c2 c1)100)
(ir_cam pas1 c2 c3)
(=(longitud_cam pas1 c2 c3)100)
(ir_cam pas1 c3 c2)
(=(longitud_cam pas1 c3 c2)100)
                    .
                    .
                    .
(ir_cam pas1 c90 c100)
(=(longitud_cam pas1 c90 c100)100)
(ir_cam pas1 c100 c90)
(=(longitud_cam pas1 c100 c90)100)

(=(pasajero_in_bus)0)
(=(longitud_c)0)
(=(long_max_cam)500)
(=(dist_maxima)1000)
(=(dist_max_c)0)
(=(costo-total)0)
(=(num_ascensos)0)
(=(costo_max)20)
(=(p_costo)0.25)
(=(p_tiempo)0.75)
(=(longitud_caminada)0)
)
(:goal (and (at pas1 c100)))
(:metric minimize(total-time))
 )
```

Figura 3.17: Archivo problema correspondiente al problema de rutas de transporte público.

Capítulo 4

Experimentos y resultados

Para esta sección se desarrollaron diversos modelos de planificación, teniendo todos como base el modelo presentado al final del capítulo anterior. El primer modelo que presentaremos es una versión relajada de nuestro modelo final, esto con el fin de poderlo comparar directamente con una implementación del algoritmo de Dijkstra. El modelo representa una red de transporte con solamente dos acciones y algunas métricas sencillas comunes.

Los demás modelos utilizan todas las acciones mencionadas. Para cuestiones de experimentación se desarrolló un modelo de planificación para cada función objetivo, y partiendo de ellos se agregan las diferentes preferencias a considerar por parte del usuario que funcionan como restricciones al momento de computar un plan de viaje, teniendo una variedad de sub-modelos a partir de uno de estos. En base a lo anterior, nuestro diseño de experimentos se determinó de acuerdo a las preferencias del usuario y a las métricas de optimización las ejecuciones mostradas en la tabla 4.1.

Métrica a optimizar: Tiempo.	**Modelo básico**	**Modelo completo**
Sin restricción.	Sección 4. 2. 1	Seccion 4. 2. 2
Con restricción de longitud máxima a caminar de las paradas.	Sección 4. 2. 1	Seccion 4. 2. 2
Con restricción de longitud máxima a caminar durante todo el plan.	Sección 4. 2. 1	Seccion 4. 2. 2
Combinación de las restricciones de longitud máxima a caminar de la parada y a caminar durante todo el plan.	Sección 4. 2. 1	Seccion 4. 2. 2

Tabla 4.1: Corridas de experimentos para la métrica a optimizar.

Aunque el objetivo y aportación de esta tesis es la implementación de un modelo donde se representen todas las preferencias del usuario, nuestro diseño de

experimentos permitirá la evaluación de cada una de las funciones objetivo por separado, con el fin de determinar las limitantes de los modelos de planificación y así dejar la puerta abierta a futuras investigaciones.

Para poder probar nuestros dominios hemos creado diversos archivos problema, donde las rutas de autobuses son generadas de forma aleatoria. El tiempo de traslado entre paradas se asigna de forma aleatoria. A cualquier nodo se puede llegar caminando y su métrica está regida por la longitud entre ellos, la cual es de 100 metros para cada arista. Cada archivo problema se diferencia de los demás por las métricas a utilizar según lo explicado anteriormente. En la tabla 4.2 se presenta las características de las redes a tratar.

Núm. De nodos	Núm. De rutas	Tamaño de las rutas	Nodo inicio	Nodo objetivo
100	6	11	1	100
144	6	15	1	144
225	7	20	1	225
289	5	40	1	289
400	6	45	1	400
625	4	120	1	625

Tabla 4.2: Caraterísticas de las redes a usar.

El número de nodos representa la red por donde se puede desplazar el usuario caminando, mientras que las rutas de autobuses se consideran que son de ida y vuelta por lo tanto se puede decir que se tiene el doble de las rutas mostradas en la tabla 4.2 en la columna Núm. de rutas. El número de paradas por dirección se muestran en la columna Tamaño de las rutas.

Los datos que presentaremos como resultados de nuestros experimentos con los planificadores son los siguientes:

- Tiempo de ejecución.
- Calidad del plan.

Donde el tiempo de ejecución es el tiempo que tarda el planificador en obtener un plan de viaje para el usuario. La calidad del plan se refiere al valor final de la métrica a optimizar.

4.1 Modelo básico

Como lo mencionamos anteriormente, para poder hacer una comparativa de nuestro modelo de planificación con algún método de los denominados clásicos, se decidió implementar el Algoritmo de Dijkstra clásico [19] [35] [62], con diversas modificaciones para representar algunas de las preferencias de los usuarios como distancia máxima a caminar total y entre paradas.

En este modelo básico de planificación solo se modelaron dos acciones, las cuales corresponden a aquellas que nos permiten desplazarnos, éstas son: *Caminar* y *Desplazar_bus*, y que incluyen las preferencias antes mencionadas.

4.1.1 Modelo básico: Tiempo de recorrido

Tal como se mencionó y como se puede deducir, ésta parte trata de optimizar el tiempo de duración del plan, contemplando diversas combinaciones de las preferencias de distancia caminada por parte de los usuarios. Primeramente se hará la experimentación usando un modelo sin éstas restricciones para el Algoritmo de Dijkstra y para el modelo de planificación utilizando los mismos problemas. Una vez hecho ésto, se procederá hacer experimentación agregando las preferencias de los usuarios: distancia máxima a caminar a la parada, distancia máxima a caminar durante el plan de viaje y la combinación de éstas restricciones. Todos los resultados obtenidos en orden de aparición serán presentados a continuación.

Para el planificador PLAN se ejecutaron los experimentos en un equipo diferente al usado para los demás planificadores por el tipo de ejecutable de dicho solver. Las características del equipo utilizado para el planificador PLAN son las siguientes:

- Procesador Intel(R) Core(TM) i3-2310M CPU @ 2.10GHz
- Disco duro: 30 GB
- Memoria RAM: 4GB
- Sistema Operativo: 64 bits

Modelo básico: Tiempo de recorrido sin preferencias

En esta sección presentamos los resultados obtenidos por nuestro modelo básico (Figura 4.1) que no considera restricciones o preferencias. Como podemos observar en la Figura 4.1, nuestro modelo considera únicamente dos acciones, la primera representa el caminar de los usuarios a las y de las paradas, y la segunda modela el desplazamiento de los autobuses. Podemos notar fácilmente la ausencia de preferencias, siendo el tiempo de duración del plan la única métrica a optimizar.

Para tener una dimensión real de cómo se comporta nuestro modelo básico, implementamos una versión tradicional de Dijkstra. El propósito de este primer conjunto de experimentos consiste en verificar si un modelo básico de planificación es capaz de escalar y ser competitivo con respecto a un benchmark, en este caso Dijkstra. Los resultados en términos de los tiempos de ejecución de estos experimentos pueden observarse en la Figura 4.2 y en la Tabla 4.3. Cabe la pena resaltar dos aspectos de los experimentos, el primero es que no todos los planificadores pudieron escalar a los problemas más complejos. Esto nos da una indicación de que el modelo, aunque simplificado, es bastante complejo. Creemos que la complejidad radica en que cada posible nodo en la red es un destino u origen potencial, lo cual incrementa la complejidad del proceso de búsqueda si no diferenciamos entre ellos. El segundo aspecto a resaltar es que uno de los planificadores (SGPLAN) resuelve todos los problemas, y es más eficiente que la solución tradicional (Dijkstra). SGPLAN particiona el espacio de búsqueda dados los objetivos, lo que le permite diferenciar entre ellos, y así evitar esos nodos en la red que no son relevantes. Como lo mencionabamos

```
(define (domain ruta-transporte)
    (:requirements :numeric-fluents :durative-actions)
    (:predicates
        (pasajero ?x)
        (autobus ?r)
        (ir_cam ?x ?p1 ?p2)
        (at ?x ?p)
        (via_bus ?x ?r ?p1 ?p2)
        )
    (:functions
        (longitud_cam ?pas ?de ?a)
        (tiempo_b ?pas ?bus ?de ?a)
    )
    (:durative-action Caminar
        :parameters(?pas ?de ?a)
        :duration(=?duration (/(longitud_cam ?pas ?de ?a)1.39))
        :condition(and
            (at start(pasajero ?pas))
            (at start(at ?pas ?de))
            (at start(ir_cam ?pas ?de ?a))
            )
        :effect(and
            (at start(not(at ?pas ?de)))
            (at end(at ?pas ?a))
            )
    )
    (:durative-action Desplazar_bus_pas
        :parameters(?pas ?bus ?de ?a)
        :duration(=?duration (tiempo_b ?pas ?bus ?de ?a))
        :condition(and
            (at start(pasajero ?pas))
            (at start(autobus ?bus))
            (at start(at ?pas ?de))
            (at start(via_bus ?pas ?bus ?de ?a))
            )
        :effect(and
            (at start(not(at ?pas ?de)))
            (at end(at ?pas ?a))
            )))
```

Figura 4.1: Modelo básico de planificación de rutas de viaje sin restricciones.

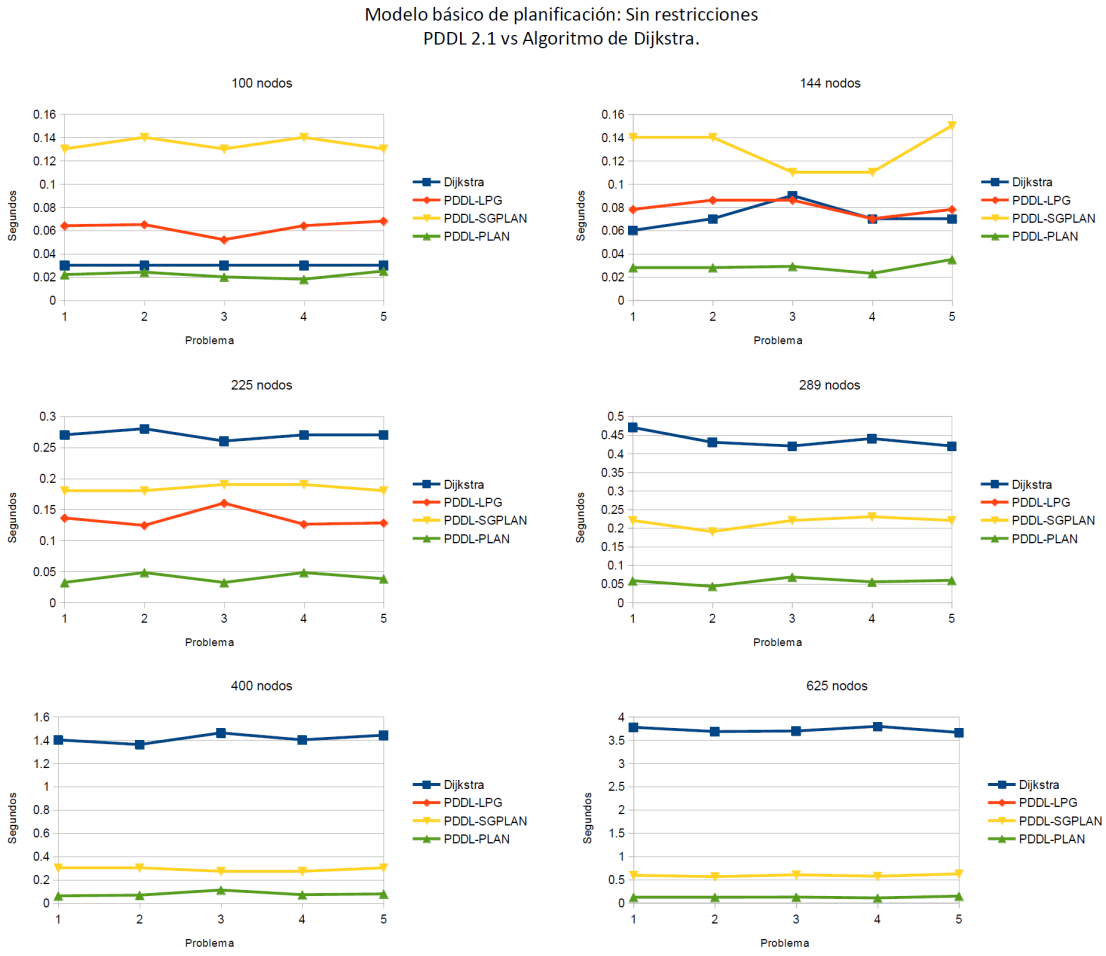


Figura 4.2: Resultados experimentales obtenidos de un modelo básico sin restricciones. Se reportan tiempos de la síntesis de solución en segundos.

# Nodos	Archivo	Dijkstra	LPG-TD	SGPLAN	PLAN
100	1	0.03	0.064	0.13	0.022
	2	0.03	0.065	0.14	0.024
	3	0.03	0.052	0.13	0.02
	4	0.03	0.064	0.14	0.018
	5	0.03	0.068	0.13	0.025
144	1	0.06	0.078	0.14	0.028
	2	0.07	0.086	0.14	0.029
	3	0.09	0.086	0.11	0.023
	4	0.07	0.07	0.11	0.035
	5	0.07	0.078	0.15	0.024
225	1	0.27	0.136	0.18	0.032
	2	0.28	0.124	0.18	0.048
	3	0.26	0.16	0.19	0.038
	4	0.27	0.126	0.19	0.036
	5	0.27	0.128	0.18	0.042
289	1	0.47	—	0.22	0.058
	2	0.43	—	0.19	0.043
	3	0.42	—	0.22	0.068
	4	0.44	—	0.23	0.055
	5	0.42	—	0.22	0.059
400	1	1.4	—	0.3	0.059
	2	1.36	—	0.3	0.065
	3	1.46	—	0.27	0.109
	4	1.4	—	0.27	0.069
	5	1.44	—	0.3	0.076
625	1	3.77	—	0.59	0.118
	2	3.68	—	0.56	0.118
	3	3.69	—	0.6	0.122
	4	3.79	—	0.57	0.103
	5	3.66	—	0.62	0.142

Tabla 4.3: Tiempo de ejecución en segundos obtenidos del modelo básico de planificación sin restricciones.

anteriormente, este primer experimento demuestra que, aunque el problema es complejo, es viable resolverlo usando técnicas de planificación. Por otro lado se tienen

los tiempos obtenidos para el planificador PLAN, aunque se corrío en otro equipo se puede observar que sus tiempos de ejecución son muy pequeños.

# Nodos	Archivo	Dijkstra	LPG-TD	SGPLAN	PLAN
100	1	534	832	580	534
	2	331	713.75	679	331
	3	439	642.8	676	439
	4	346	664.2	629	346
	5	389	1137.4	1189	389
144	1	463	844.6	1528	463
	2	618	1373.4	1528	618
	3	491	1408.8	884	491
	4	718	1160.2	884	718
	5	481	1063.2	877	481
225	1	598	1137.6	1100	598
	2	821	921.626	1035	821
	3	1036	1872	2016	1036
	4	747	1080	1041	747
	5	738	1382.4	1039	737
289	1	1071	—	1854	1071
	2	636	—	1176	636
	3	875	—	1244	875
	4	1071	—	2243	1071
	5	892	—	1242	892
400	1	603	—	1407	603
	2	1143	—	1345	1143
	3	840	—	1451	840
	4	806	—	1179	806
	5	1007	—	1398	1007
625	1	697	—	1525	697
	2	1314	—	1811	1314
	3	1358	—	1816	1358
	4	598	—	1317	598
	5	1155	—	3456	1155

Tabla 4.4: Duración del plan obtenidos del modelo básico de planificación sin restricciones.

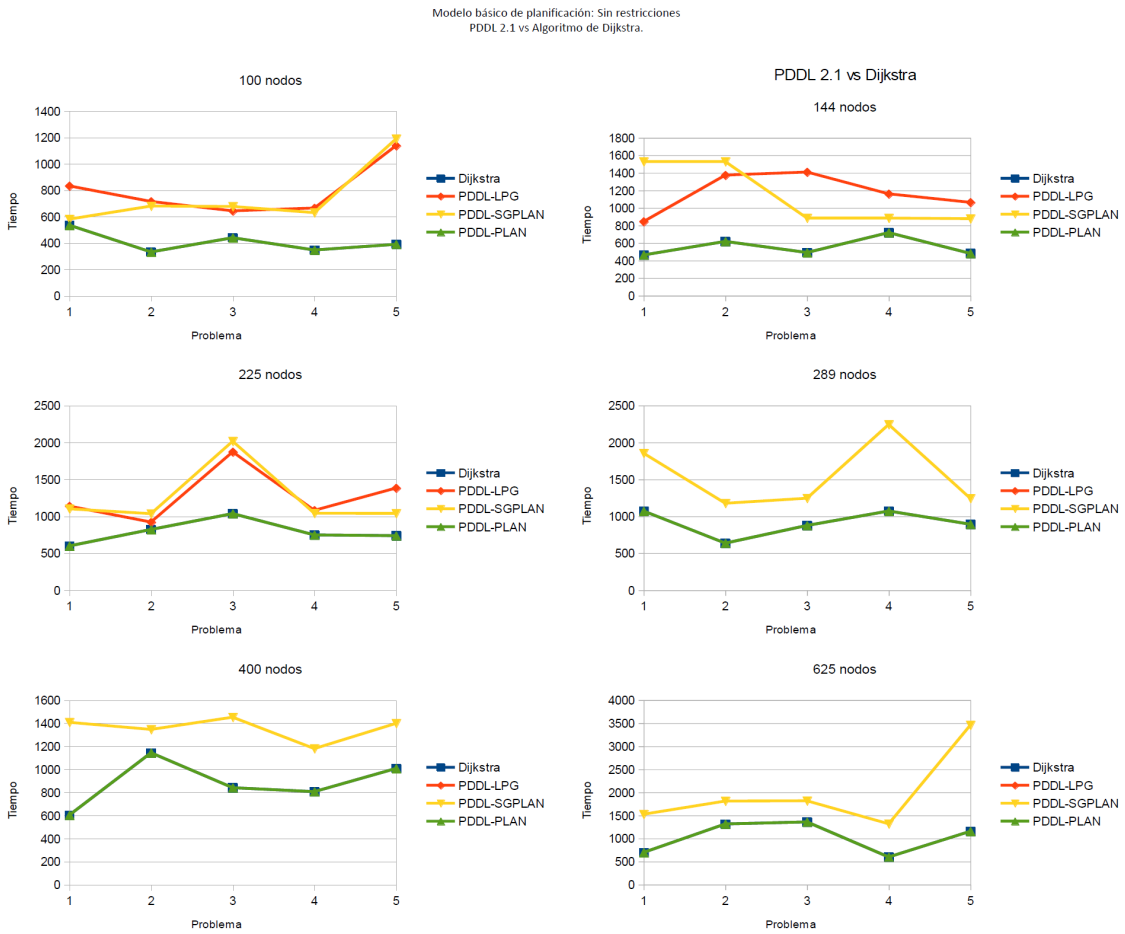


Figura 4.3: Resultados experimentales obtenidos de un modelo básico sin restricciones para el tiempo de duración del plan.

```
(:functions
        (longitud_cam ?pas ?de ?a)
        (longitud_c)
        (long_max_cam)
        (tiempo_b ?pas ?bus ?de ?a)
        (aux_1)
)
```

Figura 4.4: Function del modelo básico con restrición de longitud a caminar para alcanzar la parada de autobuses.

En la Tabla 4.4 y Figura 4.3 mostramos los tiempos de duración de los planes obtenidos por los planificadores que estamos evaluando. Es muy importante hacer notar que uno de los planificadores (PLAN) encuentra la solución óptima como Dijkstra pero sin incurrir en los costos de tiempo que Dijkstra. Este aspecto es realmente relevante si consideramos que el modelo que estamos resolviendo es un modelo básico y sin restricciones, y de no haberlo resuelto de manera eficiente, tendríamos pocas probabilidades de éxito en los modelos completos que involucran múltiples preferencias o restricciones.

Modelo básico: Tiempo de recorrido-distancia máxima a caminar a la parada

En esta parte se realizó una modificación al algoritmo de Dijkstra y a nuestro modelo de planificación limitando o restringiendo la distancia a caminar para alcanzar alguna parada de autobuses, ya sea desde el punto de origen o desde alguna parada a otra debido a la necesidad de hacer una transferencia y/o para ir desde una unidad de transporte al objetivo. El dominio de planificación es similar al de la figura 4.1, solo que se agregan un par de funciones correspondientes a las métricas de esta preferencia del usuario. En la figura 4.4 se puede ver cómo queda definido la sección functión de nuestro modelo de planificación.

En la acción *Caminar* de la figura 4.5 agregamos la preferencia de la distancia

```
(:durative-action Caminar
        :parameters(?pas ?de ?a)
        :duration(=?duration (/(longitud_cam ?pas ?de ?a)1.39))
        :condition(and
            (at start(<=(longitud_c)(long_max_cam)))
            (at start(pasajero ?pas))
            (at start(at ?pas ?de))
            (at start(ir_cam ?pas ?de ?a))
            )
        :effect(and
            (at start(not(at ?pas ?de)))
            (at end(at ?pas ?a))
            (at end(increase (longitud_c)(longitud_cam ?pas ?de ?a)))
            )
)
```

Figura 4.5: Acción Caminar del modelo básico con restrición de longitud a caminar para alcanzar la parada de autobuses.

máxima a caminar a una parada o al objetivo, la cual queda definida como *(at start(<=(longitud_c) (long_max_cam)))*, donde *(longitud_c)* representa la distancia caminada por parte del usuario la cual debe ser menor que la restricción establecida por el mismo para que la acción se pueda ejecutar. Una vez ejecutada la acción, esta misma métrica se incrementa con la longitud del segmento caminado. Como esta métrica solamente considera la distancia a caminar entre paradas, se decrementa a cero una vez que el usuario toma el autobús tal y como se ve en la Figura 4.6.

Podemos observar en la Figura 4.7 los tiempos reportados para sintetizar la solución por parte de los planificadores y Dijkstra. Notase que los tiempos son muy buenos, especialmente para el planficador SGPLAN que es hasta 3 veces más eficiente que Dijkstra. Mientras tanto el planificador PLAN muestra tiempos muy bajos para este conjunto de problemas, mostrando una vez mas su superioridad en esta sección de modelo básico. Para un mayor entendimiento estos resultados se muestran en la Tabla 4.5.

```
(:durative-action Desplazar_bus_pas
                :parameters(?pas ?bus ?de ?a)
                :duration(=?duration (tiempo_b ?pas ?bus ?de ?a))
                :condition(and
                        (at start(pasajero ?pas))
                        (at start(autobus ?bus))
                        (at start(at ?pas ?de))
                        (at start(via_bus ?pas ?bus ?de ?a))
                        )
                :effect(and
                        (at start(not(at ?pas ?de)))
                        (at end(at ?pas ?a))
                        (at end(decrease(longitud_c)(longitud_c)))
                        ))
)
```

Figura 4.6: Acción Desplazar_bus_pas del modelo básico con restrición de longitud a caminar para alcanzar la parada de autobuses.

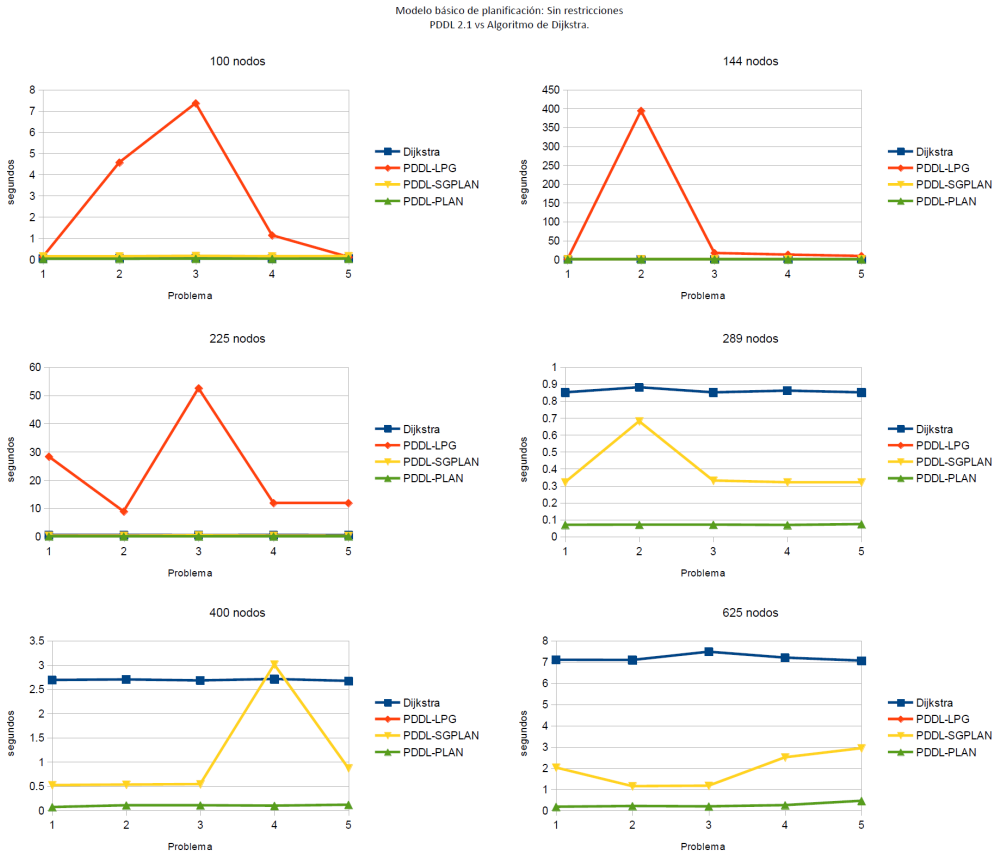


Figura 4.7: Tiempos de ejecución de los planificadores para el modelo de planificación correspondiente a la preferencia de la longitud a caminar a la parada de autobuses.

# Nodos	Archivo	Dijkstra	LPG-TD	SGPLAN	PLAN
100	1	0.04	0.118	0.14	0.02
	2	0.05	4.566	0.14	0.02
	3	0.05	7.36	0.16	0.027
	4	0.04	1.132	0.14	0.023
	5	0.05	0.11	0.15	0.026
144	1	0.14	0.176	0.21	0.035
	2	0.14	393.976	0.22	0.043
	3	0.13	16.542	0.19	0.04
	4	0.13	12.272	0.28	0.037
	5	0.13	8.626	0.22	0.029
225	1	0.55	28.312	0.29	0.067
	2	0.56	8.886	0.27	0.053
	3	0.54	52.48	0.43	0.057
	4	0.55	11.87	0.29	0.059
	5	0.54	11.87	0.24	0.069
289	1	0.85	—	0.32	0.07
	2	0.88	—	0.68	0.068
	3	0.85	—	0.33	0.073
	4	0.86	—	0.32	0.072
	5	0.85	—	0.32	0.067
400	1	2.69	—	0.52	0.104
	2	2.7	—	0.53	0.096
	3	2.68	—	0.54	0.115
	4	2.71	—	3.01	0.14
	5	2.67	—	0.87	0.11
625	1	7.1	—	2.02	0.168
	2	7.09	—	1.14	0.203
	3	7.48	—	1.16	0.186
	4	7.2	—	2.5	0.245
	5	7.06	—	2.93	0.449

Tabla 4.5: Tiempo de ejecución en segundos para la obtención de un modelo básico de planificación con restricción a la parada.

Los tiempos de duración de los planes resultantes puede verse en la Tabla 4.6 y Figura 4.8. El planificador PLAN obtuvo en casi todas las ejecuciones el valor

# Nodos	Archivo	Dijkstra	LPG-TD	SGPLAN	PLAN
100	1	452	597.4	687	452
	2	477	1358.2	563	477
	3	432	938.2	822	432
	4	448	822.8	622	448
	5	415	655	615	415
144	1	434	788.8	545	434
	2	508	1450.8	826	508
	3	483	1584.2	1506	483
	4	492	1562.8	1407	492
	5	645	857.4	820	645
225	1	614	1152	1015	614
	2	364	691.2	805	364
	3		1756.8	1124	393
	4	815	1396.8	1217	815
	5	726	1396.8	986	726
289	1	387	—	963	387
	2	746	—	1275	746
	3	709	—	1214	709
	4	693	—	838	693
	5	490	—	1020	490
400	1	740	—	1230	740
	2	643	—	1277	643
	3	832	—	1433	832
	4	974	—	2437	974
	5	656	—	1431	656
625	1	800	—	1647	913
	2	932	—	1670	932
	3	619	—	1238	619
	4	972	—	1820	972
	5	1234	—	1797	1234

Tabla 4.6: Duración del plan arrojado por un modelo básico de planificación con restricción a la parada.

óptimo de la función a minimizar, excepto para el problema 1 de 625 nodos. SGPLAN presenta tiempos de duración del plan algo altos, pero aun así se puede decir que

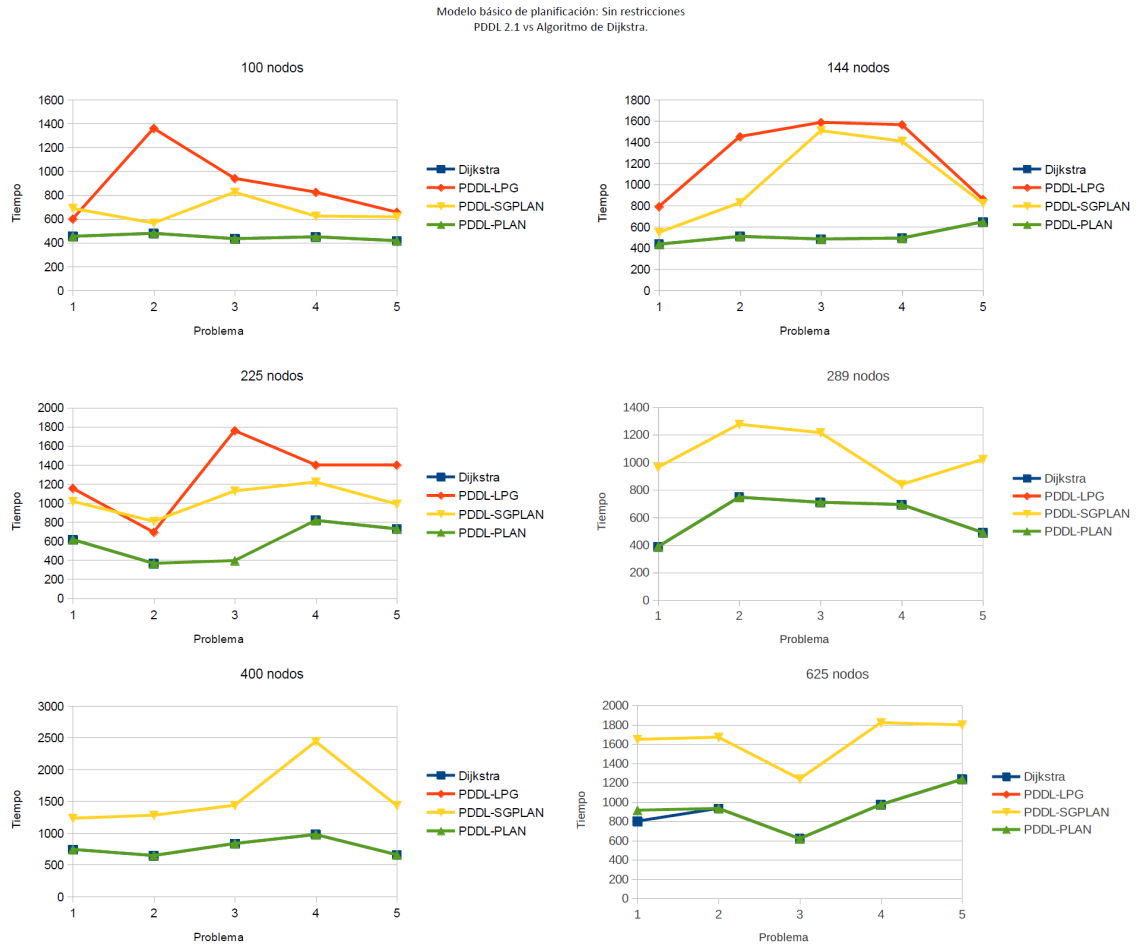


Figura 4.8: Tiempo de duración del plan para el modelo de planificación correspondiente a la preferencia de la longitud a caminar a la parada de autobuses.

este planificador es bastante competitivo. LPG presenta menor calidad en los planes arrojados.

Modelo básico: Tiempo de recorrido - distancia máxima a caminar durante todo el plan

Anteriormente se mencionaron las diversas preferencias que tiene el usuario al momento de generar un plan de viaje usando el transporte público, una de estas preferencias es la longitud máxima a caminar durante todo el plan, para ello partiendo del modelo de la figura 4.1, se agregan las functions *(dist_maxima)* y *(dist_max_cam)*

```
(:functions
        (longitud_cam ?pas ?de ?a)
        (tiempo_b ?pas ?bus ?de ?a)
        (aux_1)
        (dist_maxima)
        (dist_max_c)
)
```

Figura 4.9: Functions del modelo básico donde se limita la distancia máxima a caminar durante todo el plan.

```
(:durative-action Caminar
        :parameters(?pas ?de ?a)
        :duration(=?duration (/(longitud_cam ?pas ?de ?a)1.39))
        :condition(and
                (at start(<(dist_max_c)(dist_maxima)))
                (at start(pasajero ?pas))
                (at start(at ?pas ?de))
                (at start(ir_cam ?pas ?de ?a))
                )
        :effect(and
                (at start(not(at ?pas ?de)))
                (at end(at ?pas ?a))
                (at end(increase(dist_max_c)(longitud_cam ?pas ?de ?a)))
                )
)
```

Figura 4.10: Acción Caminar del modelo básico donde se limita la distancia máxima a caminar durante todo el plan, donde se ven las functions participantes para esta preferencia del usuario.

figura 4.9, las cuales se describieron anteriormente. Su implementación es llevada a cabo en la acción *Caminar*, esto se puede ver en la figura 4.10. Donde se indica que el usuario debe caminar menos de *(dist_maxima)*, siendo la function *(dist_max_c)* quien almacena la longitud recorrida caminando.

En la Figura 4.11 y Tabla 4.7 se puede ver los tiempos de ejecución del planificador para encontrar un plan para este modelo de planificación básico. Se puede observar que algunos problemas el planificador LPG no pudo encontrar una solución dentro del tiempo especificado (una hora).

Se puede ver que LPG aparte de no dar solución a algunos problemas, para la red de 144 y 225 nodos presenta tiempos de ejecución muy altos, ver figura 4.11. Sin embargo SGPLAN presenta tiempos pequeños muy similares e inclusive menores que

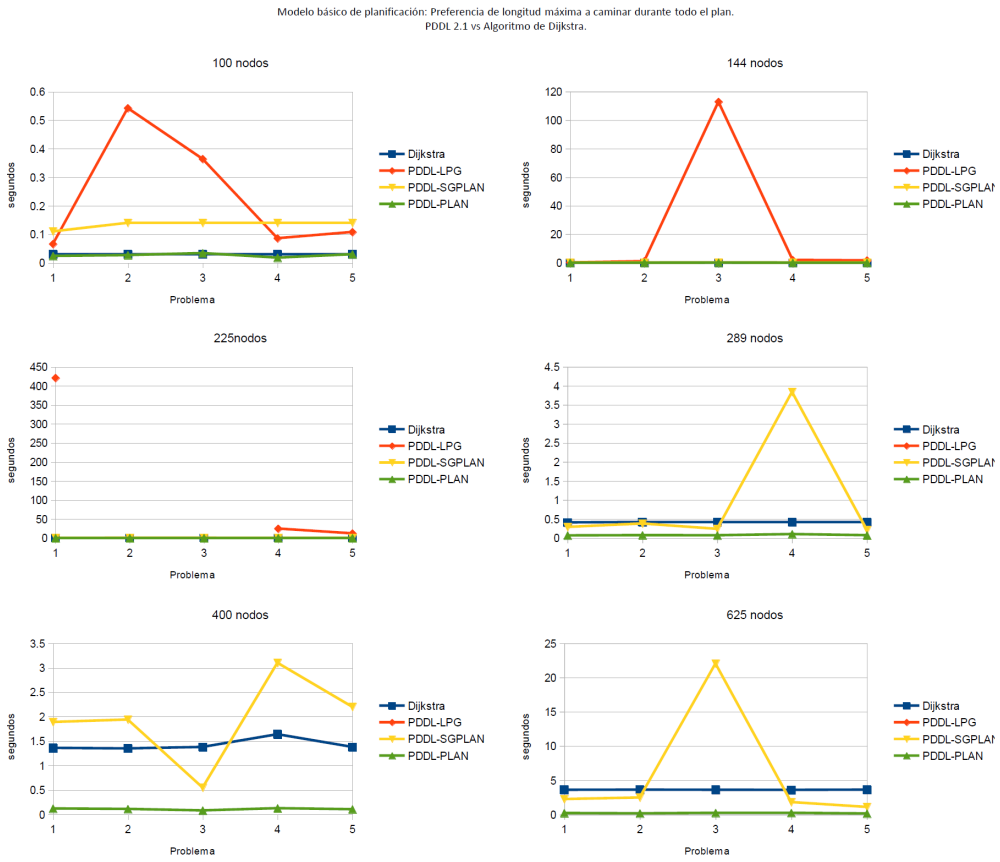


Figura 4.11: Tiempos de ejecución obtenidos de un modelo básico con restricción de longitud máxima a caminar durante todo el plan.

# Nodos	Archivo	Dijkstra	LPG-TD	SGPLAN	PLAN
100	1	0.03	0.066	0.11	0.024
	2	0.03	0.542	0.14	0.027
	2	0.03	0.364	0.14	0.034
	2	0.03	0.086	0.14	0.018
	4	0.03	0.108	0.14	0.03
144	1	0.07	0.214	0.15	0.031
	2	0.07	1.156	0.15	0.033
	2	0.07	112.81	0.22	0.091
	2	0.07	1.95	0.16	0.035
	4	0.07	1.696	0.16	0.042
225	1	0.27	420.638	0.17	0.064
	2	0.26	—	0.54	0.074
	2	0.26	—	0.48	0.062
	2	0.27	24.77	0.23	0.053
	4	0.27	12.13	0.18	0.055
289	1	0.41	—	0.29	0.069
	2	0.42	—	0.38	0.075
	2	0.42	—	0.24	0.071
	2	0.42	—	3.84	0.103
	4	0.42	—	0.21	0.073
400	1	1.36	—	1.89	0.122
	2	1.35	—	1.94	0.113
	2	1.38	—	0.55	0.083
	2	1.64	—	3.1	0.128
	4	1.36	—	2.2	0.107
625	1	3.6	—	2.25	0.209
	2	3.62	—	2.48	0.169
	2	3.6	—	22.04	0.234
	2	3.59	—	1.81	0.237
	4	3.62	—	1.1	0.152

Tabla 4.7: Tiempo de ejecución en segundos para la obtención de un plan en el modelo básico de planificación con restricción de distancia máxima a caminar durante todo el plan.

los obtenidos por el Algoritmo de Dijkstra, salvo por el problema tres para una red de 625 nodos, aunque es un poco elevado aún es razonable este tiempo de respuesta. Por otro lado en esta misma figura se muestra que PLAN reporta tiempos muy pequeños de nueva cuenta.

En el tiempo de duración del plan de viaje se obtuvieron los resultados mostrados en la Figura 4.12 y en la Tabla 4.8. Donde se puede ver que el planificador PLAN en todos los problemas encuentra el valor óptimo. SGPLAN entrega tiempos de duración del plan bastante competitivos en estos experimentos.

LPG entrega resultados similares a SGPLAN aunque no pudo resolver todos los problemas que se le presentaron y en base a las gráficas y resultados mostrados se puede ver que existe una solución. Además de lo anterior, para este modelo básico LPG no tiene la capacidad de trabajar con redes mayores de 225 nodos, lo cual lo hace un poco incompetente, ya que por el tipo de error que arroja nos da a indicar que se trata de un desborde de la capacidad de memoria del planificador.

Modelo básico: Tiempo de recorrido - Preferencias de longitud a caminar a la parada y durante todo el plan

En esta parte de la experimentación decidimos utilizar las preferencias anteriormente mencionadas en el modelo de planificación básico, para tener un mayor control de la longitud a caminar por parte del usuario, es decir, si solo utilizamos la preferencia que limita la longitud para alcanzar una parada de autobuses o el objetivo, se podría caer en el caso de hacer varias transferencias durante el plan, lo cual da a posibilidad a recorrer una longitud muy grande caminando.

Por otro lado, si solo usamos la preferencia de la longitud a caminar durante todo el plan de viaje, cabe la posibilidad que nos mande a caminar toda esa distancia de forma continua. Por ello se implementa un modelo de planificación donde se utilicen ambas preferencias para contrarrestar lo anteriormente mencionado.

Esto nos llevó a la modificación del dominio de planificación mostrado en el

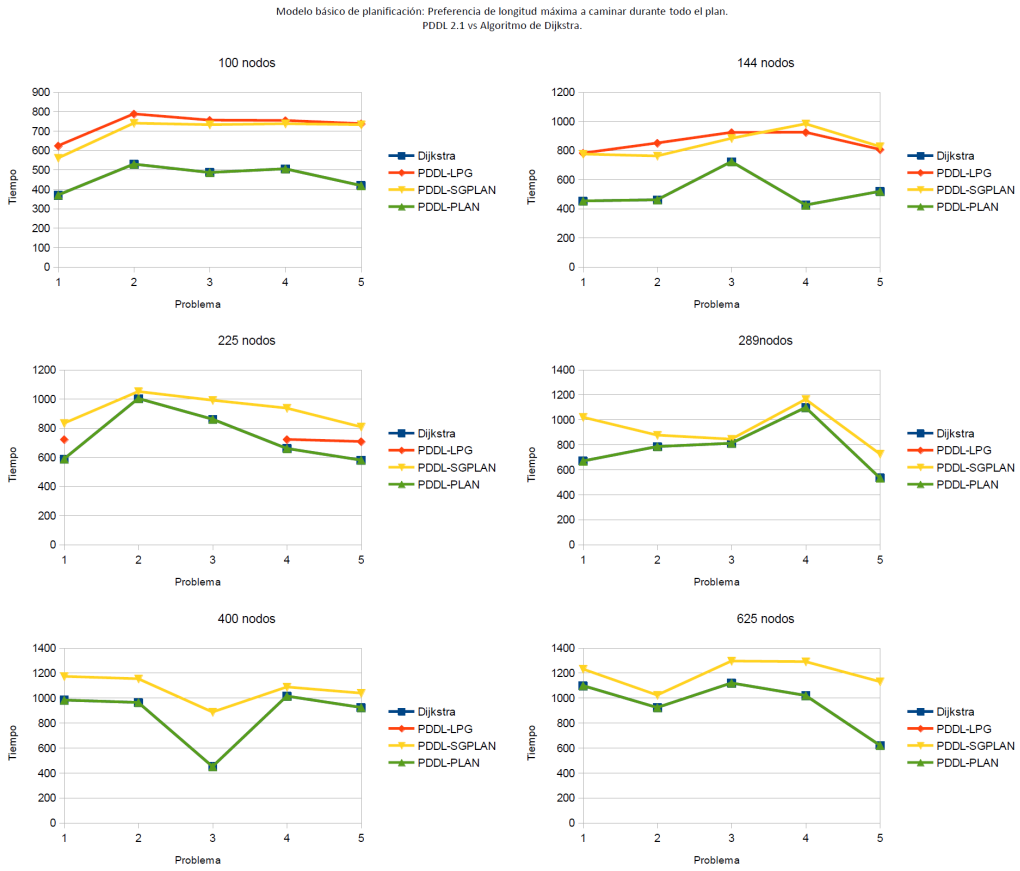


Figura 4.12: Tiempo de duración del plan obtenido de un modelo básico con restricción de longitud máxima a caminar durante todo el plan.

# Nodos	Archivo	Dijkstra	LPG-TD	SGPLAN	PLAN
100	1	369	622.6	559	369
	2	528	786.8	739	528
	2	485	754.6	731	485
	2	504	753	736	504
	4	418	736.6	731	418
144	1	451	779.8	773	451
	2	459	849	760	459
	2	721	922	881	721
	2	423	922.6	981	423
	4	517	804.8	825	517
225	1	586	720	831	586
	2	1001	—	1049	1001
	2	859	—	989	859
	2	658	720	935	658
	4	578	705.6	806	578
289	1	667	—	1018	667
	2	783	—	874	783
	2	809	—	842	809
	2	1096	—	1161	1096
	4	532	—	724	532
400	1	981	—	1171	981
	2	962	—	1152	962
	2	449	—	885	449
	2	1013	—	1086	1013
	4	922	—	1037	922
625	1	1097	—	1230	1097
	2	921	—	1021	921
	2	1118	—	1294	1118
	2	1018	—	1289	1018
	4	619	—	1129	619

Tabla 4.8: Tiempo de duración del plan obtenido de un modelo básico con restricción de longitud máxima a caminar en todo el plan de viaje.

modelo básico: Tiempo de recorrido mostrado en la figura 4.1 donde se implementan estas preferencias pertenecientes a la acción *Caminar*, quedando tal como se muestra

```
(:durative-action Caminar
        :parameters(?pas ?de ?a)
        :duration(=?duration (/(longitud_cam ?pas ?de ?a)1.39))
        :condition(and
                (at start(<(longitud_c)(long_max_cam)))
                (at start(<(dist_max_c)(dist_maxima)))
                (at start(pasajero ?pas))
                (at start(at ?pas ?de))
                (at start(ir_cam ?pas ?de ?a))
                )
        :effect(and
                (at start(not(at ?pas ?de)))
                (at end(at ?pas ?a))
                (at end(increase (longitud_c)(longitud_cam ?pas ?de ?a)))
                (at end(increase(dist_max_c)(longitud_cam ?pas ?de ?a)))
                )
)
```

Figura 4.13: Acción Caminar para el modelo básico con restricción de longitud máxima a caminar.

en la figura 4.13. Pero como se mencionó antes se debe decrementar la function acumuladora de la distancia a caminar entre transbordo, llevándose a cabo en la acción *Desplazar_bus_pas*, tal como se puede ver en la figura 4.14.

Una vez realizada la experimentación para este modelo de planificación se puede observar en las gráficas de la Tabla 4.9 y en Figura 4.15 que los tiempos de ejecución son muy buenos y competitivos sobre el algoritmo implementado para comparación y del posible tiempo de espera por parte del usuario en obtener el plan.

El planificador PLAN muestra resultados óptimos de nueva cuenta en esta experimentación, mientras tanto el planificador SGPLAN presenta resultados para la función a optimizar bastante competitivos. Tal como se puede ver en la Tabla 4.10 y en las gráficas de la Figura 4.16. LPG muestra resultados muy buenos para la métrica a optimizar aunque en su mayoría superiores a los arrojados por SGPLAN.

Conclusiones del modelo básico de planificación

Como bien se observó en los resultados presentados para el modelo básico en sus diversas variantes, el planificador LPG no pudo resolver cuatro problemas dentro del tiempo limitado de solución, aparte de eso que no brindo capacidad para todos

```
(:durative-action Desplazar_bus_pas
        :parameters(?pas ?bus ?de ?a)
        :duration(=?duration (tiempo_b ?pas ?bus ?de ?a))
        :condition(and
                (at start(pasajero ?pas))
                (at start(autobus ?bus))
                (at start(at ?pas ?de))
                (at start(via_bus ?pas ?bus ?de ?a))
                )
        :effect(and
                (at start(not(at ?pas ?de)))
                (at end(at ?pas ?a))
                (at end(decrease(longitud_c)(longitud_c)))
                )
)
```

Figura 4.14: Acción Desplazar_bus_cam para el modelo básico con restricción de longitud máxima a caminar.

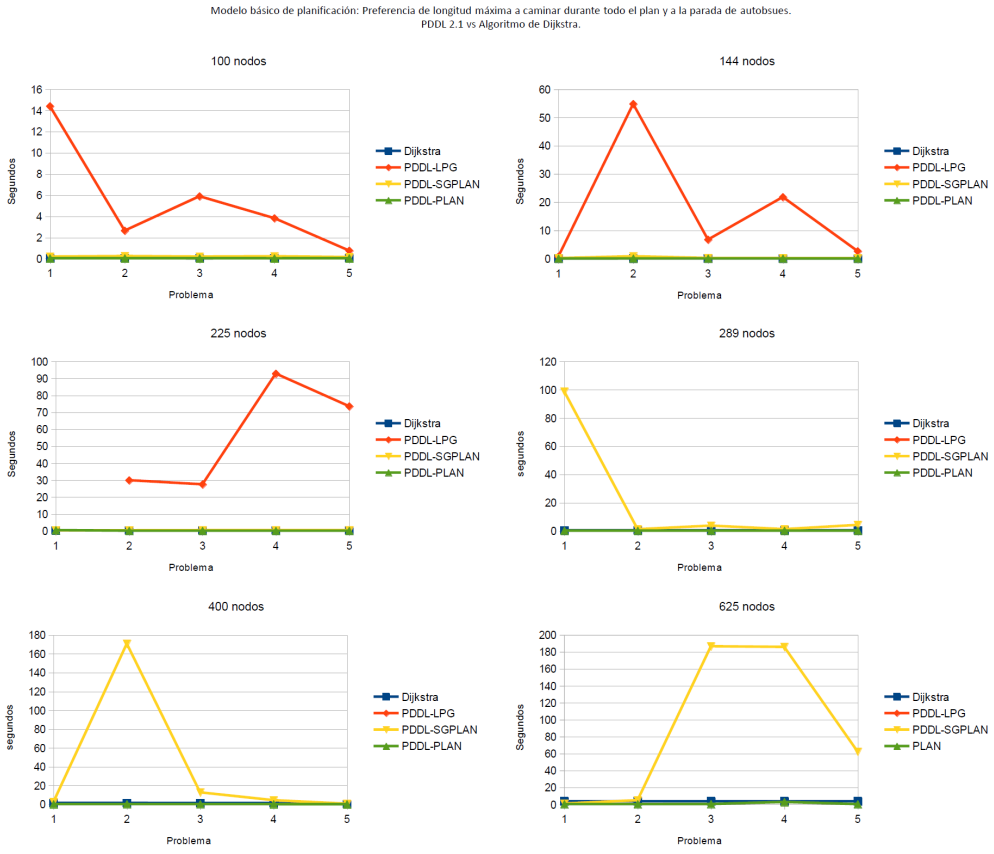


Figura 4.15: Tiempos de ejecución obtenidos para el modelo básico con restricción de longitud máxima a caminar.

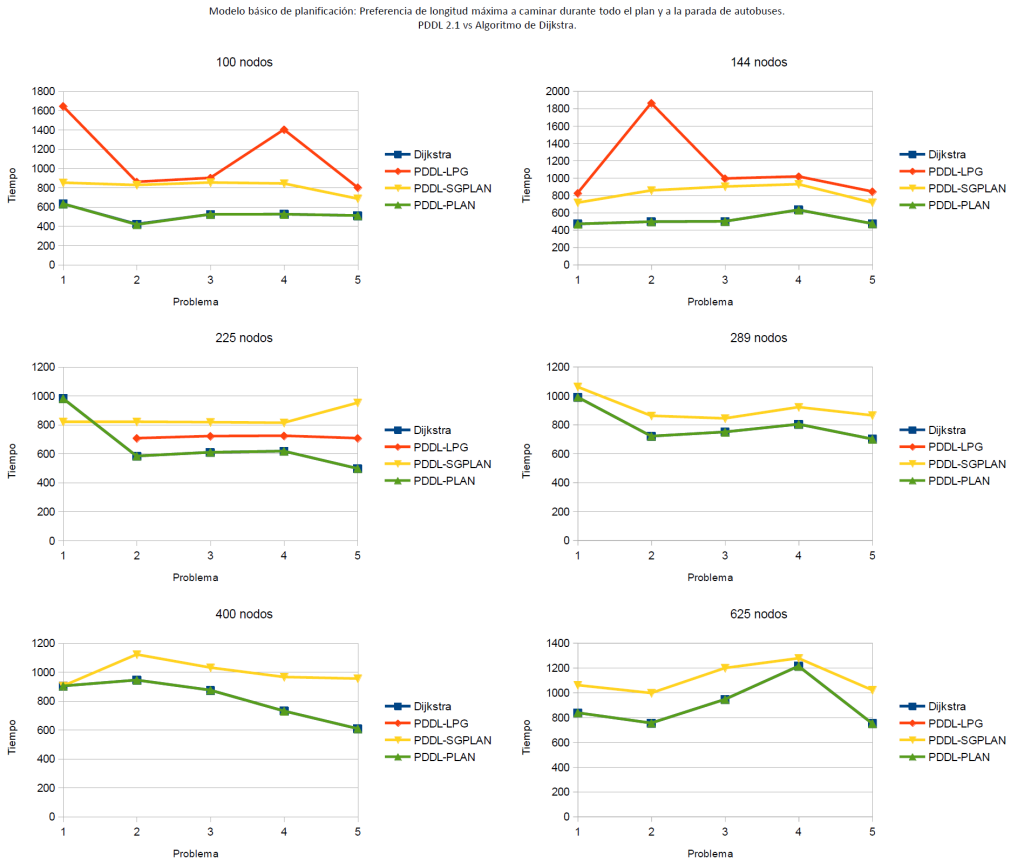


Figura 4.16: Tiempos de duración del plan obtenidos para el modelo básico con restricciónes de longitud máxima a caminar.

# Nodos	Archivo	Dijkstra	LPG-TD	SGPLAN	PLAN
100	1	0.03	14.398	0.19	0.035
	2	0.03	2.64	0.24	0.032
	3	0.02	5.886	0.19	0.034
	4	0.03	3.816	0.22	0.029
	5	0.03	0.752	0.14	0.034
144	1	0.07	0.748	0.16	0.044
	2	0.07	54.778	0.84	0.048
	3	0.07	6.72	0.22	0.06
	4	0.07	21.81	0.16	0.047
	5	0.07	2.64	0.16	0.055
225	1	0.25	—	0.33	0.429
	2	0.27	29.878	0.33	0.077
	3	0.27	27.47	0.42	0.076
	4	0.27	92.83	0.42	0.079
	5	0.27	73.556	0.42	0.078
289	1	0.42	—	98.68	0.127
	2	0.42	—	1.14	0.122
	3	0.42	—	3.69	0.093
	4	0.42	—	1.19	0.091
	5	0.42	—	4.19	0.114
400	1	1.37	—	2.48	0.141
	2	1.37	—	170.66	0.199
	3	1.36	—	12.7	0.174
	4	1.36	—	4.37	0.16
	5	1.38	—	0.63	0.16
625	1	3.65	—	0.79	0.252
	2	3.67	—	4.84	0.226
	3	3.65	—	186.59	0.343
	4	3.66	—	185.81	2.76
	5	3.66	—	62.2	0.242

Tabla 4.9: Tiempo de obtención de un plan para el modelo básico con restricción de longitud a caminar máxima y de distancia a la parada.

aquellos con más de 225 nodos, debido aun fallo en la memoria del planificador. Los demás planificadores terminaron sin problemas la tarea encomendada teniendo

# Nodos	Archivo	Dijkstra	LPG-TD	SGPLAN	PLAN
100	1	630	1643	849	630
	2	420	857.8	825	414
	3	521	899	851	521
	4	522	1400.2	841	522
	5	508	799.6	683	508
144	1	468	822.6	713	468
	2	494	1861	854	494
	3	498	991.8	899	498
	4	630	1015.4	926	630
	5	472	841.8	714	472
225	1	980	—	819	980
	2	582	705.6	819	582
	3	608	720	816	608
	4	616	722.2	812	616
	5	496	705.6	950	496
289	1	989	—	1060	989
	2	719	—	860	719
	3	748	—	842	748
	4	802	—	920	802
	5	699	—	863	699
400	1	902	—	902	902
	2	943	—	1120	943
	3	873	—	1029	873
	4	729	—	964	729
	5	607	—	953	607
625	1	836	—	1059	836
	2	753	—	995	753
	3	944	—	1197	944
	4	1212	—	1276	1212
	5	750	—	1019	750

Tabla 4.10: Duración de un plan para el modelo básico con restricción de longitud a caminar máxima y de distancia a la parada.

mejores tiempos que el Dijkstra para la mayoría de los casos presentados.

En la mayor parte de los tiempos de ejecución de los planificadores mostrados

en las figuras 4.2, 4.7, 4.11 y 4.15 se puede observar que son bastante competitivos entre ellos y con lo que el usuario esperaría, ya que este último requiere de una solución instantánea a su consulta.

Para los valores obtenidos de los tiempos de duración del plan en las figuras 4.3, 4.8, 4.12 y 4.16 correspondientes a estos resultados se observa que los tiempos de duración del plan son muy competitivos entre todos los planificadores con respecto al método Dijkstra, alcanzando en el caso de PLAN los valores óptimos en todos los casos presentados. Como caso general tenemos a SGPLAN, el cual presenta tiempos de ejecución muy pequeños en la mayoría de los casos, y su valor objetivo está en la mayor parte de las veces muy cercano al óptimo, lo cual lo hace un planificador muy competente al menos para estos experimentos.

Por lo tanto se concluye para esta primera sección de resultados que nuestro modelo básico de planificación es muy competitivo o mejores a los resultados obtenidos por el Algoritmo Dijkstra.

4.2 Modelo de planificación completo para la generación de planes de viaje para una red de transporte público

En esta parte de la experimentación se llevaron a cabo corridas para la generación de planes utilizando el modelo completo de planificación para nuestro problema (figuras 3.16 y 3.17), donde solo para recordar, se dijo anteriormente que este modelo contiene todas las acciones expuestas con anterioridad así como las preferencias de los usuarios.

Esto se hace con la finalidad de analizar si los planificadores actuales y de los cuales hacemos uso pueden soportar este modelo de planificación en base a a los resultados obtenidos, buscando de este modo tener pautas para mejoras de los

```
(:durative-action Caminar
        :parameters(?pas ?de ?a)
        :duration(=?duration (/(longitud_cam ?pas ?de ?a)1.39))
        :condition(and
            (at start(<=(longitud_c)(long_max_cam)))
            (at start(pasajero ?pas))
            (at start(ir_cam ?pas ?de ?a))
            (at start(at ?pas ?de))

            )
        :effect(and
            (at start(not(at ?pas ?de)))
            (at end(not(ir_cam ?pas ?de ?a)))
            (at end(at ?pas ?a))
            (at end(increase(longitud_c)(longitud_cam ?pas ?de ?a)))
            )
    )
```

Figura 4.17: Acción caminar modificada para la eliminación de ciclos en el plan arrojado al usuario.

mismos. En esta sección no se muestran resultados para el planificador PLAN, ya que no soporto nuestro modelo de planifiación.

4.2.1 Modelo de planificación completo: Métrica a optimizar Tiempo

Siguiendo con lo presentado anteriormente presentaremos los resultados obtenidos para cuando se desea optimizar el tiempo de recorrido de viaje, solo que ahora no será para un modelo específico para esa métrica, si no que en este se soporta todas las preferencias anteriormente mencionadas, ya sea como preferencias (restricciones) o como métrica a optimizar.

Resultados obtenidos para un modelo sin restricciones

Siguiendo con la presentación de los resultados obtenidos, en esta subsección mostramos de forma gráfica lo obtenido para el modelo de planificación completo, donde el usuario no ingresa preferencias que restrinjan el plan a generar. Para evitar visitar dos veces un mismo nodo, la acción *Caminar* queda definida como se muestra

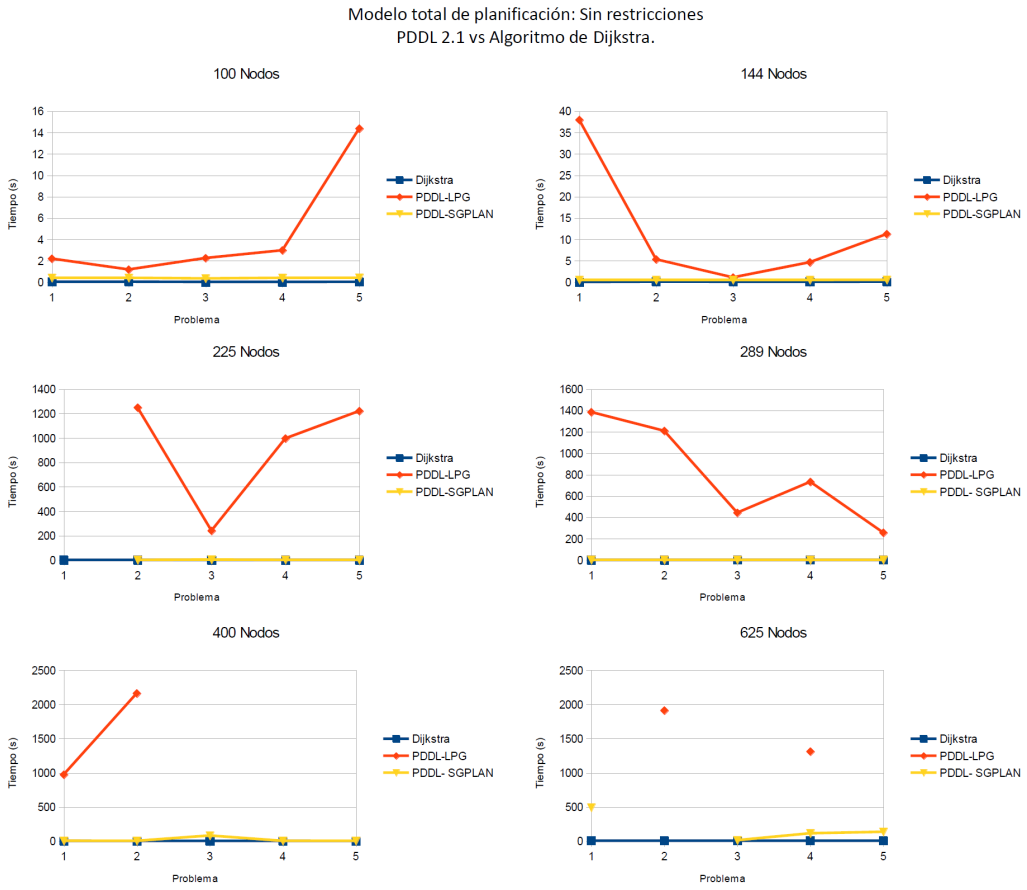


Figura 4.18: Tiempo de ejecución para la obtención de un plan en un modelo total sin restricciones.

en la figura 4.17.

Los tiempos de ejecución del planificador para arrojar una solución para esta tanda de experimentos se muestran los resultados en la Tabla 4.11 y en la Figura 4.18. Donde se observa que son bastante competitivos, pero irregulares entre cada experimento, esto último para el planificador LPG. SGPLAN tal como se ha mostrado en toda la experimentación presentada mantiene los mejores tiempos entre estos recursos que estamos utilizando para nuestras pruebas.

Por otro lado tenemos que la calidad del plan dada por el tiempo de duración del viaje, en la mayor parte de los resultados obtenidos por los planes generados son de muy buena calidad salvo los arrojados por el planificador LPG para los problemas

# Nodos	Archivo	Dijkstra	LPG-TD	SGPLAN
100	1	0.04	2.204	0.4
	2	0.04	1.184	0.4
	3	0.02	2.254	0.35
	4	0.02	2.98	0.4
	5	0.03	14.376	0.41
144	1	0.03	37.95	0.52
	2	0.12	5.342	0.52
	3	0.09	1.104	0.51
	4	0.07	4.662	0.51
	5	0.11	11.252	0.54
225	1	0.31	—	—
	2	0.29	1247.746	1.01
	3	0.44	240.062	3.4
	4	0.3	995.03	1.19
	5	0.34	1219.596	1.04
289	1	0.53	1383.996	1.58
	2	0.58	1208.218	1.54
	3	0.61	442.614	1.49
	4	0.64	731.57	1.48
	5	0.83	255.914	1.58
400	1	2.18	972.05	4.21
	2	1.52	2159.43	3.31
	3	1.79	—-	80.55
	4	1.91	—-	3.2
	5	1.68	—	3.36
625	1	4.79	—	489.48
	2	5.19	1909.865	—
	3	4.61	—	12.53
	4	4.99	1309.185	113.78
	5	4.4	—	135.08

Tabla 4.11: Tiempo de ejecución de los planificadores para la obtención de un plan para el modelo completo sin restricciones.

cuatro y dos de las gráficas de 225 y 289 (también se puede ver en la Tabla 4.12 nodos respectivamente. Se puede observar que los resultados obtenidos por SGPLAN son

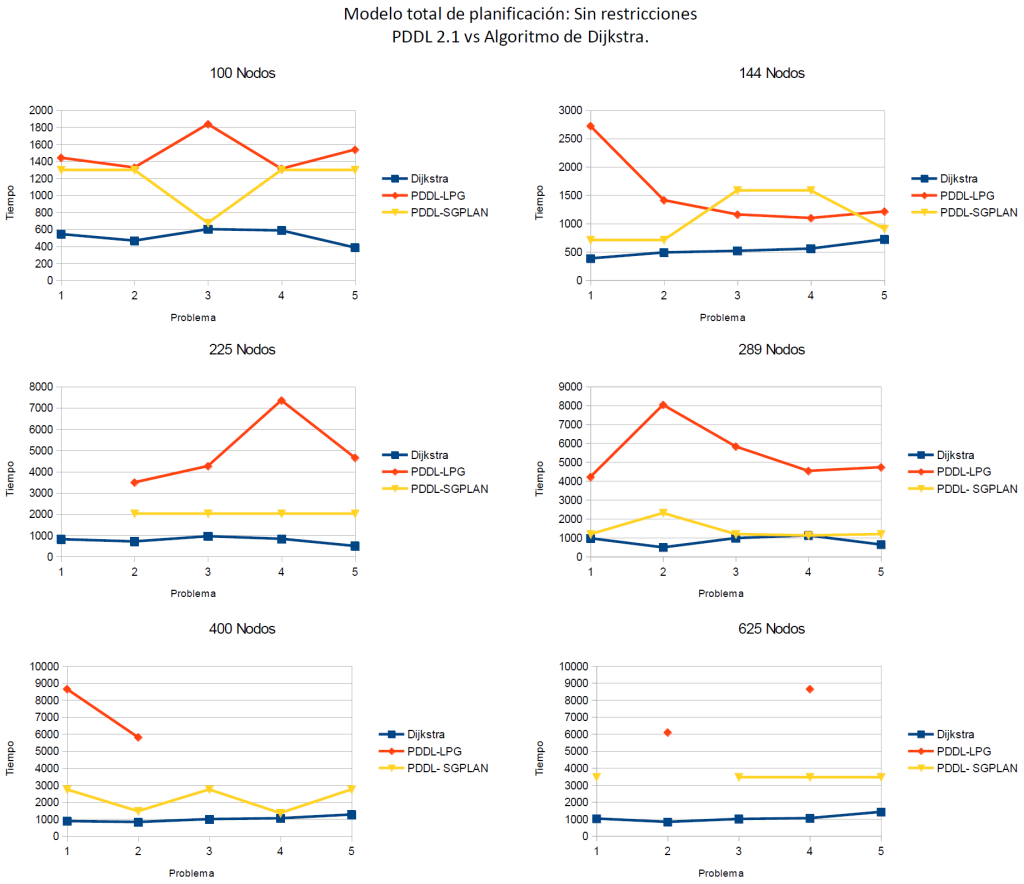


Figura 4.19: Tiempo necesario por el usuario para poder llevar a cabo un plan obtenido del modelo total sin restricciones.

los más cercanos al óptimo en cada uno de los problemas resueltos. Para constatar esto ver la figura 4.19.

Debido a lo mostrado anteriormente se tiene que al menos para esta sección de resultados el planificador SGPLAN es muy superior al planificador LPG, ya que las gráficas muestran que este planificador tiene el menor tiempo de ejecución y los valores más cercanos al óptimo para el tiempo de duración del viaje en la mayoría de los problemas resueltos, considerando también que resolvió la mayoría de los casos propuestos para ser exactos 28/30, mientras que LPG solo soluciono 23/30.

# Nodos	Archivo	Dijkstra	LPG-TD	SGPLAN
100	1	542	1439.6	1296
	2	465	1324.8	1296
	3	600	1835.4	672
	4	585	1310.4	1296
	5	384	1535.2	1296
144	1	384	2721	709
	2	489	1412	709
	3	517	1158.8	1584
	4	557	1096.8	1584
	5	720	1212.2	907
225	1	812	—	—
	2	706	3486	2016
	3	951	4253.8	2016
	4	830	7344.33	2016
	5	495	4648.2	2016
289	1	963	4209.6	1182
	2	482	8028.4	2304
	3	972	5823.2	1182
	4	1117	4530.75	1117
	5	630	4723.2	1182
400	1	872	8640	2736
	2	819	5804.5	1458
	3	991	—	2736
	4	1040	—	1345
	5	1260	—	2736
625	1	1021	—	3456
	2	826	6091	—
	3	998	—	3456
	4	1047	8640	3456
	5	1415	—	3456

Tabla 4.12: Tiempo de duración de los planes de viaje obtenidos del modelo completo sin restricciones.

Resultados obtenidos para un modelo con restricción de longitud caminada a la parada de autobuses o al objetivo

En esta parte como bien se ha explicado anteriormente se utiliza la métrica que restringe la distancia a caminar por parte del usuario para alcanzar una parada de autobuses o al objetivo. Dado los resultados obtenidos y tras varias pruebas se observó que *(at end(not(ir_cam ?pas ?de ?a)))* en la acción *Caminar* tal como se muestra en la figura 4.17, incrementaba los tiempos computacionales de solución cuando se computaba un plan con métricas de distancia a caminar, por ello se elimina en esta sección esa línea en el archivo dominio, es de allí él porqué del dominio de la figura 3.16, esto sucede para el planifiacador LPG debido a su tipo de búsqueda local, donde creemos que cae en óptimos locales y hace ciclos en ellos. Este cambio mejoró significativamente los tiempos obtenidos y que se mostraron anteriormente, ya que los archivos problemas son similares.

En las figuras 4.20 y 4.21, así como en las Tablas 4.13 y 4.14 se tiene que en total se resolvieron 19 de los 30 problemas correspondientes a esta sección, de los cuales LPG resolvió 18 de estos problemas y SGPLAN solo 11. En la Figura 4.20 y en la Tabla 4.13, se muestra los tiempos de ejecución de los planificadores y del algoritmo de Dijkstra para la obtención de un plan, donde la mayor parte de los tiempos de resolución son muy competitivos, aunque SGPLAN tiene los mejores tiempos los cuales son muy similares a los arrojados por el algoritmo exacto.

Los resultados obtenidos para la métrica a optimizar se observan en la Tabla 4.14 y en la Figura 4.21 donde se tiene que el planificador LPG arroja los peores resultados, siendo estos moderadamanete altos, mientras que SGPLAN arroja resultados más cercanos al óptimo, pero este último resolvio el menor número de archivos problema.

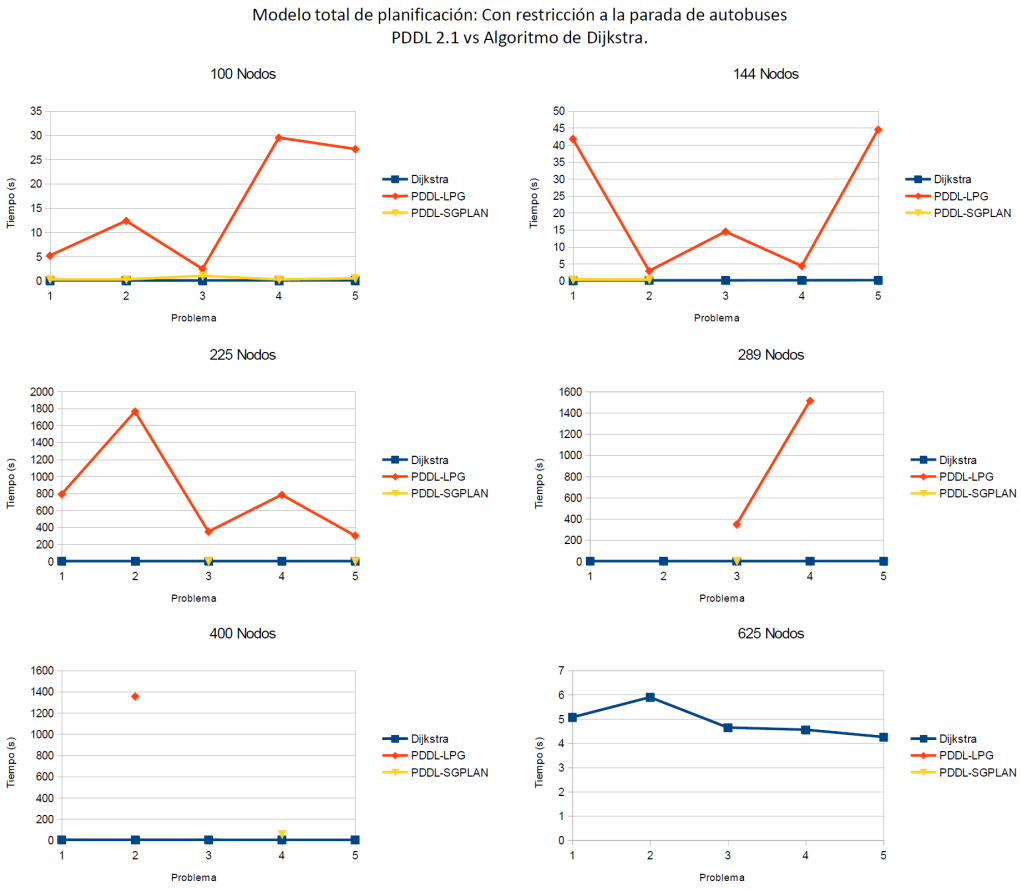


Figura 4.20: Tiempo de ejecución de los planificadores para la obtención de un plan de viaje.

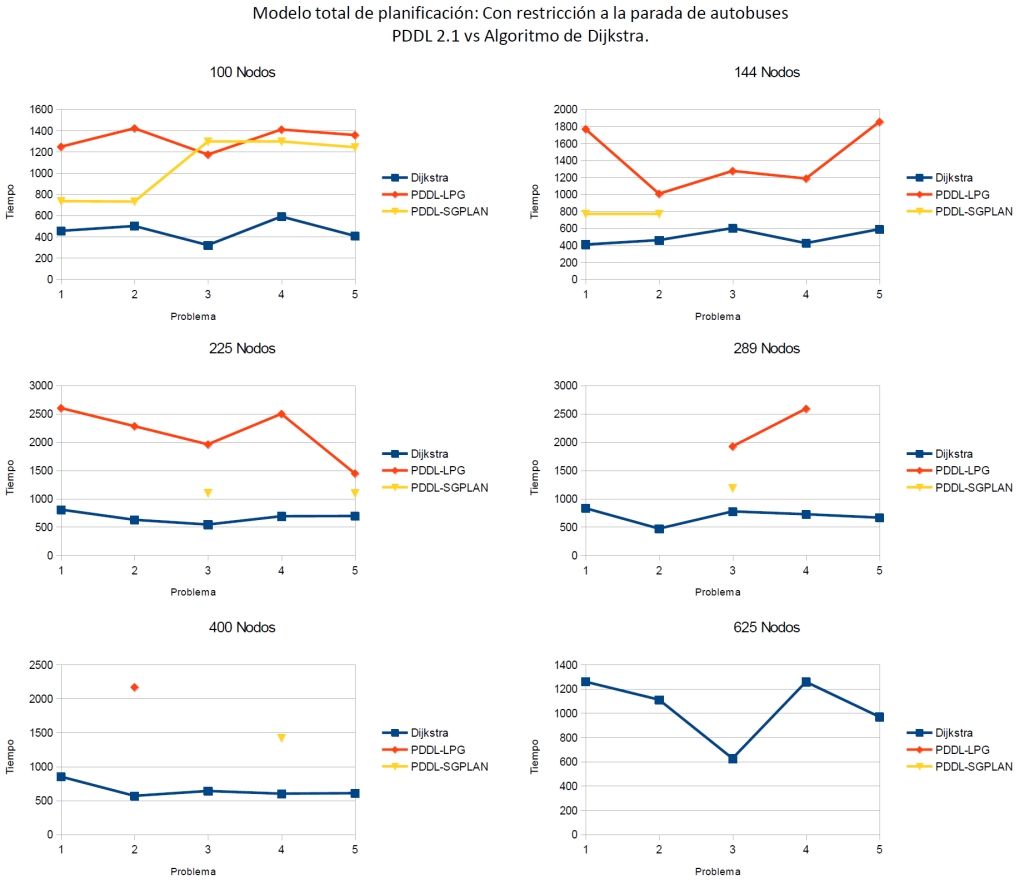


Figura 4.21: Calidad del plan de viaje.

# Nodos	Archivo	Dijkstra	LPG-TD	SGPLAN
100	1	0.04	5.144	0.26
	2	0.03	12.33	0.27
	3	0.03	2.408	0.98
	4	0.04	29.478	0.25
	5	0.04	27.124	0.47
144	1	0.04	41.736	0.31
	2	0.07	2.852	0.31
	3	0.07	14.436	—
	4	0.11	4.322	—
	5	0.13	44.486	—
225	1	0.29	789.418	—
	2	0.45	1763.1825	—
	3	0.38	348.05	0.56
	4	0.53	782.15	—
	5	0.4	299.92	0.51
289	1	0.7	—	—
	2	0.83	—	—
	3	0.63	347.328	2.06
	4	0.77	1511.735	—
	5	0.91	—	—
400	1	2.64	1354.33	—
	2	2.22	1354.33	—
	3	2.41	—	—
	4	1.91	—	54.87
	5	1.91	—	—
625	1	5.06	—	—
	2	5.89	—	—
	3	4.64	—	—
	4	4.55	—	—
	5	4.25	—	—

Tabla 4.13: Tiempo de ejecución de los planificadores para la obtención de un plan correspondientes al modelo completo con restricción de distancia a caminar a la parada.

# Nodos	Archivo	Dijkstra	LPG-TD	SGPLAN
100	1	454	1246.4	734
	2	500	1420	730
	3	320	1172.2	1296
	4	589	1408.4	1296
	5	407	1356.4	1242
144	1	407	1765.6	768
	2	459	1004.4	768
	3	600	1273.2	—
	4	423	1183.4	—
	5	588	1851.2	—
225	1	803	2596.8	—
	2	624	2277	—
	3	540	1955.4	1096
	4	687	2494	—
	5	691	1442.8	1094
289	1	828	—	—
	2	468	—	—
	3	772	1919	1182
	4	722	2585	—
	5	662	—	—
400	1	848	—	—
	2	566	2165	—
	3	637	—	—
	4	598	—	1416
	5	606	—	—
625	1	1257	—	—
	2	1109	—	—
	3	625	—	—
	4	1256	—	—
	5	968	—	—

Tabla 4.14: Tiempo de duración del plan correspondientes al modelo completo con restricción de distancia a caminar a la parada.

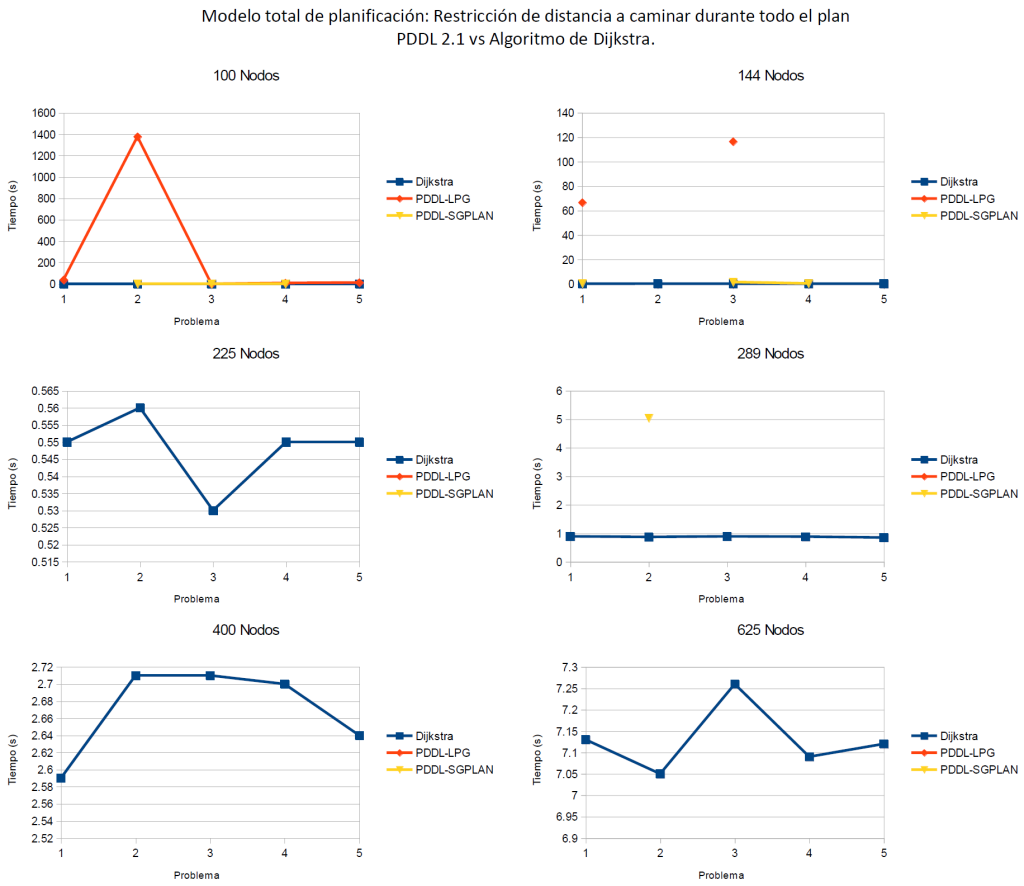


Figura 4.22: Tiempo de ejecución de los planificadores para la obtención de un plan de viaje.

Resultados obtenidos para un modelo con restricción de longitud caminada durante todo el plan de viaje

En estos resultados para la preferencia de máxima distancia a caminar por parte del usuario dentro del plan de viaje a computar, se tiene que solo se resolvieron 9 problemas de los 30 presentados, donde ambos planificadores pudieron resolver siete cada uno.

Los tiempos de solución son mostrados en la Tabla 4.15 y gráficamente en la Figura 4.22 donde se tiene que salvo para el problema 2 correspondiente a la gráfica de 100 nodos resuelto por el planificador LPG los tiempos presentados son muy bajos.

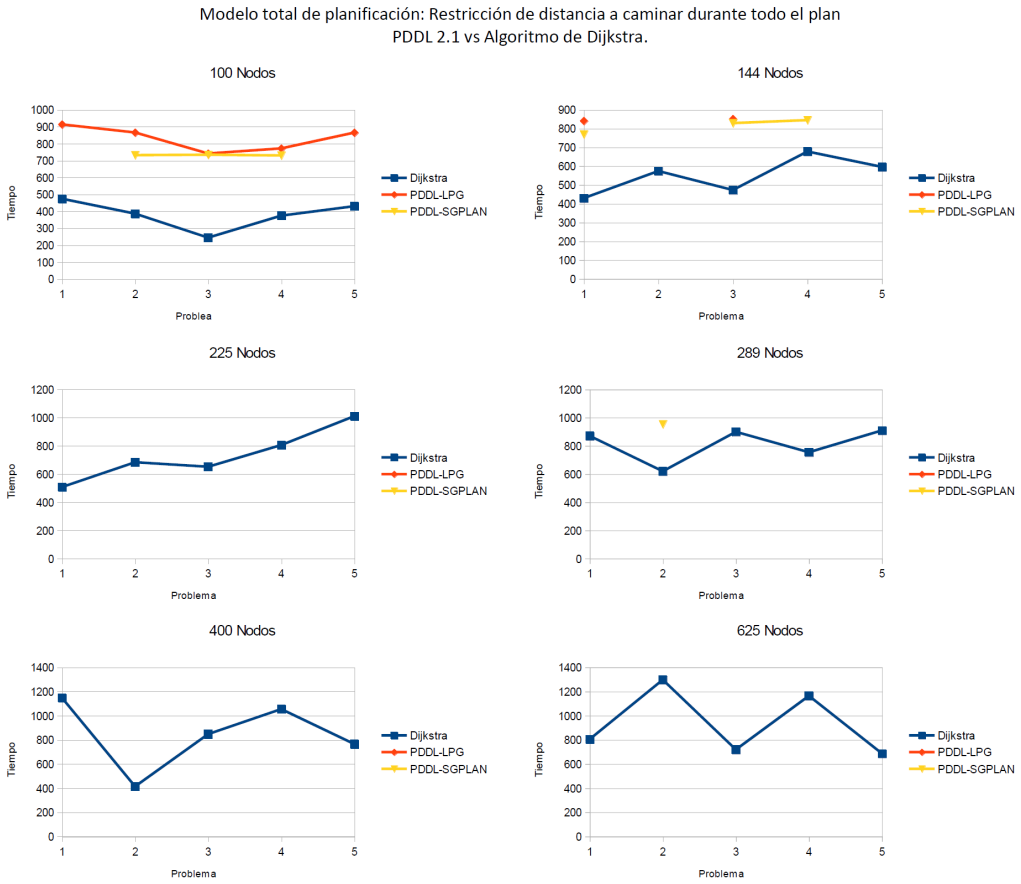


Figura 4.23: Calidad del plan obtenido por los planificadores en uso.

# Nodos	Archivo	Dijkstra	LPG-TD	SGPLAN
100	1	0.08	38.104	—
	2	0.07	1376.834	0.45
	3	0.07	0.414	0.39
	4	0.07	8.702	0.4
	5	0.07	12.558	—
144	1	0.07	66.546	0.38
	2	0.12	—	—
	3	0.13	116.502	1.52
	4	0.13	—	0.35
	5	0.12	—	—
225	1	0.55	—	—
	2	0.56	—	—
	3	0.53	—	—
	4	0.55	—	—
	5	0.55	—	—
289	1	0.89	—	—
	2	0.87	—	5.04
	3	0.89	—	—
	4	0.88	—	—
	5	0.85	—	—
400	1	2.59	—	—
	2	2.71	—	—
	3	2.71	—	—
	4	2.7	—	—
	5	2.64	—	—
625	1	7.13	—	—
	2	7.05	—	—
	3	7.26	—	—
	4	7.09	—	—
	5	7.12	—	—

Tabla 4.15: Tiempo de ejecuciń de los planificadores para la obtención de un plan correspondientes al modelo completo con restricción de distancia máxima a caminar durante todo el plan.

# Nodos	Archivo	Dijkstra	LPG-TD	SGPLAN
100	1	474	913.6	—
	2	386	865.8	732
	3	244	741.6	734
	4	374	771.6	730
	5	430	865	—
144	1	430	841.2	769
	2	574	—	—
	3	473	851.4	829
	4	678	—	845
	5	596	—	—
225	1	507	—	—
	2	683	—	—
	3	651	—	—
	4	805	—	—
	5	1010	—	—
289	1	870	—	—
	2	619	—	953
	3	899	—	—
	4	754	—	—
	5	909	—	—
400	1	1147	—	—
	2	414	—	—
	3	847	—	—
	4	1054	—	—
	5	765	—	—
625	1	805	—	—
	2	1297	—	—
	3	719	—	—
	4	1164	—	—
	5	686	—	—

Tabla 4.16: Tiempo dureción de un plan correspondiente al modelo completo con restricción de distancia máxima a caminar durante todo el plan.

Para la calidad del plan se obtuvieron resultados muy buenos en ambos planificadores en comparación con lo obtenido por el algoritmo exacto, esto se puede ver

en la Tabla4.16 y los mostramos gráficamente en la figura 4.23, aunque SGPLAN fue mejor que LPG en los resultados entregados para la métrica a optimizar para los diversos problemas, aunque la diferencia es muy poca entre ambos planificadores.

Resultados obtenidos para un modelo de planificacción con combinación de restricciones de longitud

En esta última sección de resultados para el modelo de planificación total, se presentan los resultados obtenidos para cuando el usuario desea hacer una combinación de restricciones de longitud, las cuales son: longitud de recorrido a caminar a la parada o al objetivo y la de distancia máxima a caminar durante todo el plan por parte del usuario.

En la figura 4.22 y en la Tabla 4.17 se muestran los tiempos de ejecución obtenidos de espera por un plan de parte de los planificadores, de los pocos planes generados, se puede ver en esta figura 4.22 que la mayor parte de ellos se solucionaron de forma bastante rápida, excepto por un plan generado que se puede observar en la gráfica correspondiente de 100 nodos. El planificador SGPLAN muestra los tiempos de solución más pequeños, sin embargo soluciono muy pocos archivos problemas.

La calidad de los planes generados en esta parte de la experimentación es mostrada en las gráficas de la Tabla 4.18 y en la Figura 4.25 donde se tiene que estos son bastante competitivos, es decir están muy cercanos al valor óptimo en todos los planes generados. A comparación de los demás resultados mostrados para el modelo completo de planificación, no se distingue una gran diferencia entre los planificadores en base a los resultados obtenidos salvo que LPG pudo solucionar 11/30 archivos problema y SGPLAN solo 5.

Conclusiones del modelo completo de planificación

Tal como se puede ver en los resultados mostrados con anterioridad existe una diferencia importante en los tiempos de solución reportados por los planificadores

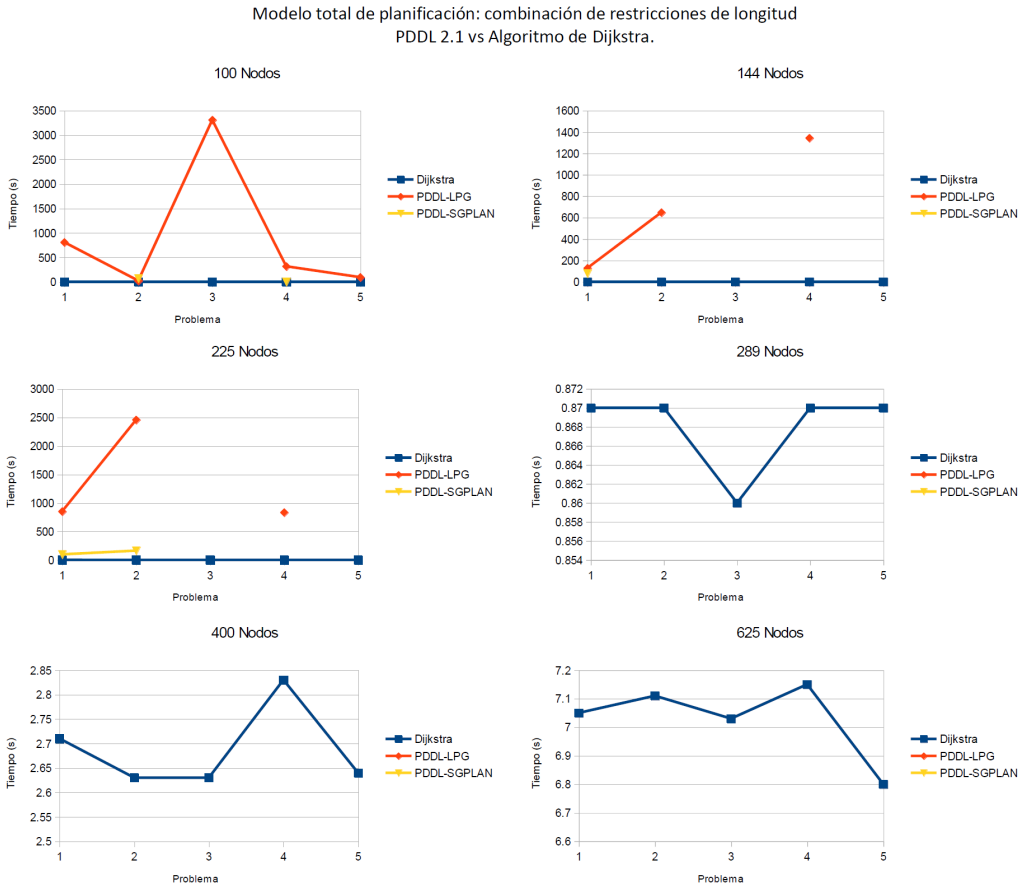


Figura 4.24: Tiempo de ejecuón para la obtención de un plan con combinación de métricas de distancia a recorrer para el modelo completo de planificación.

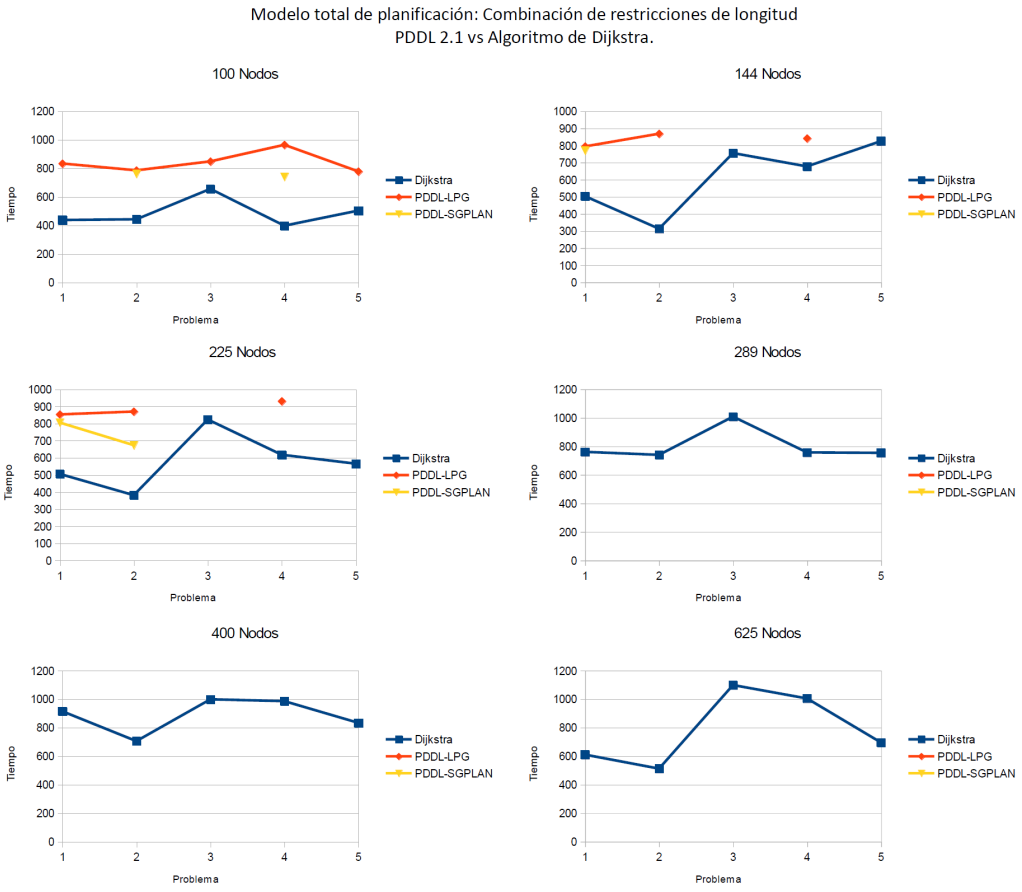


Figura 4.25: calidad de los planes obtenidos de un plan con combinación de métricas de distancia a recorrer para el modelo completo de planificación.

# Nodos	Archivo	Dijkstra	LPG-TD	SGPLAN
100	1	0.04	808.37	—
	2	0.04	22.992	70.25
	3	0.04	3309.39	—
	4	0.05	316.8	0.2
	5	0.04	93.414	—
144	1	0.04	127.955	78.41
	2	0.13	648.518	—
	3	0.13	—	—
	4	0.14	1343.09	—
	5	0.13	—	—
225	1	0.57	851.0625	97.94
	2	0.56	2455.725	162.82
	3	0.56	—	—
	4	0.55	832.9025	—
	5	0.54	—	—
289	1	0.87	—	—
	2	0.87	—	—
	3	0.86	—	—
	4	0.87	—	—
	5	0.87	—	—
400	1	2.71	—	—
	2	2.63	—	—
	3	2.63	—	—
	4	2.83	—	—
	5	2.64	—	—
625	1	7.05	—	—
	2	7.11	—	—
	3	7.03	—	—
	4	7.15	—	—
	5	6.8	—	—

Tabla 4.17: Tiempo de ejecución de los planificadores para la obtención de un plan correspondiente al modelo completo con combinación de restricciones de longitud

utilizados durante la experimentación en comparación a los reportados por el Algoritmo de Dijkstra, esto debe principalmente a:

# Nodos	Archivo	Dijkstra	LPG-TD	SGPLAN
100	1	437	833	—
	2	442	785	761
	3	655	848	—
	4	397	964	740
	5	503	777.4	—
144	1	503	794	770
	2	313	868.2	—
	3	755	—	—
	4	677	841	—
	5	825	—	—
225	1	505	852.75	805
	2	381	869.5	673
	3	823	—	—
	4	617	930.25	—
	5	565	—	—
289	1	761	—	—
	2	740	—	—
	3	1008	—	—
	4	757	—	—
	5	754	—	—
400	1	914	—	—
	2	705	—	—
	3	998	—	—
	4	985	—	—
	5	833	—	—
625	1	610	—	—
	2	512	—	—
	3	1098	—	—
	4	1004	—	—
	5	695	—	—

Tabla 4.18: Tiempo de duración de un plan correspondiente al modelo completo con combinación de restricciones de longitud

a. En Dijkstra se asume que la unidad de transporte se encuentra ubicada en la parada, en cambio en el modelo de planificación mostrado esta asunción no

es cierta, por lo tanto mediante la acción *Desplazar_bus* se simula el desplazamiento de las unidades para interceptarse con el usuario, por lo tanto nuestro modelo permite conocer la ubicación real de la unidad.

b. De todas las posibles rutas el planificador se debe decidir por aquella que resulte ser la mejor para ser abordada por el usuario.

c. El algoritmo de Dijkstra siempre trabaja sobre una sola opción mientras que en planificación se tienen varias alternativas de donde escoger, las cuales debe ir simulando internamente para poder estimar un buen plan.

d. Otro de las situaciones y quizá la de mayor importancia en el incremento de los tiempos de cómputo en planificacón es que va podando rutas, es decir, va generando un plan pero al momento de no cumplir con alguna de las restricciones tiende a eliminar lo avanzado y considera otra opción, Dijkstra solo descarta a los nodos que no puede alcanzar en base a las restricciones sin caer en la necesidad de recalcular la ruta de viaje.

Con base en lo anterior Dijkstra siempre se satisface con la primera ruta que toma, mientras que nuestro modelo considera diversas opciones y va generando posibles rutas en base a ellas y selecciona la mejor, es decir si se tiene 6 rutas de camiones, por lo menos se considerara 6 posibles planes para alcanzar el objetivo, las cuales computan y se van descartando en base a las restricciones y a su calidad. Esto nos lleva a la idea de utilizar el algoritmo de Dijkstra como nuestra heuristica de busqueda en un planificador, mejorando con ello los tiempos de ejecución.

En la figura 4.26 mostramos un plan arrojado por el planificador SGPLAN, donde en color rojo se muestra la simulación de avance de la unidad de transporte hacia la parada de autobuses, y el resto indica los movimientos a realizarse por el usuario para alcanzar su objetivo.

```
; Time 78.41
; ParsingTime 0.20
; NrActions 19
; MakeSpan
; MetricValue 770.019
; PlanningTechnique Modified-FF(best-first search) as the subplanner

0.001: (CAMINAR PAS1 C1 C13) [72.0000]
0.002: (DESPLAZAR_BUS RUTA_3_2 C131 C119) [0.0000]
0.003: (DESPLAZAR_BUS RUTA_3_2 C119 C120) [0.0000]
72.004: (CAMINAR PAS1 C13 C25) [72.0000]
144.005: (CAMINAR PAS1 C25 C37) [72.0000]
216.006: (ASCENDER_BUS PAS1 RUTA_2_1 C37) [0.0000]
216.007: (DESPLAZAR_BUS_PAS PAS1 RUTA_2_1 C37 C49) [19.0000]
235.008: (DESPLAZAR_BUS_PAS PAS1 RUTA_2_1 C49 C61) [17.0000]
252.009: (DESPLAZAR_BUS_PAS PAS1 RUTA_2_1 C61 C60) [14.0000]
266.010: (DESCENDER_BUS PAS1 RUTA_2_1 C60) [0.0000]
266.011: (CAMINAR PAS1 C60 C72) [72.0000]
338.012: (CAMINAR PAS1 C72 C84) [72.0000]
410.013: (CAMINAR PAS1 C84 C96) [72.0000]
482.014: (CAMINAR PAS1 C96 C108) [72.0000]
554.015: (CAMINAR PAS1 C108 C120) [72.0000]
626.016: (ASCENDER_BUS PAS1 RUTA_3_2 C120) [0.0000]
626.017: (DESCENDER_BUS PAS1 RUTA_3_2 C120) [0.0000]
626.018: (CAMINAR PAS1 C120 C132) [72.0000]
698.019: (CAMINAR PAS1 C132 C144) [72.0000]
```

Figura 4.26: Plan arrojado por el planificador SGPLAN, donde se indica las acciones de simulación de movimiento de la unidad de transporte sincronizandolo con el usuario en un punto.

4.2.2 Escabilidad de los planificadores utilizados para nuestro modelo de planificacón de transporte público

Esta parte se llevó a cabo con la finalidad de conocer la competitividad de los planificadores utilizados de acuerdo al enfoque de nuestro problema de planificación de rutas de viaje. Para ello se tomó en cuenta el número de rutas que maneja una de las empresas de transporte público de la Ciudad de Monterrey, N. L., donde se tiene 25 rutas, sin embargo considerando los ramales que manejan se hace un total de 42 rutas. Por lo tanto veremos si los planificadores utilizados soportan esta cantidad de requerimientos.

Escabilidad del planificador LPG para nuestro modelo completo de planificación

En base a la característica anterior del número de rutas existentes para la empresa mencionada y mediante la experimentación con nuestro modelo completo de planificación pudimos constatar que este planificador puede soportar una red 1024 nodos como máximo y un total de 2436 paradas repartidas entre todas las rutas (en nuestro caso cada una de estas tiene el mismo número de puntos de ascensos, para cada una 58).

Teniendo para esta red un total de 9080 posibles movimientos de desplazamiento que se pueden realizar en la red, de las cuales 4208 corresponden a los desplazamientos a realizarse caminando y 4872 corresponden a la parte de las rutas de autobuses.

Escabilidad del planificador SGPLAN para nuestro modelo completo de planificación

Para el caso del planificador SGPLAN se tiene que el máximo número de nodos de la red es de 6276 nodos para las 42 rutas planteadas y todas ellas con un tamanño de 90 paradas, dando así un total de 3780 paradas tomando en cuenta nuestro modelo de planificación completo. Todo esto permitiendo un total de 10568 movimientos de desplazamiento, ya sea caminando o en alguna unidad de transporte.

Este planificador es capaz de soportar un mayor número de paradas solo que hasta 90 de ellas entrega un resultado.

Lo mostrado anteriormente demuestra que ambos planificadores pueden soportar un sistema de transporte público de autobuses de una ciudad mediana, ya que el número de rutas así como de paradas son suficientes para satisfacer los requerimientos que estas podrían necesitar. Lo anteriormente mencionado se muestra en la tabla 4.19, donde se presenta un listado de algunas de las ciudades medianas y más representativas de México con su respectivo número de rutas de autobuses de transporte público tomadas de [18], así como la población para cada una de estas localidades.

Ciudad	**Número de rutas**	**Población**
Villahermosa	37	756,065
Aguascalientes	47	752,903
Cancún	31	628,306
Culiacán	67	733,370
Durango	57	582,267
Tuxtla Gutierrez	111	553,374
Torréon	29	1,215,993
San Luis Potosí	55	1,097,906
Los Mochis	25	416,299
Manzanillo	15	161,420
Mazatlán	42	438,434
Morelia	47	729,279
Puerto Vallarta	49	203,342
Saltillo	51	823,128

Tabla 4.19: Número de rutas correspondientes a ciudades representativas de México, así como la población según el censo de población 2010.

Como se puede ver en los datos de la tabla 4.19, los planificadores utilizados para el desarrollo de nuestra investigación son capaces de soportar la red de transporte público de autobuses de la mayoría de estas ciudades, inclusive podríamos

decir que de la mayor parte de los sistemas existentes en el país.

4.2.3 Dinámismo en el Ambiente

Hasta el momento solo se ha trabajado con redes de transporte estáticas, pero ¿Qué pasaría si existiera una eventualidad que afectara el transito del sistema?, para ello se podría reescribir totalmente el archivo problema y poder así arrojar planes confiables considerando estos cambios. Para ello se estimó que reescribir el archivo problema para una red de un millón de nodos tomaría a nuestros recursos crearlo en aproximadamente 6 segundos, lo cual no es un tiempo considerable, debido a que solo se realizará esta actualización cuando exista una eventualidad en el sistema.

La otra forma sería modificar el modelo de planificación donde en las acciones de movimiento exista una condición para que no contemple nodos por donde no se pueda transitar, y de este modo solo se actualizaría en el estado inicial del archivo problema la sección de nodos afectados cuando exista un percance disminuyendo con esto el tiempo que se lleva en reescribir tal archivo, aunque esto podría aumentar el tiempo de espera por parte del usuario de un plan de viaje, debido a que se optimiza en espacio y tiempo.

Capítulo 5

Conclusiones y trabajo futuro

5.1 Conclusiones

En el presente trabajo podemos concluir que las técnicas de planificación de Inteligencia Artificial tienen una factibilidad práctica muy alta para el desarrollo de modelos de planificación para el transporte público, permitiendo representar las propiedades de la red de transporte, la cual utilizamos en esta investigación para la generación de planes de viaje para el usuario.

Estas técnicas, además de lo anterior, permiten la implementación de métricas (preferencias del usuario) que fungieron como restricciones en esta investigación de forma muy simple, las cuales nunca antes se habían tratado. Teniendo así dominios de planificación relativamente pequeños para nuestros propósitos en comparación con las implementaciones de algoritmos tradicionales existentes.

En base a los resultados obtenidos tenemos que a un hay mucho trabajo que realizar desde la parte de los planificadores, ya que los tiempos de resolución de un problema dado en varias ocasiones tendieron a ser muy altos sobre todo los mostrados por el planificador LPG, y esto no es aplicable a lo que pretendemos hacer, ya que el usuario necesita una respuesta pronta y confiable.

En la calidad de los planes arrojados por estos planificadores tienden ser bastante competitivas, y en el caso del modelo básico el planificador PLAN encontró en todos los casos el valor óptimo, y logro solucionar cada uno de los problemas que se

le presentaron, solo que no pudo soportar los modelos posteriores. Siguiendo con la calidad de los planes LPG reportó los peores resultados, mientras que SGPLAN fue superior en los modelos posteriores al básico. Sin embargo LPG logró resolver más archivos problema que SGPLAN.

Otra de nuestras conclusiones es que entre más restricciones tienen este tipo de modelos de planificación más cercanos al óptimo se encontrarán sus resultados del valor óptimo, sin embargo sus tiempos de resolución tienden a aumentar.

Entrados en los modelos generados, si el usuario no quiere hacer uso de ninguna preferencia que funja de restriccón, es necesario hacer uso de *(at end(not(via_cam ?pas ?de ?a)))* en la acción *Caminar*, esto para evitar que se visite más de una vez un mismo nodo.

Otro de los problemas encontrados en los planificadores a parte de los tiempos de resolución altos, es su capacidad, es decir, el planificador LPG solo nos permite representar una red de 1089 nodos, mientras que SGPLAN tiene una capacidad de 729 nodos siendo esto un número pequeño de nodos considerando a los necesarios para cubrir una ciudad mediana.

Por lo tanto tenemos que los modelos de planificación son aptos para el tratado de este tipo de problemas con métricas, pero no así los planificadores debido a lo antes mencionado.

Otra de nuestras conclusiones es que se pudo determinar que los planificadores SGPLAN y LPG tienen la capacidad de soportar una red de transporte público de la mayoría de las ciudades de México, o al menos una inmensa parte del sistema de servicio público para nuestro modelo completo de planificación.

5.2 Recomendaciones

Para un mejor funcionamiento del modelo propuesto tenemos lo siguiente:

- Ampliar el modelo de planificación agregando otros tipos de transporte público, ya sea el sistema del metro, taxis, o algún otro medio.
- Realizar pruebas con el modelo donde se considera tiempos de espera.
- Agregar al modelo de planificación la preferencia de viaje cómodo. Es decir, asegurarle con cierta probabilidad que obtendrá un asiento al momento de abordar o que la unidad no valla congestionada.
- Realizar una presentación llamativa de entrada y salida de datos para el usuario.

5.3 Trabajo futuro

Para trabajo futuro proponemos los siguientes puntos:

- Actualización de datos de forma continua.
- Llevar el modelo de planificación desarrollado a un sistema real de transporte público.
- Incorporar al modelo de planificación heurísticas para la proyección de carga de usuarios.
- Capacitar al modelo de planificación para la proyección de tiempos de viaje.
- Desarrollo de un modelo de planificación continuo.
- Desarrollo de un planificador que genere planes de forma continua.
- Ampliación de los planificadores centralizandolos a nuestro problema.

Apéndice A

Apéndice

A.0.1 Modelo de planificacón con extensión de líneas del metro

Para este modelo de planificación partimos del modelo total presentado en el capítulo 3, donde se retoma el dominio presentado en la figura 3.16 en su totalidad, los cambios a realizar para esta extensión se presentan en el archivo problema donde se deben declarar los objetos, así como sus estados iniciales correspondientes al servicio del metro.

Los objetos a declarar son los vagones del metro, que en lugar de rutas son definidos por líneas, así como también indicando su dirección, todo esto muy similar a lo mostrado cuando se declaró las unidades de autobuses anteriormente, esto se ve en la figura A.1

Linea_1_1

Linea_1_2

Linea_2_1

Linea_2_2

Figura A.1: Objetos a agregar al modelo de planificación en el archivo problema para la extención de lineas del metro

Para el caso de los estados iniciales se debe declarar la unidad y la ubicación

```
(autobus Line_1_1)
(autobus Line_1_2)
(autobus Line_2_1)
(autobus Line_2_2)
(=(costo Line_1_1)4.5)
(=(costo Line_1_2)4.5)
(=(costo Line_2_1)4.5)
(=(costo Line_2_2)4.5)
(via_bus  Line_1_1 c49 c48)
 (=(tiempo_b  Line_1_1 c49 c48)4)
 (via_bus  Line_1_2 c48 c49)
 (=(tiempo_b  Line_1_2 c48 c49)4)
 (via_bus  Line_1_1 c48 c47)
                    *
                    *
                    *
(via_bus  Line_2_1 c50 c51)
 (=(tiempo_b  line_2_1 c50 c51)4)
 (via_bus  line_2_2 c51 c50)
 (=(tiempo_b  line_2_2 c51 c50)4)
```

Figura A.2: Estados inciales para los nuevos objetos a agregar al modelo de planificación en el archivo problema para la extención de lineas del metro

de estos vagones del metro y por donde se pueden desplazar, así como también los tiempos que le toma desplazarse entre esos puntos, esto se ve en la figura A.2 donde se observa la inicialización de los costos correspondientes por abordar este servicio, agregando esto al archivo problema de la figura 3.17.

Apéndice B

Apéndice

B.0.2 Dominios de modelos de planificación para una métrica a optimizar

En este apéndice mostramos dominios de planificación semi-básicos, donde se muestra las acciones descritas anteriormente, solo que apegadas a la métrica a optimizar en cada dominio, estos dominios se enlistan a continuación.

- Dominio de planificación: Transferencias
- Dominio de planificación: Distancia a recorrer
- Dominio de planificación: Costo
- Dominio de planificación: Costo-Tiempo

```
(define (domain ruta-transporte)
    (:requirements :numeric-fluents :durative-actions)

    (:predicates
        (pasajero ?x)
        (autobus ?r)
        (ir_cam ?x ?p1 ?p2)
        (at ?x ?p)
        (rut_bus ?r ?p)
        (via_bus ?r ?p1 ?p2)
        (in ?x ?r)
        )

    (:functions
        (longitud_cam ?pas ?de ?a)
        (tiempo_b ?bus ?de ?a)
        (pasajero_in_bus)
        (longitud_c)
        (long_max_cam)
        (dist_maxima)
        (dist_max_c)
        (num_ascensos)
    )

    (:durative-action Caminar
        :parameters(?pas ?de ?a)
        :duration(=?duration 0.0001)
        :condition(and
            (at start(pasajero ?pas))
            (at end(<(longitud_c)(long_max_cam)))
            (at start(<(dist_max_c)(dist_maxima)))
            (at start(ir_cam ?pas ?de ?a))
            (at start(at ?pas ?de))

            )
        :effect(and
            (at start(not(at ?pas ?de)))
            (at end(at ?pas ?a))
            (at end(increase(dist_max_c)(longitud_cam ?pas ?de ?a)))
            (at end(increase(longitud_c)(longitud_cam ?pas ?de ?a)))
            )
    )

(:durative-action Ascender
        :parameters(?pas ?bus ?de)
        :duration (=?duration 0.0001)
        :condition(and
                (at start(pasajero ?pas))
                (at start(autobus ?bus))
                (at start(rut_bus ?bus ?de))
                (at start(at ?pas ?de))
                )
            :effect(and
                (at start(not(at ?pas ?de)))
                (at end(in ?pas ?bus))
                (at end(increase (pasajero_in_bus)100))
                (at end(decrease(longitud_c)(longitud_c)))
                (at end(increase(num_ascensos)1))
                )
    )

    (:durative-action Desplazar_bus_pas
        :parameters(?pas ?bus ?de ?a)
        :duration(=?duration 0.0001)
        :condition(and
            (at start(pasajero ?pas))
            (at start(autobus ?bus))
            (at start(in ?pas ?bus))
            (over all(in ?pas ?bus))
            (at start(>(pasajero_in_bus)0))
            (at start(rut_bus ?bus ?de))
            (at start(via_bus ?bus ?de ?a))
            )
        :effect(and
            (at start(not(rut_bus ?bus ?de)))
            (at end(rut_bus ?bus ?a))
            )
    )

(:durative-action Desplazar_bus
        :parameters(?bus ?de ?a)
        :duration (=?duration 0.0001)
        :condition(and
            (at start(autobus ?bus))
            (at start(rut_bus ?bus ?de))
            (at start(via_bus ?bus ?de ?a))
            (at start(<(pasajero_in_bus)100))
            )
        :effect(and
            (at start(not(rut_bus ?bus ?de)))
            (at end(rut_bus ?bus ?a))
            )
    )

(:durative-action Descender
        :parameters (?pas ?bus ?a)
        :duration (=?duration 0.0001)
        :condition(and
            (at start(pasajero ?pas))
            (at start(autobus ?bus))
            (at start(in ?pas ?bus))
            (at start(rut_bus ?bus ?a))
            )
        :effect(and
            (at start(not (in ?pas ?bus)))
            (at start(not (rut_bus ?bus ?a)))
            (at end(at ?pas ?a))
            (at end(decrease (pasajero_in_bus)(pasajero_in_bus)))
            )
    )
)
```

Figura B.1: Dominio para el modelo de planificación para optimizar el número de transferencias en el plan de viaje

```
(define (domain ruta-transporte)
    (:requirements :numeric-fluents :durative-actions)

    (:predicates
        (pasajero ?x)
        (autobus ?r)
        (ir_cam ?x ?p1 ?p2)
        (at ?x ?p)
        (rut_bus ?r ?p)
        (via_bus ?r ?p1 ?p2)
        (in ?x ?r)
        )
    (:functions
        (longitud_cam ?pas ?de ?a)
        (tiempo_b ?bus ?de ?a)
        (pasajero_in_bus)
        (longitud_c)
        (long_max_cam)
        (dist_maxima)
        (dist_max_c)
        (longitud_caminada)
    )

    (:durative-action Caminar
        :parameters(?pas ?de ?a)
        :duration(=?duration (/(longitud_cam ?pas ?de ?a)1.39))
        :condition(and
            (at start(pasajero ?pas))
            (at end(<(longitud_c)(long_max_cam)))
            (at start(<(dist_max_c)(dist_maxima)))
            (at start(ir_cam ?pas ?de ?a))
            (at start(at ?pas ?de))

            )
        :effect(and
            (at start(not(at ?pas ?de)))
            (at end(at ?pas ?a))
            (at end(increase(dist_max_c)(longitud_cam ?pas ?de ?a)))
            (at end(increase(longitud_c)(longitud_cam ?pas ?de ?a)))
            (at end(increase(longitud_caminada)(longitud_cam ?pas ?de ?a)))
            )
    )

(:durative-action Ascender
        :parameters(?pas ?bus ?de)
        :duration (=?duration 0.0001)
        :condition(and
                (at start(pasajero ?pas))
                (at start(autobus ?bus))
                (at start(rut_bus ?bus ?de))
                (at start(at ?pas ?de))
                )
            :effect(and
                (at start(not(at ?pas ?de)))
                (at end(in ?pas ?bus))
                (at end(increase (pasajero_in_bus)100))
                (at end(decrease(longitud_c)(longitud_c)))
                )
    )

    (:durative-action Desplazar_bus_pas
        :parameters(?pas ?bus ?de ?a)
        :duration(=?duration 0.0001)
        :condition(and
            (at start(pasajero ?pas))
            (at start(autobus ?bus))
            (at start(in ?pas ?bus))
            (over all(in ?pas ?bus))
            (at start(>(pasajero_in_bus)0))
            (at start(rut_bus ?bus ?de))
            (at start(via_bus ?bus ?de ?a))
            )
        :effect(and
            (at start(not(rut_bus ?bus ?de)))
            (at end(rut_bus ?bus ?a))
            )
    )

(:durative-action Desplazar_bus
        :parameters(?bus ?de ?a)
        :duration (=?duration 0.0001)
        :condition(and
            (at start(autobus ?bus))
            (at start(rut_bus ?bus ?de))
            (at start(via_bus ?bus ?de ?a))
            (at start(<(pasajero_in_bus)100))
            )
        :effect(and
            (at start(not(rut_bus ?bus ?de)))
            (at end(rut_bus ?bus ?a))
            )
    )

(:durative-action Descender
        :parameters (?pas ?bus ?a)
        :duration (=?duration 0.0001)
        :condition(and
            (at start(pasajero ?pas))
            (at start(autobus ?bus))
            (at start(in ?pas ?bus))
            (at start(rut_bus ?bus ?a))
            )
        :effect(and
            (at start(not (in ?pas ?bus)))
            (at start(not (rut_bus ?bus ?a)))
            (at end(at ?pas ?a))
            (at end(decrease (pasajero_in_bus)(pasajero_in_bus)))
            )
    )
)
```

Figura B.2: Dominio para el modelo de planificación donde solo se toma en cuenta las métricas referentes a longitud recorrida caminando esto para optimizar la distancia a recorrer por el usuario en el plan a computar

```
(define (domain ruta-transporte)
    (:requirements :numeric-fluents :durative-actions)

    (:predicates
        (pasajero ?x)
        (autobus ?r)
        (ir_cam ?x ?p1 ?p2)
        (at ?x ?p)
        (rut_bus ?r ?p)
        (via_bus ?r ?p1 ?p2)
        (in ?x ?r)
        )

    (:functions
        (longitud_cam ?pas ?de ?a)
        (tiempo_b ?bus ?de ?a)
        (pasajero_in_bus)
        (longitud_c)
        (long_max_cam)
        (dist_maxima)
        (dist_max_c)
        (costo-total)
        (costo ?bus)
        (costo_maximo)
    )

    (:durative-action Caminar
        :parameters(?pas ?de ?a)
        :duration(=?duration 0.0001)
        :condition(and
            (at start(pasajero ?pas))
            (at end(<(longitud_c)(long_max_cam)))
            (at start(<(dist_max_c)(dist_maxima)))
            (at start(ir_cam ?pas ?de ?a))
            (at start(at ?pas ?de))

            )
        :effect(and
            (at start(not(at ?pas ?de)))
            (at end(at ?pas ?a))
            (at end(increase(dist_max_c)(longitud_cam ?pas ?de ?a)))
            (at end(increase(longitud_c)(longitud_cam ?pas ?de ?a)))
            )
    )

(:durative-action Ascender
        :parameters(?pas ?bus ?de)
        :duration (=?duration 0.0001)
        :condition(and
                (at start(pasajero ?pas))
                (at start(autobus ?bus))
                (at start(rut_bus ?bus ?de))
                (at start(at ?pas ?de))
                (at start(<(costo-total)(costo_maximo)))
                )
            :effect(and
                (at start(not(at ?pas ?de)))
                (at end(in ?pas ?bus))
                (at end(increase(costo-total)(costo ?bus)))
                (at end(increase (pasajero_in_bus)100))
                (at end(decrease(longitud_c)(longitud_c)))
                (at end(increase(num_ascensos)1))
                )
    )

    (:durative-action Desplazar_bus_pas
        :parameters(?pas ?bus ?de ?a)
        :duration(=?duration 0.0001)
        :condition(and
            (at start(pasajero ?pas))
            (at start(autobus ?bus))
            (at start(in ?pas ?bus))
            (over all(in ?pas ?bus))
            (at start(>(pasajero_in_bus)0))
            (at start(rut_bus ?bus ?de))
            (at start(via_bus ?bus ?de ?a))
            )
        :effect(and
            (at start(not(rut_bus ?bus ?de)))
            (at end(rut_bus ?bus ?a))
            )
    )

(:durative-action Desplazar_bus
        :parameters(?bus ?de ?a)
        :duration (=?duration 0.0001)
        :condition(and
            (at start(autobus ?bus))
            (at start(rut_bus ?bus ?de))
            (at start(via_bus ?bus ?de ?a))
            (at start(<(pasajero_in_bus)100))
            )
        :effect(and
            (at start(not(rut_bus ?bus ?de)))
            (at end(rut_bus ?bus ?a))
            )
    )

(:durative-action Descender
        :parameters (?pas ?bus ?a)
        :duration (=?duration 0.0001)
        :condition(and
            (at start(pasajero ?pas))
            (at start(autobus ?bus))
            (at start(in ?pas ?bus))
            (at start(rut_bus ?bus ?a))
            )
        :effect(and
            (at start(not (in ?pas ?bus)))
            (at start(not (rut_bus ?bus ?a)))
            (at end(at ?pas ?a))
            (at end(decrease (pasajero_in_bus)(pasajero_in_bus)))
            )
    )
)
```

Figura B.3: Dominio para el modelo de planificación para optimizar el costo de económico de transporte para el plan de viaje computado

```
(define (domain ruta-transporte)
    (:requirements :numeric-fluents :durative-actions)

    (:predicates
        (pasajero ?x)
        (autobus ?r)
        (ir_cam ?x ?p1 ?p2)
        (at ?x ?p)
        (rut_bus ?r ?p)
        (via_bus ?r ?p1 ?p2)
        (in ?x ?r)
        )
    (:functions
        (longitud_cam ?pas ?de ?a)
        (tiempo_b ?bus ?de ?a)
        (pasajero_in_bus)
        (longitud_c)
        (long_max_cam)
        (dist_maxima)
        (dist_max_c)
        (p_tiempo)
        (p_costo)
        (costo-total)
        (costo ?bus)
    )

    (:durative-action Caminar
        :parameters(?pas ?de ?a)
        :duration(=?duration (/(longitud_cam ?pas ?de ?a)1.39))
        :condition(and
            (at start(pasajero ?pas))
            (at end(<=(longitud_c)(long_max_cam)))
            (at start(<(dist_max_c)(dist_maxima)))
            (at start(ir_cam ?pas ?de ?a))
            (at start(at ?pas ?de))

            )
        :effect(and
            (at start(not(at ?pas ?de)))
            (at end(at ?pas ?a))
            (at end(increase(dist_max_c)(longitud_cam ?pas ?de ?a)))
            (at end(increase(longitud_c)(longitud_cam ?pas ?de ?a)))
            )
    )

(:durative-action Ascender
        :parameters(?pas ?bus ?de)
        :duration (=?duration 0.0001)
        :condition(and
                (at start(pasajero ?pas))
                (at start(autobus ?bus))
                (at start(rut_bus ?bus ?de))
                (at start(at ?pas ?de))
                )
            :effect(and
                (at start(not(at ?pas ?de)))
                (at end(in ?pas ?bus))
                (at end(increase (pasajero_in_bus)100))
                (at end(increase(costo-total)(costo ?bus)))
                (at end(decrease(longitud_c)(longitud_c)))
                )
    )

(:durative-action Desplazar_bus_pas
        :parameters(?pas ?bus ?de ?a)
        :duration(=?duration (tiempo_b ?bus ?de ?a))
        :condition(and
            (at start(pasajero ?pas))
            (at start(autobus ?bus))
            (at start(in ?pas ?bus))
            (over all(in ?pas ?bus))
            (at start(>(pasajero_in_bus)0))
            (at start(rut_bus ?bus ?de))
            (at start(via_bus ?bus ?de ?a))
            )
        :effect(and
            (at start(not(rut_bus ?bus ?de)))
            (at end(rut_bus ?bus ?a))
            )
    )

(:durative-action Desplazar_bus
        :parameters(?bus ?de ?a)
        :duration (=?duration 0.0001)
        :condition(and
            (at start(autobus ?bus))
            (at start(rut_bus ?bus ?de))
            (at start(via_bus ?bus ?de ?a))
            (at start(<(pasajero_in_bus)100))
            )
        :effect(and
            (at start(not(rut_bus ?bus ?de)))
            (at end(rut_bus ?bus ?a))
            )
    )

(:durative-action Descender
        :parameters (?pas ?bus ?a)
        :duration (=?duration 0.0001)
        :condition(and
            (at start(pasajero ?pas))
            (at start(autobus ?bus))
            (at start(in ?pas ?bus))
            (at start(rut_bus ?bus ?a))
            )
        :effect(and
            (at start(not (in ?pas ?bus)))
            (at start(not (rut_bus ?bus ?a)))
            (at end(at ?pas ?a))
            (at end(decrease (pasajero_in_bus)(pasajero_in_bus)))
            )
    )
)
```

Figura B.4: Dominio para el modelo de planificación para optimizar la combinación de preferencias costo-tiempo para el plan de viaje computado

Bibliografía

[1] 1998, I., «International Planing Competition 1998», recurso libre, disponible en http://ipc.icaps-conference.org/, 1998.

[2] 2002, I., «International Planing Competition 2002», recurso libre, disponible en http://ipc.icaps-conference.org/, 2002.

[3] 2004, I., «International Planing Competition 2004», recurso libre, disponible en http://ipc.icaps-conference.org/, 2004.

[4] Aragón, I. R., «Búsqueda heurística: Algoritmo A*», recurso libre, disponible en http://poiritem.wordpress.com/2010/01/14/6-5-2-busqueda-heuristica-algoritmo-a/, 2010.

[5] Bast, H., S. Funke y D. Matijevic, «TRANSIT-ultrafast shortest-path queries with linear-time preprocessing», *In 9th DIMACS Implementation Challenge*, 2006.

[6] Bast, H., S. Funke, D. Matijevic, P. Sanders y D. Schultes, «In Transit to Constant Time Shortest-Path Queries in Road Networks», *In Proceedings of the Ninth Workshop on Algorithm Engineering and Experiments and the Fourth Workshop on Analytic Algorithmics and Combinatorics 2007*, págs. 46–59, 2007.

[7] Benitez, B. N., «El transporte urbano en el área metropolitana de Monterrey», recurso libre disponible en hhttp://juridicas.unam.mx/publica/librev/rev/gac/cont/36/pr11.pdf, 1989.

[8] BORNSTEIN, N. y S. SVANEI, *Master thesis: Heuristics in generic planning using pddl*, primera edición, Technical University of Denmark, DTU Informatics Department of Informatics and Mathematical Modelling Richard Petersens plads, Building 321, Dinamarca, 2011.

[9] CASTILLO, L., E. ARMENGOL, E. ONAINDÍA, L. SEBASTIÁ, J. GONZÁLEZ-BOTICARIO, A. RODRÍGUEZ, S. FERNÁNDEZ, J. D. ARIAS y D. BORRAJO, «SAMAP. An user-oriented adaptive system for planning tourist visits», *Expert Syst. Appl.*, **2**(34), págs. 1318–1332, 2008.

[10] CHEN, Y., C. W. HSU y B. W. WAH, «SGPlan 4: Subgoal Partitioning and Resolution in Planning», recurso online, disponible en http://wah.cse.cuhk.edu.hk/wah/programs/SGPlan/sgplan4.html, 2004.

[11] CHEN, Y., C. W. HSU y B. W. WAH, «SGPLAN: Subgoal partitioning and resolution in planning», *American Association for Artificial Intelligence*, **1**(1), págs. 1–2, 2004.

[12] CHEN, Y., C. W. HSU y B. W. WAH, «Subgoal Partitioning and Resolution in SGPLAN», *American Association for Artificial Intelligence*, **1**(1), págs. 1–2, 2005.

[13] CHEN, Y., C. W. HSU y B. W. WAH, «Temporal Planning using Subgoal Partitioning and Resolution in SGPlan», *Artificial Intelligence Research*, **26**(1), págs. 323–369, 2006.

[14] CODDINGTON, A., M. FOX y D. LONG, «Handling Durative Actions in Classical Planning Frameworks», *Proceedings of the 20th Workshop of the UK Planning and Scheduling Special Interest Group*, **1**(1), págs. 44–58, 2001.

[15] COMPUTING WITH PERL, P. D. L. S., «Perl Data Language Scientific computing with Perl», recurso libre, disponible en http://pdl.perl.org/, 2012.

[16] CTLC-ST, «Regulación del transporte público», recurso libre, disponible en http://www.ctlc-st.gob.pe/PDF2004.

[17] DANTZIG, G. B., *Linear programming and extensions*, Princeton Univ. Press, Princeton, NJ, 1963.

[18] DE CAMIONES, R., «Rutas de Camiones», recurso libre, disponible en http://www.rutasdecamiones.com/, 2013.

[19] DIJKSTRA, E. W., «A note on two problems in connexion with graphs», *Numerische Mathematik*, **1**(34), págs. 269–271, 1959.

[20] DIRECTA, R., «Rutas de Camiones y Líneas del Metro de Monterrey y su Área Metropolitana», recurso libre, disponible en http://mty.rutadirecta.com/, 2013.

[21] FLÓREZ, J. E., A. TORRALBA, J. GARCÍA, C. L. LÓPEZ, A. GARCÍA-OLAYA y D. BORRAJO, «TIMIPLAN: An Application to Solve Multimodal Transportation Problems», *Proceedings of SPARK, Scheduling and Planning Applications workshop, ICAPS'10*, **1**(1), págs. 0–0, 2010.

[22] FOX, M. y D. LONG, «PDDL2.1: An Extension to PDDL for Expressing Temporal Planning», *Artificial Intelligence Research*, **1**(1), págs. 61–124, 2001.

[23] GARCÍA, C. M. y G. M. ORTEGA, «Route planning algorithms: Planific@ Project», *International Journal of Artificial Intelligence and Interactive Multimedia*, **1**(2), págs. 57–66, 1959.

[24] GARCÍA-OLAYA, A., «Seminario PLG: Overview of PDDL», recurso libre, disponible en http://www.plg.inf.uc3m.es/slides/11-09-07-angel-pddl.pdf, 2007.

[25] GARRIDO, A., M. FOX y D. LONG, «A temporal Planning System for Durative Actions of PDDL2.1», *ECAI 2002*, **1**(1), págs. 586–590, 2002.

[26] GARRIDO, A., E. MARZAL y E. ONAINDÍA, «Dos aproximaciones de planificación temporal en el lenguaje PDDL2.1», *Taller de Razonamiento Temporal*, **1**(1), págs. 41–49, 2001.

[27] GEREVINI, A., A. SAETTI, I. SERINA y P. TONINELLI, «Planning in Pddl 2.2 Domains whith LPG-TD», *14th International Conference on Automated Planning and Scheduling ICAPS 2004*, 2004.

[28] GEREVINI, A., I. SERINA, A. SAETTI y S. SPINONI, «Local Search Techniques for Temporal Planning in LPG», *ICAPS*, págs. 91–104, 2003.

[29] GEREVINI, A., I. SERINA, A. SAETTI y S. SPINONI, «Local Search Tecniques for Temporal Planning in LPG», *13th International Conference on Automated Planning and Scheduling , AAAI*, **1**(1), págs. 62–71, 2003.

[30] GEREVINI, A. E., A. SAETTI y I. SERINA, «An approach to efficient planning with numerical fluents and multi-criteria plan quality*», *Science Direct*, **172**(1), págs. 899–944, 2008.

[31] GEREVINI, A. E., A. SAETTI, I. SERINA y M. VALLATI, «Welcome to the homepage of LPG», recurso libre, disponible en http://zeus.ing.unibs.it/lpg/, 2011.

[32] GHALLAB, M., A. HOWE, C. KNOBLOCK, D. MCDERMOTT, A. RAM, M. VELOSO, D. WELD y D. WILKINS, «PDDL- the planning domain definition language Version 1.2», recurso libre, disponible en http://www.inf.ed.ac.uk/teaching/courses/propm/papers/pddl.pdf, 1998.

[33] GOLDBERG, A. V., H. KAPLAN y R. WERNECK, «Reach for A*: an efficient point-to-point shortest path algorithm», recurso libre, disponible en http://www.cs.princeton.edu/courses/archive/spr09/cos423/Lectures/reach-mit.pdf, 2009.

[34] GUTMAN, R., «Reach-Based Routing: A New Approach to Shortest Path Algorithms Optimized for Road Networks», *ALENEX, 2004*, págs. 100–11, 2004.

[35] HAMDY, T., *Investigación de operaciones*, 7ª edición, Pearson Educación, Distrito Federal, México, 2004.

[36] HELMERT, M., «The Fast Downward Planning System», *Journal of Artificial Intelligence Research*, **26**, págs. 191–246, 2006.

[37] HOUGHTON, J., J. REINERS y L. LIM, «Transporte inteligente: Como mejorar la movilidad en las ciudades», *IBM Global Business Services*, págs. 1–18, 2009.

[38] HRNCÍR, J., *Thesis: Improving a Collaborative Travel Planning Application*, primera edición, School of Informatics University of Edinburgh, Reino Unido, 2011.

[39] HRNCÍR, J. y M. ROVATSOS, «Applying Strategic Multiagent Planning to Real World Travel Sharing Problems», *7th Workshop on Agents in Traffic and Transportation, AAMAS (June 2012)*, **1**(1), págs. 0–0, 2011.

[40] HSU, C. W., B. W. WAH, R. HUANG y Y. CHEN, «SGPlan 5: Subgoal Partitioning and Resolution in Planning», recurso libre, disponible en http://wah.cse.cuhk.edu.hk/wah/programs/SGPlan/, 2012.

[41] IDELAB, «Algoritmo A*», recurso libre, disponible en http://idelab.uva.es/algoritmo, 2012.

[42] INEGI, «Zona Metropolitana del Valle de México: Encuesta Origen-Destino 2007», INEGI, 2007.

[43] KLEINER, A., «Writing Planning Domains and Problems in PDDL», recurso libre, disponible en http://www.ida.liu.se/ TDDC17/info/labs/planning/2004/writing.html, 2004.

[44] LIU, C.-L., «Best-Path Planning for Public Transportation Systems», *Proceedings of the 5th International Conference on Intelligente Transportation Systems*, págs. 834–839, 2002.

[45] MALTE, H., «An Introduction to PDDL», recurso libre, disponible en http://www.cs.toronto.edu/ sheila/2542/w09/A1/introtopddl2.pdf, 2009.

[46] MARES PEÑA, J. A., *Tesis: Calidad en el Servicio del Transporte Urbano*, primera edición, U. A. N. L., San Nicolas de los Garza, N.L., 1996.

[47] MOLINERO, A. R. y L. I. ARELLANO, *Transporte público: planeación, diseño, operación y administración*, primera edición, UAEM, México, 1997.

[48] MOLINERO, A. R. y L. I. ARELLANO, *Transporte público: planeación, diseño, operación y administración*, primera edición, UAEM, México, 2005.

[49] PEDNAULT, E. P. D., «Formulating Multiagent, Dynamic-World Problems in the Classical Planning Framework», *Morgan Kaufmann Publishers*, págs. 47–82, 1987.

[50] PEDNAULT, E. P. D., «ADL: Exploring the Middle Ground Between STRIPS and the Situation Calculus», *Proceedings of the First International Conference on Principles of Knowledge Representation and Reasoning*, págs. 324–332, 1989.

[51] PITV, «Programa de Transporte y Vialidad de la Ciudad de México», recurso libre disponible en hhtp://www.ville-en-mouvement.com/ameriquelatine/telechargements/PITVNUESTRO.pdf, 2009.

[52] PLANNING, S. A., «Description of the SHOP Project», recurso libre, disponible en http://www.cs.umd.edu/projects/shop/description.html, 2012.

[53] RÖGER, G., P. EYERICH y R. MATTMÜLLER, «TFD: A Numeric Temporal Extension to Fast Downward», recurso libre, disponible en http://ipc.informatik.uni-freiburg.de/Planners?action=AttachFile&do=view&target=TFD.pdf, 2008.

[54] RUSSELL, S. y P. NORVIG, *Inteligencia Artificial. Un Enfoque Moderno*, segunda edición, Pearson - Prentice Hall., Madrid, España., 2004.

[55] SAETTI, A., «On Managing Temporal Information for Handling Durative Actions in LPG», *AI*IA, 2003*, **2829**, págs. 91–104, 2003.

[56] SANDERS, P. y D. SCHULTES, «Engineering highway hierarchies», *ESA 2006 Proceedings of the 14th conference on Annual European Symposium*, **14**, págs. 804–816, 2006.

[57] SANDERS, P. y D. SCHULTES, «Highway Hierarchies Hasten Exact Shortest Path Queries», *ESA*, **3669**, págs. 568–579, 2006.

[58] SANDERS, P. y D. SCHULTES, «Engineering Fast Route Planning Algorithms», *WEA*, **4525**, págs. 23–36, 2007.

[59] TADEO, J., «México: La realidad del transporte público en Ciudad de México», recurso libre, disponible en http://es.globalvoicesonline.org/2011/04/15/mexico-la-realidad-del-transporte-publico-en-ciudad-de-mexico/, 2011.

[60] TRANSANTIAGO, «Transantiago», recurso libre, disponible en http://transantiago.cl/, 2012.

[61] WIKIPEDIA, «Algoritmo de búsqueda A*», recurso libre, disponible en http://es.wikipedia.org/wiki/Algoritmo_de_b2012.

[62] WINSTON, W. L., *Investigación de operaciones. Aplicaciones y algoritmos*, cuarta edición, Thomson, México, 2005.

Ficha autobiográfica

Fernando Elizalde Ramírez

Candidato para el grado de Maestro en Ciencia
con especialidad en Ingeniería de Sistemas

Universidad Autónoma de Nuevo León

Facultad de Ingeniería Mecánica y Eléctrica

Tesis:

Planificación Dinámica de Rutas del Transporte Público a Partir de los Requerimientos del Usuario

Oriundo de la Ciudad de Cerro Azul, Veracruz, desde pequeño me acostumbre a la vida del campo, donde aprendí el valor de las cosas, del respeto hacía mí, hacía las personas, los animales, inculcado en valores y en trabajar duro para alcanzar u obtener lo que deseo midiendo las consecuencias hacía los demás y no solo para beneficio propio.

De igual modo desde pequeña edad empece a mostrar inclinación a la ciencia a lo que me llevo a concursos de Física, Ciencias básicas (Física, Química, Matemáticás), Creatividad, y en Ingeniería Industrial, por mencionar algunos, obteniendo excelentes resultados en tales competiciones junto a mis amigos.

Siguiendo con mi historial académico en el año 2005 recibí el título de Técnico Operador de Microcomputadoras con Diseño Gráfico por parte del Instituto de Computación de Cerro Azul, para el 2006 recibo la carta de pasante de Técnico en Electrónica del CBTis 30. Para el 2009 realizo un diplomado en Seguridad Industrial Y Normatividad Ambiental En Instalaciones Petroleras en el Centro De Estudios Universitarios Del Norte De Veracruz para ese entonces ya había iniciado la licenciatura la cual recibí el título de Ingeniero Industrial por parte del Instituto Tecnológico de Cerro Azul en el 2010.

Durante la licenciatura a aparte de haber participado en varios concursos, asistí a algunos congresos, además de participar en las Semana Nacional de Ciencia y Tecnología del 2006 y 2009, como ponente e instructor según el orden, además de participar de forma directa e indirecta en varios proyectos, teniendo mayor relevancia el proyecto nombrado Alucompactador 4.0, donde participe de forma directa en el dieño mecánico y en la automatización del mismo, el cual usamos para un concurso y que después nos valío para titularnos de la licenciatura.

Para el 2010 logre obtener una beca por parte de la Academia Mexicana de Ciencias para participar en el Verano de Investigación Científica en el 2010, siendo asesorado por la Dra. Yasmín Ríos Solís, en el tema de investigación Planeación Justo a Tiempo Mediante Reformulaciones Convexas.

Ya titulado de la licenciatura, en el segundo semestre del 2010 mi inquietud por la ciencia, la insistencia de la Dra. Ríos Solís, y lo vivido durante ese verano, me decidí a ingresar a la maestría en el Posgrado en Ingeniría de Sistemas, lo cual ocurrió en el 2011, durante este tiempo he impartido cuatro cursos en diversos eventos en compa nía de un par de mis compa´ neras, además de haber asistido como ponente y presentador de cartel en diversos congresos en el país, en uno de ellos fui acreedor al premio del mejor cartel, pero la mejor experiencia es haber fungido como asesor de cuatro chicos en el proyecto Generación de Planes de Viaje Utilizando Cloud Computing en marco al Verano Científico de la Investigación 2012. Durante esta estancia en la Ciudad de Monterrey, logre la certificación en Personal Software

Process (Psp), donde obtuve un reconocimiento por alto desempeño.

Mi experiencia laboral se ve reducida a 5 años en el Instituto de Computación de Cerro Azul, donde me desempeñe en el área de mercadotecnia, supervición, docencia, entre otras. Para poder llevar a cabo mis estudios tuve que trabajar duro para poder ayudar a mi madre pagarlos, y me gusta escribir poemas cosa que hago desde pequeño.