PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ

FACULTAD DE CIENCIAS E INGENIERÍA

IMPLEMENTACIÓN DE UN ALGORITMO RECOCIDO SIMULADO PARA EL DISEÑO DE RUTAS DE TRANSPORTE PÚBLICO PARA LIMA CENTRO

Tesis para optar el Título de Ingeniero Informático que presenta el bachiller:

Walter Alonso Barrantes Ríos

ASESOR: Ing. Gissella María Bejarano Nicho

Lima, febrero de 2015

RESUMEN

El sistema de transporte público en muchos países es un servicio muy utilizado por personas que desean movilizarse. Sin embargo, un servicio tan utilizado por lo general tiene problemas que causan deficiencias tales como el tiempo de viaje de alto, la cobertura ineficiente de la demanda de transporte y alta congestión de tráfico. Tener una red de rutas de transporte eficaz, que permita a las personas a movilizarse rápidamente y cubra tanta demanda como sea posible de manera eficiente, es muy importante para la estructura social y económica de las áreas urbanas.

El problema de Diseño de Rutas de Tránsito es un problema NP-difícil, el cual se enfoca en "la optimización de una serie de objetivos que representa la eficiencia de la red de transporte público bajo limitaciones operativas y de recursos" (Guihare, Hao 2008). Básicamente, el diseñar una red de rutas de transporte público intenta minimizar (o maximizar) algún tema objetivo deseado bajo diferentes restricciones, que reflejan los requisitos de rendimiento del sistema y / o limitación de recursos (Fan, Machemehl 2006).

Contenido

Índice de Figuras

Índice de Pseudocódigos

Índice de Tablas

Índice de Ecuaciones

Capítulo 1 Planteamiento del proyecto de fin de carrera

1 Problemática

El informe final de la Encuesta de Recolección de Información Básica del Transporte Urbano en el Área Metropolitana de Lima y Callao indican que la participación modal del transporte público es de aproximadamente el 70% del total de viajes realizados en Lima Metropolitana (JICA, 2013). Sin embargo, esto no quiere decir que las personas estén satisfechas con este servicio de transporte, Claudia Bielich (2009) indicó que el 46% de los limeños que participaron en una encuesta realizada por el Instituto de Opinión Pública de la Pontifica Universidad Católica del Perú en abril de 2008 consideran al caos y la mala calidad del transporte público como el segundo mayor problema de Lima.

Para poder entender las causas de dicho problema es necesario hacer una revisión a la "evolución" del sistema de transportes detallado por Claudia Bielich (2009) en el informe La Guerra del Centavo: El transporte informal apareció en los años 30 debido a una gran cantidad de demanda insatisfecha, pero en dicho momento el Estado aplicó una serie de normativas para controlar dicha informalidad. Sin embargo, en los años 50 las empresas formales de transporte entraron en crisis y la informalidad fue adquiriendo mayor presencia y esto continuó aumentando a pesar de los cambios que realizaba el Estado (intentaron promover el transporte formal e incluso formaron su propia línea de transporte público, pero no dio resultado). El panorama continuó de la misma manera hasta el año 1991, cuando se dicta el Decreto Legislativo 651 y el Decreto Supremo 080-91-F.

- DL 651: Este decreto daba una gran libertad para poder realizar el servicio de transporte público otorgando la libre competencia de las tarifas, permitiendo a cualquier persona, ya sea natural o jurídica, brindar este servicio. Asimismo, establecía que cualquier vehículo, salvo los camiones y los vehículos de dos ruedas, pudiese ser utilizado como medio de transporte público. Finalmente, otorgaba un libre acceso a las rutas, esto quiere decir que cada prestador del servicio podía escoger la ruta a utilizar e incluso generar nuevas (Decreto Legislativo 651, 1991).

- DS 080-91-F: Este decreto elimina completamente cualquier restricción en la exportación e importación de vehículos, permitiendo la importación de vehículos de segunda mano. A raíz de esto, la compra de autos usados para realizar transporte público aumentó de manera significativa (Decreto Supremo 080-91-F, 1991).

Lo que ocasionó la promulgación de dichos decretos fue que la sobredemanda de transporte público dejó de existir, pero surgió un nuevo problema que hasta la actualidad persiste: la sobreoferta del mencionado servicio. Además, “El DL 651, entre otras cosas, supuso la legalización de la informalidad” (Bielich, 2009).

Claudia Bielich (2009) señala como una de las causas principales que provocan el desorden vehicular, a la denominada “Guerra del Centavo”. Este término hace referencia a la lucha que realizan los choferes y cobradores por los pasajeros. Los choferes reciben sus ingresos dependiendo de las ganancias que obtengan en un día, la falta de un sueldo mensual con monto fijo conlleva a que conduzcan de manera imprudente e irresponsable, donde su objetivo principal es obtener la mayor cantidad de pasajeros posibles sin importarles la seguridad de estos (Bielich, 2009).

Otra de las causas principales es la desorganización de las rutas y paraderos existentes. Al mencionar que el sistema de rutas y paraderos está desordenado, se hace referencia a que existen una gran cantidad de rutas y una gran cantidad de paraderos, los cuales, no están organizados de manera óptima para poder cubrir la demanda y evitar, en lo posible, un alto congestionamiento vehicular (Bielich, 2009). En Lima Metropolitana existe una sobreoferta, pero si esta no está distribuida de una manera óptima no podrá cubrir eficientemente la demanda.

Entre las consecuencias producto del caos del transporte público se encuentra la existencia de una alta congestión vehicular en la ciudad de Lima. Además, los tiempos de viaje de las personas suelen ser muy altos principalmente para las zonas alejadas, el tiempo de viaje del transporte público es 1.8 veces más que el del privado (JICA, 2013), esto genera un descontento y la idea de que el servicio es de mala calidad.

Por lo tanto, una de las maneras en las que se podría contribuir a reducir el caos vehicular existente es diseñando un conjunto de rutas que logre cubrir eficientemente la demanda de transporte en el mínimo tiempo de viaje posible. En ese contexto, el proyecto de investigación “Diseño y desarrollo de un algoritmo de reorganización de rutas de transporte público” financiado por el FINCYT, plantea implementar algoritmos meta-heurísticos para diseñar rutas de transporte público. El presente proyecto de fin de carrera es parte de dicho proyecto de investigación y contribuye con su logro.

El problema de generar rutas de transporte público se define como el problema de Diseño de Red de Rutas de Tránsito de Buses, el cual es un problema es uno muy complejo que se divide en los subproblemas diseño de rutas de transporte, establecimiento de frecuencias de salida de buses, planificación de tiempos de salida y llegada de los buses, desarrollo de horarios y asignación de buses y choferes (Fan, B. Machemehl 2004). El proyecto de investigación “Diseño y desarrollo de un algoritmo de reorganización de rutas de transporte público” tiene como propósito desarrollar una solución para el primer subproblema mencionado que implica el diseñar las rutas de transporte público según características y metas que se deseen cumplir, este subproblema también es conocido como el problema de Diseño de Rutas de Tránsito. En el proyecto de investigación se definieron implementar algoritmos meta-heurísticos que resuelvan este problema, siendo uno de ellos el Recocido Simulado. Implementar este algoritmo es el objetivo asignado del presente proyecto de fin de carrera enfocándose en resolver el problema de Diseño de Rutas de Tránsito. Se busca que este conjunto de rutas a diseñar sea lo más óptimo posible en términos de reducir el tiempo de viaje y cubrir la mayor cantidad de demanda

La reforma de transporte impulsada por la Municipalidad Metropolitana de Lima plantea un sistema de corredores y rutas alimentadoras como propuesta para solucionar el problema vehicular, donde el Metropolitano es un elemento clave de ese Sistema. Las rutas a generar se pretenden que sirvan como una alternativa distinta o complemento a la de los corredores.

El proyecto se aplicará a los distritos incluidos en el área de Lima Centro debido a que esta concentra la mayor cantidad de demanda y se produce la mayor proporción de viajes dentro del área en comparación con otras zonas de Lima (JICA et al. , 2005). De esta manera se pretende abordar uno de los factores que generan el caos vehicular existente.

1.1 Objetivo general

El objetivo general del proyecto es implementar un algoritmo meta-heurístico Recocido Simulado que genere un diseño de rutas de un área de Lima Metropolitana para disminuir el tiempo de viaje total y la demanda insatisfecha.

1.2 Objetivos específicos

Los objetivos específicos son:

1.2.1 Establecer las estructuras de almacenamiento de la información y estructuras del algoritmo.

1.2.2 Implementar un algoritmo Recocido Simulado que genere un diseño de rutas para disminuir el tiempo de viaje total y la demanda insatisfecha.

1.2.3 Realizar la experimentación numérica para comparar los resultados entre el algoritmo Recocido Simulado y el algoritmo PIA.

1.2.4 Implementar una aplicación donde se ejecutará el algoritmo y se muestren los resultados.

1.3 Resultados esperados

Los resultados esperados por cada objetivo específico son:

1.3.1 Resultado esperado 1 para el objetivo número 1 (R1O1): Formato de almacenamiento de la demanda de transporte por zona, vías transitables y solución. Diseño de base de datos para almacenar la información recolectada.

1.3.2 Resultado esperado 2 para el objetivo número 1 (R2O1): Diseño de las estructuras de datos y de la solución del algoritmo. Formulación del Problema

1.3.3 Resultado esperado 3 para el objetivo número 2 (R3O2): Pseudocódigo del algoritmo Recocido Simulado, que genere un diseño rutas para el área a considerar.

1.3.4 Resultado esperado 4 para el objetivo número 3 (R4O3): Documentación, diseño y resultados de la experimentación numérica.

1.3.5 Resultado esperado 5 para el objetivo número 4 (R5O4): Aplicación, la cual mostrará el conjunto de rutas resultado.

2 Herramientas, métodos, metodologías y procedimientos

En la Tabla 1 se mapea las herramientas a utilizarse por cada resultado esperado.

Resultados esperado	**Herramientas a usarse**
R1O1: Formato de almacenamiento de la demanda de transporte por zona, vías transitables y solución.	Osmosis-OpenStreetMap Postguis PgAdmin III
R3O2: Diseño e implementación de un algoritmo Recocido Simulado que genere un diseño de rutas para el área a considerar.	Code::Blocks
R4O3: Documentación, diseño y resultados de la experimentación numérica.	Microsoft Excel Metodología de Prueba de Hipótesis Método de Comparación de Múltiples Tratamientos
R5O4: Aplicación, la cual mostrará el conjunto de rutas resultado.	API de Google Maps

Tabla 1 Herramientas a utilizar

2.1 Herramientas

PgAdmin III: Entorno de escritorio visual que facilita la gestión y administración de base de datos PostgreSQL, ya sea mediante instrucciones SQL o haciendo uso de un entorno gráfico. Es instalable en plataformas Linux, FreeBSD, Solaris, Mac OSX y Windows (PgAdmin III, 2014). Esta herramienta ayudara a la administracion de la base de datos planteada en el resultado número 1

Postguis: Es un módulo que añade soporte de objetos geográficos a la base de datos objeto-relacional PostgreSQL, convirtiendo a esta en una base de datos espacial para permitir utilizarla en sistemas de información geográfica y esta

licecenciado bajo la licencia pública general de GNU (Postguis, 2014). Con Postguis es posible trabajar con objetos geográficos en la base de datos diseñada en el proyecto.

Code::Blocks: Es un entorno de desarrollo integrado libre y multiplataforma para el desarrollo de programas en lenguaje C y C++. Se puede utilizar libremente en diversos sistemas operativos y está licenciado bajo la licencia pública general de GNU (Code::Blocks, 2014). Este entorno será utilizado para desarrollar el algoritmo.

OpenStreetMap (OSM): Es una aplicación que permite visualizar mapas, que ha sido creado utilizando información geográfica capturada con dispositivos GPS móviles, y otras fuente móviles. La aplicación distribuye bajo licencia abierta, la cartografía, las imágenes creadas y los datos vectoriales guardados en su base de datos (OpenStreetMap, 2013). Esta herramienta ayudará a levantar el mapa del área establecida.

Osmosis: Es un comando Java para el procesamiento de datos de OSM. La herramienta consiste de componentes anexables que pueden ser encadenados para realizar una operación más compleja. Uno de sus componentes es el de escritura/lectura de base de datos (OpenStreetMap, 2013). Gracias a este comando es posible procesar la estructura vial de un área deseada y almacenarla en una base de datos.

API de Google Maps: Es una interfaz de programación de aplicaciones que permite a desarrolladores integrar Google Maps en aplicaciones web. Esta API es de uso gratuito y permitirá mostrar gráficamente las rutas generadas por el algoritmo en la aplicación.

2.2 Métodos y Procedimientos

Método de Comparación de Múltiples Tratamientos: Método utilizado para comparar las diferencias entre dos o más tratamientos. Las preguntas principales que se realizan son ¿Existen diferencias significativas entre las medias de los tratamientos? y si las existen ¿Cuáles de los tratamientos son los diferentes? (Comparación de Múltiples Tratamientos, 2013).

El procedimiento inicia verificando que los tratamientos tienen una distribución normal. Al verificar la normalidad, se procede a realizar el análisis de varianza para determinar si los tratamientos son distintos y finalmente se compara las medias para determinar que tratamiento tiene una media mayor (Comparación de Múltiples Tratamientos, 2013). En el proyecto los tratamientos estarían compuestas por los valores de las funciones objetivos obtenidas de cada conjunto de corridas de un algoritmo.

Siguiendo el método de Múltiples Tratamientos, se deben seguir 3 pasos durante la experimentación las pruebas utilizadas en cada uno de los pasos son los siguientes:

- *Prueba de Kolgomorov:* El primer paso es determinar si los resultados de los algoritmos poseen una distribución normal. Esta distribución normal también se le denomina campana de Gauss, debido que al representar su función de probabilidad tiende a parecerse a una campana. La prueba de Kolgomorov permite determinar si los resultados poseen dicha distribución, esta prueba es la más usada y se considera una de las pruebas más potentes para una cantidad de datos mayores a 30 (Curso General de Estadistica, 2014).

- *Prueba F*: Como segundo paso de la experimentación se debe determinar si existen diferencias significativas entre las varianzas de los resultados obtenidos por ambos algoritmos, para esto se realiza un análisis de varianza. La prueba F permite realizar este análisis (Berenson, Mark, David; 2006).

- *Prueba Z*: Al determinar que los datos poseen una distribución normal y poseen varianzas distintas el último paso es determinar cuál algoritmo posee los mejores resultados. Esta prueba es preferible usarla para cantidades de datos mayores a 30, a diferencia de la prueba T-Student que se usa para cantidades de datos menores a 30. Usando la prueba Z se evalúa las medias de las muestras y se determina cuál de las dos posee una media mayor (Berenson, Mark, David; 2006).

2.3 Metodologías

Contraste de Hipótesis: Un contraste estadístico de hipótesis consiste en determinar si una hipótesis o teoría formulada sobre una población deba ser rechazada o no deba serlo, evaluando la población seleccionada a través de una muestra aleatoria de la misma (Ruiz, Falcó 2009).

Falcó (2009) explica que un contraste de hipótesis requiere el establecer una hipótesis previa (hipótesis nula Ho) y compararla con una hipótesis alternativa (H1). Al establecer las hipótesis relacionadas a un parámetro de la población, se requiere encontrar un estadístico pivote que esté relacionado al parámetro, con ese estadístico obtendremos el criterio de decisión según el valor que alcance la muestra. Además es necesario determinar el porcentaje de riesgo de equivocarse (nivel de confianza), rechazando una hipótesis cuando es cierta, que se está dispuesto a asumir, con ese valor se logra determinar el valor crítico (frontera que marca el cambio de decisión). Al observar la región en la que se centra el valor de la muestra respecto al valor crítico, se determina si se acepta o rechazan las hipótesis.

3 Alcance

Como se ha mencionado, el proyecto de fin de carrera forma parte del proyecto de investigación “Diseño y desarrollo de un algoritmo de reorganización de rutas de transporte público”, el cual plantea una solución técnica al problema del transporte público, realizando un algoritmo de inteligencia artificial para optimizar el diseño de rutas para los buses del transporte público. Ambos proyectos se enfocan plenamente en diseñar las rutas de transporte público considerando la demanda de transporte (para el año 2012) y el tiempo de viaje. Tales rutas serán conformadas por las zonas del área de Lima a partir de las cuales se levantó la información de demanda, como se explica en el Capítulo 3 Sección 1. Dicha demanda de transporte es perteneciente al periodo entre las 7:00 am y 8:00 am horas y se asume que si la ruta establecida cubre la demanda de ida entre las zonas A-B, también cubre la demanda de vuelta B-A.

No se incluirá en este proyecto de fin de carrera el establecimiento de frecuencias de salida de buses, planificación de tiempos de salida y llegada de los buses, desarrollo de

horarios y asignación de buses y choferes, ya que no corresponden al objetivo establecido en el proyecto de investigación “Diseño y desarrollo de un algoritmo de reorganización de rutas de transporte público”.

3.1 Limitaciones

El proyecto es de investigación aplicada, ya que pretende implementar la solución a un área real de Lima Metropolitana, la determinada por Lima Centro. Dada la gran cantidad de datos e innumerables soluciones posibles, el problema que se pretende resolver es de Optimización Combinatoria y el área del proyecto pertenece al de Ciencias de la Computación, específicamente a la rama de Inteligencia Artificial.

En el proyecto se puede identificar como obstáculo a la existencia de varias entidades, entre ellas el Ministerio de Transporte y Comunicaciones, la Gerencia de Transporte Urbano, Protransporte (estas dos últimas pertenecientes a la Municipalidad de Lima), por mencionar algunas, que están a cargo de la gestión del transporte; sin embargo realizan esta gestión por separado e incluso la información que poseen no está integrada y por tanto no tienen la misma versión de archivos o documentos. También existe la dificultad de acceder a la información que estas entidades poseen. Las solicitudes de información suelen demorar en ser atendidas. Al comunicarse por las vías telefónicas suelen trasladar la llamada a distintas áreas lo que demuestra un limitado acceso a su propia información. Además, el conocer las demandas de transporte entre las diferentes zonas de demanda, implica solicitar esa información a las entidades encargadas de la gestión y organización del transporte público, es posible que no se logre obtener los datos actualizados de tal información o que no pueda ser entregada a tiempo. Si ese es el caso se deberá crear los datos de demanda aleatoriamente o realizar encuestas a los habitantes de las zonas para conseguirla; sin embargo, las personas generalmente muestran rechazo al dedicar su tiempo a contestar a las encuestas e incluso pueden contestar sin ser del todo sinceros.

3.2 Riesgos

En la Tabla 2 se muestran los riesgos identificados del proyecto de tesis.

Riesgo identificado	Impacto en el proyecto	Medidas correctivas para mitigar
Falta de acceso a la información y documentos del proyecto.	Se tendrá que recopilar la información y realizar cada documento nuevamente.	Realizar back ups en la nube y dispositivos de almacenamiento externos (google drive, one drive).
Las personas a encuestar se rehúsan a participar en las encuestas.	No se logra obtener información real sobre la demanda de transporte.	Generar datos ficticios sobre la demanda de transporte. Tomar datos del informe Plan Maestro Lima-Callao
Las municipalidades no proveen la información requerida sobre sus distritos.	No se logra levantar la estructura del área delimitada para el proyecto.	Generar una estructura ficticia del área a utilizar por el algoritmo. Generar una estructura real del área a utilizar, según mapas actuales de avenidas y calles que permiten el transporte público.
Los datos de información de demanda encontrados están distribuidos para zonas de área distinta a las definidas en el proyecto.	No se logra cargar la información de demanda para las zonas definidas.	Distribuir o realizar una estimación de la demanda obtenida a las zonas definidas en el proyecto.

Tabla 2 Riesgos del Proyecto de fin de carrera

4 Justificativa

Realizar el siguiente proyecto es conveniente, debido a que pretende apoyar en la organización del sistema de transporte, generando un sistema de ruta, el cual cubra la mayor cantidad de demanda y disminuya el tiempo de viaje. Esto traería beneficios a la Municipalidad de Lima, ya que podrían utilizar esta solución para apoyarse en el trabajo continuo de mejorar el transito vial en los distritos y disminuir la congestión vehicular existente. Además los tiempos de viaje podrían reducirse, lo cual beneficiaría a los usuarios del servicio de transporte público. La base de datos a construir puede ser

utilizada para futuras investigaciones y facilitar la gestión y control de las rutas de transporte en tareas sencillas como habilitación de rutas, seguimiento de rutas por vía, entre otras. Así mismo, el algoritmo se puede reutilizar para diseñar rutas de transporte alimentadoras como las que usa el sistema Metropolitano o el diseño de rutas en otras ciudades modificando los datos de entrada.

El proyecto pretende resolver el problema de Diseño de Rutas de Transito el cual es un problema de naturaleza NP y por esa razón se propone una solución algorítmica meta-heurística. El algoritmo Recocido Simulado ha sido mayormente utilizado en los otros subproblemas que forman parte del problema de Diseño de Red de Rutas de Tránsito de Buses, como se mencionó en el Capítulo 1 Sección 1 y que se centran en la planificación y establecimiento de frecuencias de partida de los buses. Los escenarios donde han sido aplicados poseen una red de rutas de transporte bien diseñada y/o establecida y la meta definida fue optimizar el servicio de transporte que usa tal red de rutas estableciendo mejores frecuencias de salida o planificando las operaciones con los buses. Sin embargo, tal caso no parece ser el de Lima Metropolitana al generarse largos viajes para las personas que desean llegar a su destino y tener una gran cantidad de rutas que aun así no logan cubrir eficientemente la demanda. Es importante desde un principio poseer un conjunto de rutas de transporte público bien diseñado que intente reducir tales problemas. Por tales motivos en este proyecto de fin de carrera se establece utilizar al algoritmo Recocido Simulado para aplicarse en este proyecto de fin de carrera con el objetivo de disminuir el tiempo de viaje total y la demanda insatisfecha.

5 Viabilidad

El desarrollo del proyecto de fin de carrera se realiza como parte del proyecto de investigación “Diseño y Desarrollo de un Algoritmo de Reorganización de Rutas de Transporte Público” financiado por FINCYT. La información sobre las rutas, red vial y paraderos solicitada a la Municipalidad de Lima y la información de la demanda de transporte serán recopiladas en el lapso de tiempo correspondiente al periodo de enero-marzo del 2014. Además se utilizará la herramienta OpenStreetMap, disponible en la web, para la carga de la estructura vial del área establecida a la base de datos.

Existe abundante material bibliográfico sobre los algoritmos de inteligencia artificial y artículos sobre propuestas antes realizadas que sirven de orientación a resolver el problema. Estas proveen información sobre las consideraciones, restricciones y variables que se pueden tener en cuenta, además del procedimiento de la solución.

Las herramientas a utilizar como OpenStreetMap, Code::Blocks, la API de Google Maps y PgAdmin III se pueden adquirir gratuita y legalmente, mientras que para Microsoft Excel se posee una licencia. Además se tiene experiencia utilizando las herramientas Code::Blocks y PgAdmin III, en cuanto al resto de herramientas existe una amplia documentación para aprender a utilizarlas.

Capítulo 2 Conceptos y Estado del arte

1 Marco conceptual

A continuación se presentarán los conceptos más importantes relacionados con la problemática y solución del problema de diseño de rutas de transporte público en Lima con los que se trabaja en este proyecto de fin de carrera.

1.1 Conceptos del Problema

Los conceptos relevantes son:

1.1.1 *Sistema de rutas*

La Municipalidad Metropolitana de Lima (2005) define al sistema de rutas como el conjunto de rutas con detalles de fichas técnicas y condiciones específicas de operación estructurada con base en las líneas estratégicas establecidas en el Plan Regulador de Rutas. Cada ruta tiene asociado paraderos y por cada ruta circulan diversas líneas de buses.

Se puede definir a los paraderos como los lugares en donde las personas toman el transporte público para dirigirse a sus destinos (Municipalidad Metropolitana de Lima, 2005).

Una línea de buses es el conjunto de aquellos que pertenecen a una misma empresa y que, a su vez, recorren una misma ruta. Puede existir más de una línea de buses que recorra la misma ruta (Municipalidad Metropolitana de Lima, 2005).

En los sistemas de rutas se pueden presentar deficiencias como la congestión vehicular. Se puede definir a la congestión vehicular como la saturación del flujo vehicular, esta condición se produce por una sobredemanda en rutas de transporte (existen demasiados vehículos y la cantidad de rutas no se da abasto) y como consecuencia de ello se produce un aumento en el tiempo de viaje promedio por persona.

1.1.2 *Demanda de Transporte*

Se le denomina demanda de transporte al término que indica la cantidad de personas desean moverse de un determinado punto a otro (Municipalidad Metropolitana de Lima, 2005).

1.1.3 *Zona de Demanda*

Se toma como zona de demanda a una región delimitada por una zona de tránsito, en las cuales se indicarán valores de demanda de transporte para distintos puntos de Lima Metropolitana u otras zonas de demandas (JICA, 2013).

1.1.4 *Red Vial*

Conformada por las carreteras que constituyen la red vial circunscrita al ámbito local, cuya función es articular las capitales de provincia con capitales de distrito, éstos entre sí, con centros poblados o zonas de influencia local y con las redes viales nacional y departamental o regional (Ministerio de Transportes y Comunicaciones, 2007).

1.2 Conceptos de la Solución

Los conceptos de la solución relevantes son:

1.2.1 *Well Known Text (WKT)*

La representación WKT es una sintaxis estandarizada para describir objetos espaciales expresados en forma vectorial. Su especificación fue aprobada por el organismo internacional Open GIS Consortium (OGC, 1999). Entre los objetos que el formato describe se encuentran:

Polígono (Polygon): Un polígono es una superficie plana, definido por un límite exterior o uno o más límites interiores. Es una figura geométrica simple.

Línea (LineString): Es una curva con interpolación lineal entre puntos. Cada par consecutivo de puntos define un segmento de línea.

1.2.2 *Grafo de zonas:*

Representa El conjunto de zonas que pertenecen al área de estudio del proyecto y las conexiones que tienen entre ellos representados por aristas.

1.2.3 *Optimización Combinatoria*

La Optimización Combinatoria es una rama de ciencias de las matemáticas que plantea la construcción de métodos que permita realizar combinaciones y ordenamientos a grupos de elementos, es decir alteraciones de las posiciones de los elementos dentro del universo (Papadimitriou, 1982). Eugene L. Lawler (1976) define dicho concepto como: “El estudio matemático de una disposición, agrupamiento, o una selección de objetos discretos, usualmente finitos en número”. En la optimización combinatoria lo más importante es encontrar la disposición o solución más óptima en un conjunto de soluciones. Los problemas que se pueden resolver por la optimización combinatoria pueden clasificarse por su nivel de dificultad en L, NL, P, NP, P-Completo, NP-difícil. Estos problemas tienen un gran dominio de variables o el espacio de búsqueda muy grande y hallar la solución más óptima se vuelve computacionalmente imposible (Lawler, 1976). Sin embargo, es posible encontrar una solución lo suficientemente cercana a la óptima en esos casos reduciendo el espacio (conjunto de soluciones) de búsqueda y explorando este de manera eficiente, lo cual se puede lograr gracias a los algoritmos heurísticos y meta-heurísticos.

1.2.4 *Problemas NP*

Se definen como problemas No Polinomiales (de ahí las siglas NP) a aquellos para los que no es posible encontrar la solución óptima en un tiempo polinomial (debido a la complejidad del mismo).

Los problemas NP son de decisión y conforman el conjunto de problemas para los cuales existe un algoritmo no determinístico, el cual puede verificar que la solución es correcta en un tiempo polinomial. Un algoritmo no determinístico se conforma de dos fases. La fase de adivinación que genera una solución, la cual puede o no estar en el formato correcto del problema y la fase de verificación que determina si esa solución se encuentra en el formato correcto. Debido a la complejidad de estos problemas se opta por escoger de entre un conjunto de soluciones, la solución que se acerque más a la óptima, de esta manera se resuelve el problema en un tiempo eficiente (Alsuwaiyel, 1998).

Papadimitriou (1982) divide y define a los problemas NP en 3 tipos diferentes: NP, NP-hard, NP – Completo.

Problemas NP: Problemas cuya solución hasta la fecha no han podido ser resueltos de manera exacta por medio de algoritmos deterministas y cuya solución son de complejidad no-deterministas y puede ser verificada en tiempo polinomial.

Problemas NP completo: Estos problemas son todos “iguales” en el sentido en que si se descubriera una solución *S* para alguno de ellos, esta solución sería fácilmente aplicable a todos ellos. Cumplen dos condiciones:

1. Es un problema NP
2. Todo problema NP se puede transformar polinomialmente en él.

Problemas NP-difícil*:* Un problema que satisface la segunda condición pero no la primera pertenece a la clase NP- difícil.

1.2.5 ***Problema de Diseño de Red de Rutas de Transito de Buses (Bus Transit Route Network Design Problem)***

El problema BTRNDP (por sus siglas en inglés de Bus Transit Route Network Design Problem) involucra determinar un diseño de rutas asociados a frecuencias y planificación de horarios para buses y conductores, y lograrlo de la manera más óptima buscando minimizar (o maximizar) un determinado objetivo, por ejemplo: Minimizar el tiempo de viaje.

Este problema es muy complejo, incluso sus subproblemas separados son considerados NP-difícil desde una perspectiva de complejidad computacional (Guihare, Hao 2008).

De acuerdo con Ceder y Wilson el BTRNDP se puede componer de los siguientes partes: Diseño de rutas, establecimiento de frecuencias, desarrollo de horarios, planificación de buses y choferes (apud. Fan, B. Machemehl 2004).

1.2.6 ***Problema de Diseño de Rutas de Transito (Transit Route Network Design Problem)***

TRNDP (por sus siglas en inglés de Transit Route Network Design Problem) es un sub-problema del BTRNDP que se centra en el diseño de las rutas. Consiste en determinar un conjunto de rutas, cada una formada por 2 terminales, tomando en cuenta los siguientes criterios: una distribución de demanda de viaje, características topológicas de

área, un conjunto de objetivos y un conjunto dado de restricciones. Este problema se centra en la optimización de un número de objetivos que representan la eficiencia de las redes de transporte pública bajo restricciones operacionales y de recursos (Guihare, Hao 2008).

Fan y B. Machemehl (2004) indican que las variables y restricciones que suelen ser consideradas en este problema son muchas, entre estas se encuentran el máximo número y longitud de las rutas, restricciones en cuanto a la forma de la red de transporte; además, se pueden incluir el costo (o beneficios) del usuario y/o del operador, recursos limitados, la demanda, estrategias operacionales e incluso requerimientos legales y estándares. Además, al aplicar este problema a una red de tamaño real, es necesario determinar muchos parámetros, y se vuelve un problema de combinación NP-difícil. También indican que la demanda de transporte puede ser fija o variable: La demanda fija es constante y estática para todos los periodos de tiempo, mientras que la demanda variable puede ser afectada por la estructura y frecuencia de servicio que provee la red pública de tránsito, lo cual aumenta el grado de dificultad. Por estas razones, los métodos meta-heurísticos son los que mejor se adaptan a este tipo de casos, ya que logran lidiar con varios diseños de rutas de tránsito y encontrar un resultado razonablemente bueno (Fan, B. Machemehl 2004).

Guihare y Hao (2008) afirman que desde la visión de un usuario, toda red de rutas de tránsito debe cubrir un área extensa, ser muy accesible, ofrecer gran cantidad de rutas directas, poca necesidad de desviaciones y cumplir con la demanda total. En cambio desde un punto de vista del transportista, el sistema de rutas debe generar la mayor utilidad posible. Por esta razón el principal reto en la planificación de tránsito es encontrar un equilibrio entre estos dos objetivos.

Dependiendo de la cantidad y tipo de parámetros que se consideren y el objetivo que desea lograr se pueden identificar muchas variaciones de este problema. Entre las más representativas se puede encontrar al Transit Network Frequencies Setting Problem, en este problema se halla frecuencias de partida para cada ruta de la red, de modo que satisfaga la demanda de transporte y el Transit Network Timetabling Problem, en el cual se debe hallar tiempos de partida para cada paradero de cada ruta de la red y el tiempo de llegada esperado en el paradero final. Además, existen combinaciones de los mencionados y el TRNDP, obteniendo un problema multi-objetivo como el BTRNDP. En la

Figura 1 se muestra las características claves del problema, clasificándolas en el objetivo, parámetros a considerar y metodología de solución.

1.2.7 *Problema de Ubicación de Paraderos de Buses (Bus Terminal Location Problem)*

Ghanbari , Mahdavi-Amiri (2011) nombra al problema de asignación de paraderos de buses con el objetivo de maximizar el servicio de transporte público como BTLP por sus siglas en inglés (de Bus Terminal Location Problem). Dichos autores indican que el objetivo de este problema es localizar un número de paraderos “n” de un número “m” de terminales con vecindades alcanzables conocidas para que se maximice el servicio de transporte público, buscando maximizar la demanda satisfecha o minimizar el tiempo de viaje. BTLP es considerado como una variación del problema de Uncapacited Facility Location Problem (UFLP). Tal problema es NP-hard y tiene 2 conjuntos finitos de clientes y potenciales instalaciones, y además un costo asociado por abrir alguna instalación. El objetivo es seleccionar un sub-conjunto de potenciales instalaciones o locaciones (abrir esa instalación) y asignar cada cliente a una instalación seleccionada (abierta), tal que el costo total de apertura de instalaciones y el costo total del servicio sea mínimo. También existen variaciones donde se considera un área limitada en la cual el pasajero estaría dispuesto a tomar un bus, por tanto se añaden restricciones de distancia (Ghanbari, Mahdavi-Amiri; 2011).

1.2.8 *Recocido Simulado (Simulated Annealing)*

El algoritmo Recocido Simulado tiene los orígenes de su lógica en la termodinámica, proviene del proceso de calentamiento y consecuente proceso de enfriamiento (recocido) lento de una sustancia hasta volverla sólida (Kirkpatrick, Gelatt, Vecchi;1983). Fan y B. Machemehl (2004) mencionan que este algoritmo es muy usado para resolver problemas de optimización combinatoria y se puede considerar como una variación randómica de la búsqueda local, pero en un nivel más avanzado, ya que intenta minimizar la probabilidad de quedarse atrapado en un óptimo local de baja calidad. Los autores explican: (i) los algoritmos de búsqueda local parten de una solución inicial que va siendo modificada al realizar cambios (movimientos) en valores de las variables o realizar intercambios entre los valores de las variables que conforman la solución. (ii) Si este cambio logra una mejor solución que la actual, esta es sustituida por la recientemente encontrada, el proceso se repite hasta que no sea posible encontrar mejores soluciones.

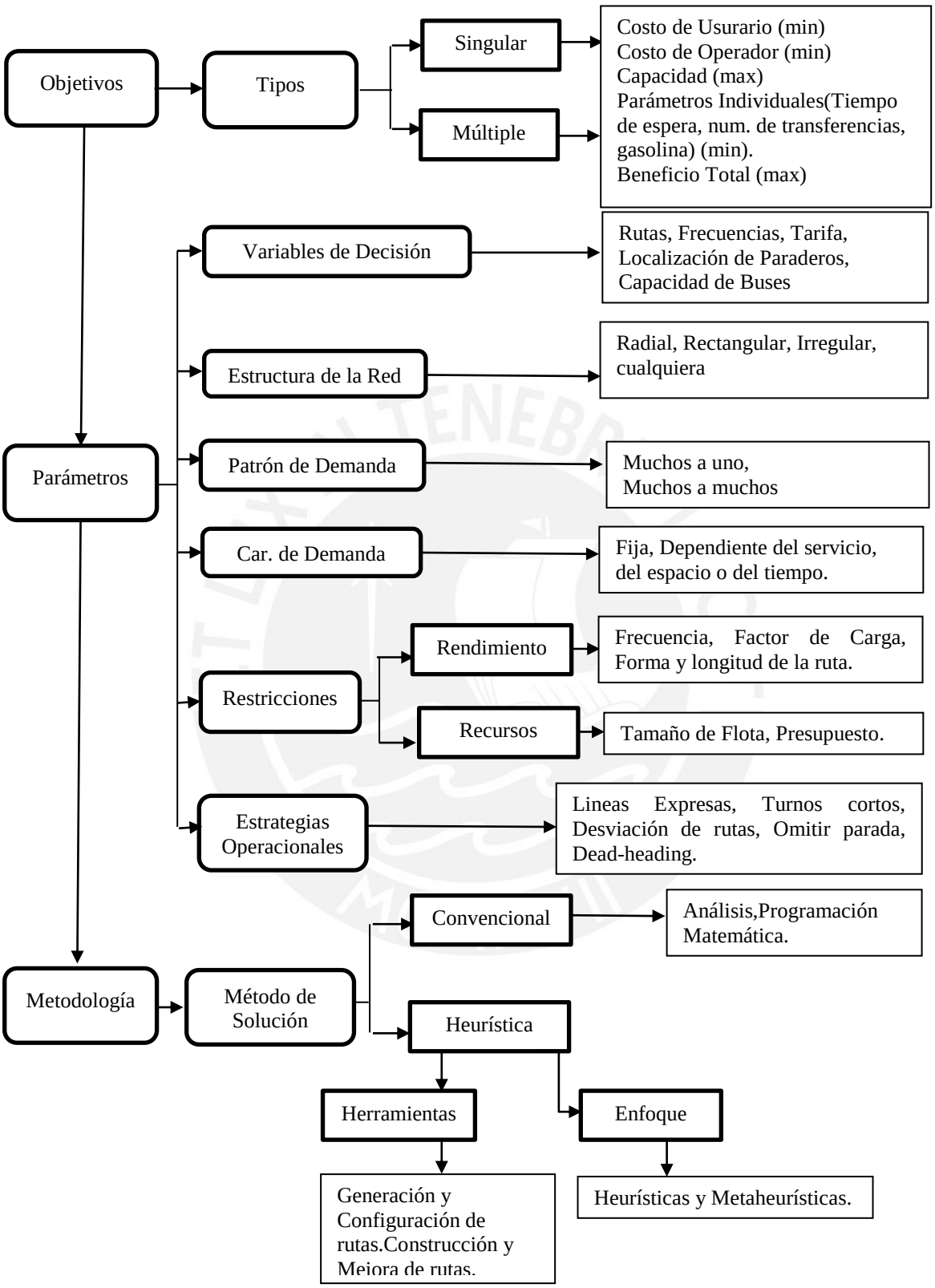


Figura 1 Características claves del TRNDP. Imagen adaptada de Konstantinos Kepaptsoglou - Matthew Karlaftis (2009)

(iii) Sin embargo esto significa que el proceso termina al encontrar el óptimo local, el cual no necesariamente es el óptimo global.Para reducir la probabilidad de que ocurra este problema, el algoritmo Recocido Simulado permite realizar algunos movimientos que generan soluciones peores, pero en caso la búsqueda esté logrando mejores resultados, estos movimientos se deben realizar de un modo controlado.(iv) El algoritmo Recocido Simulado controla estos movimientos con una función de probabilidad, la cual disminuirá la probabilidad de realizar los movimientos hacia soluciones peores conforme la búsqueda avanza (Fan, B. Machemehl 2004).

Karthryn A. Dowsland, Berlamino (2003) explican al algoritmo Recocido Simulado y cómo imita todo el procedimiento de calentamiento y enfriamiento de la siguiente manera: Empieza con una solución inicial, la cual se obtiene mediante el uso de otro algoritmo que generalmente construye la solución de manera aleatoria. Esta solución tiene asociado un costo o valor. Al comienzo del proceso se posee una temperatura alta T, que se interpreta con una mayor probabilidad de frecuencia de cambio de estado o solución. La frecuencia de cambio es usada para evitar centrarse en la búsqueda local, pero mientras ese valor sea alto también aumenta la probabilidad de encontrar una solución de baja calidad. El algoritmo selecciona un nuevo estado o solución realizando aleatoriamente un movimiento; si este cambio da un nuevo estado con un mejor valor, este estado es seleccionado. Si el nuevo valor es de menor calidad que el actual, tiene una probabilidad de aceptación que depende de la temperatura actual y los valores de la solución actual y la nueva hallada. La función que determina la probabilidad de aceptación de las nuevas soluciones se llama el criterio de Metrópolis y se define por:

$$P(\partial E) = \exp(-\partial E/t) \quad \text{donde:}$$

- ∂E es la diferencia entre el cambio del costo de la nueva solución y la actual.
- t es la temperatura actual.

Mientras el algoritmo siga ejecutándose la temperatura disminuye según una función de enfriamiento y se reduce la frecuencia y probabilidad de cambio. Esto quiere decir que a mayor tiempo de ejecución el algoritmo se “opondrá” a cambiar por una solución que tenga un valor de menor calidad (Fan, B. Machemehl 2004).

Karthryn A. Dowsland, Berlamino Adenso (2003) señalan que los métodos de enfriamiento que más han sido utilizadas son dos. El primero establece una velocidad de enfriamiento del tipo geométrico, gobernada por la función t -> t*α, con α < 1. Mediante la experimentación los autores concluyeron que con valores de α muy cercanos a 1, que corresponden a velocidades lentas de enfriamiento, se lograban mejores resultados. Utilizar esta función también implica establecer una adecuada cantidad de iteraciones a buscar en cada temperatura, es importante dedicar suficiente tiempo de búsqueda a temperaturas bajas. Es decir ir aumentando esa cantidad de iteraciones mientras se disminuye la temperatura

El segundo método propuesto por Lundy y Mees (1986), ejecuta una sola iteración por temperatura pero la velocidad de la reducción de la temperatura es muy lenta según la formula t -> t/(1+β), donde β tiene un valor muy pequeño. Mediante evidencia empírica y resultados teóricos se concluyó que los detalles concretos del programa de enfriamiento no son tan importantes como la velocidad de enfriamiento y que ambos métodos otorgan resultados similares para un mismo rango de temperaturas y cantidad total de iteraciones similares (apud. Karthryn A. Dowsland, Berlamino Adenso 2003).

La temperatura final es importante también, en teoría debería reducirse a 0, pero en la práctica el algoritmo encuentra su óptimo final mucho antes de que se llegue a ese valor, por lo tanto si esta temperatura final es muy baja estaremos utilizando tiempo de ejecución ineficientemente, pero si el valor fijado es muy alto no se garantiza que se encuentre ningún óptimo local. Por esta razón se sugiere detener la búsqueda cuando se llega a la temperatura final o al haberse producido un numero de iteraciones en las cuales no ha habido ninguna aceptación de una nueva solución (Karthryn A. Dowsland, Berlamino Adenso 2003).
Del Pseudocódigo se observa que la función Metrópolis es (F(j) – F(i))/ T. Una solución de menor calidad se elegirá si el valor hallado aleatoriamente entre 0 y 1 es menor al valor hallado por la función metrópolis.

Este algoritmo es muy usado y puede producir soluciones de buena calidad; sin embargo, para lograr mayor calidad en las soluciones requiere más tiempo de ejecución, lo cual se convertiría en su única desventaja. Por esta razón se requiere que la velocidad del proceso de enfriamiento sea lenta, y esta velocidad depende del valor de la función de

enfriamiento el valor del factor que utiliza la función y de las iteraciones por cada temperatura (Karthryn A. Dowsland, Berlamino Adenso 2003).

El procedimiento del algoritmo se muestra en Pseudocódigo 1:

Recocido Simulado:
Seleccionar un estado inicial i ∈ S;
Seleccionar una temperatura inicial T > 0;
Establecer contador de cambio de temperatura t = 0 ;
Repetir
Establecer contador repeticiones n=0 (número de iteraciones a realizar por cada temperatura)
Repetir
Generar estado j, vecino de i;
Calcular ∂ = f (j) – f (i);
Si ∂<0 entonces i = j;
Sino Si random(0,1) < exp (-∂/T) entonces i = j;
n = n +1;
Hasta n = N(t);
T = T(t);
Hasta que el criterio de parada se cumpla.

Pseudocódigo 1 Pseudocódigo del algoritmo Recocido Simulado. Recuperada de Wei Fan - Randy Machemehl (2004)

1.2.9 ***PIA (Pair Insertion Algorithm)***

Este algoritmo fue construido por Antonio Mauttone y María E. Urquhartpara (2009) para diseñar rutas de transporte cumpliendo con restricciones de demanda y teniendo en cuenta los intereses de parte del operador y del usuario.

En el procedimiento, el grafo a utilizar se compone por vértices que representan zonas del área a cubrir y las aristas representan conexiones lógicas entre estas zonas. Las rutas a diseñar son rutas conformadas por las zonas escogidas (Mauttone, E. Urquhartpara ; 2009). Los autores afirman que los resultados luego pueden ser llevados a la realidad tomando como referencia estas rutas para formar las rutas conformadas por calles, por esta razón la duración del viaje puede ser ligeramente distinto.

Mauttone y Urquhartpara (2009) explican que la idea principal del algoritmo es conectar pares de vértices con altos valores de demanda. Esta conexión la realizan creando una ruta directa entre ambos vértices con el camino más corto o insertando los vértices en una ruta ya existente. El procedimiento del algoritmo según los autores es el siguiente: Se

empieza con un conjunto de rutas vacío e iterativamente se escogen de una lista de pares de vértices los que tienen demanda insatisfecha. En cada iteración se escoge el par que tenga mayor demanda. Seguidamente se evalúa el costo de crear una ruta directa o de insertar los vértices seleccionados en la mejor posición de la mejor ruta posible (esto ayuda a minimizar el costo de inserción), el método que genera menor costo es el elegido y se procede a actualizar la matriz de demanda cubierta. El proceso se repite hasta cumplir con la restricción de demanda establecida. El proceso principal se muestra en el pseudocódigo 2.

Ambos autores indican que el diseño del algoritmo se basa en el pensamiento que el diseño de transporte tiene como objetivo más probable, cubrir la mayor demanda posible de la mejor manera posible con los recursos disponibles, aunque otros objetivos pueden ser tomados también. Como la demanda de transporte tiene una naturaleza origen-destino, la inserción de pares de vértices es la clave en este algoritmo. Por otro lado también se procura limitar la extensión de las rutas ya que para el operador la longitud de la ruta es directamente proporcional a la flota que requerirá y eso aumentaría sus costos. Además se compara el costo de crear una ruta directa con el costo que implica la inserción del par de vértices para siempre minimizar el tiempo de duración del viaje el usuario (Mauttone, E. Urquhartpara ; 2009).

```
PIA
Inicializar lista de rutas LR y matrices de demanda
Construir lista de pares L de origen-destino con demanda distinta de 0 ;
Mientras No se cubra la demanda mínima
        (u,v) = Seleccionar par de vértices con mayor demanda en L;
        R = Crear ruta directa entre u y v
        R' = Crear ruta insertando el par u, v en la mejor posición de la ruta existente  más
        conveniente R''
        Si costo(R) < costo(R') – costo(R'') entonces
                LR = LR U { R};
                Borrar de L pares cuya demanda haya sido cubierta por R;
        Sino    entonces
                LR = LR U { R'} – {R''}
                Borrar de L pares cuya demanda haya sido cubierta por R';
        Actualizar matrices de demanda;
Filtrar rutas;
Retornar LR;
```

Pseudocódigo 2 Estructura general del algoritmo PIA. Imagen recuperada de Antonio Mauttone y María E. Urquhartpara (2009)

Por último, explican que el algoritmo puede ser utilizado para generar soluciones iniciales para luego utilizar algoritmos de mejora local o de evolución, así como para completar una solución inviable con respecto a limitaciones de la cobertura de demanda. Los estudios experimentales realizados al algoritmo mostraron que logra obtener diversas soluciones en problemas objetivos y decisión; esta característica es muy útil si se desea utilizar este algoritmo como subrutina en otros algoritmos como meta-heurísticos, en particular en un problema multiobjetivo como es el TRNDP (Mauttone, E. Urquhartpara ; 2009).

1.2.10 ***Algoritmo DIJKSTRA***

Este algoritmo encuentra el camino más corto de un vértice inicial hacia el resto de vértices en un grafo *G = (V, E)* ponderado (cada arista tiene asociado un peso), este grafo no puede tener pesos negativos.

El algoritmo DIJKSTRA mantiene una lista *S* de vértices, cuyos caminos cortos finales desde el vértice inicial ya han sido explorados. El algoritmo repetidamente selecciona un vértice *u* que pertenece a *V-S* con el camino más corto estimado, añade *u* a *S*, relaja todas la aristas que salen de *u*. (Cormen; Leiserson; Rivest; Stein ,2009)

```
DIJKSTRA()
D← [∞, ∞, ...∞]  (η copias de ∞)// Se inicializan las distancias en infinito
D[s] ← 0// se inicializa la distancia del vertice inicial en 0
P ← [ ] //lista de padres de cada vértice
Q← V  (lista de nodos a visitar)
MIENTRAS tamaño(Q) > 0 hacer
        encontrar υ ∈ Q tal que D[υ] es mínimo
        Q←  obtener(Q; υ)//se extrae el vértice de la lista W
        para cada μ adyacente a v ∩ Q hacer
                si D[μ] > D[υ] + peso (υμ) luego
                  D[μ]  ←D[υ] + peso (υμ)
                  P[μ] ← υ
Retornar (D; P)
```

Pseudocódigo 3 Estructura del algoritmo Dijkstra.

El algoritmo da como resultado la lista *D* de manera que *D[v]* es la distancia del camino más corto de *s* a cada vértice *v* y la lista *P* de vértices de manera que *P[v]* es el padre de *v*.

2 Estado del arte

Con el objetivo de contribuir a solucionar el problema del transporte en Lima se han tomado muchas iniciativas sociales y legales como el imponer papeletas más severas, mantenimiento y ampliación de vías de transporte, el Metropolitano, entre otras. En este proyecto de fin de carrera se abordará el enfoque científico donde anteriormente se han estudiado distintos métodos para solucionar este problema presente no solo en la ciudad mencionada. Estos métodos van desde los más exactos como el matemático hasta los meta-heurísticos y algoritmos bio-inspirados, además se revisarán sistemas de información que actualmente son utilizados y cumplen con algunas funcionalidades relacionadas a la generación de rutas.

Con la revisión del estado del arte se logra que el lector obtenga conocimiento sobre los métodos que se han utilizado para resolver el problema de diseño de ruteo y los distintos enfoques considerados, así como también los sistemas de información que se pueden optar por utilizar si desea afrontar este tipo de problemas. Además, con la información expuesta se ayuda a reconocer que enfoques y métodos aún no han sido explorados o no muy a fondo en el TRNDP.

2.1 Artículos de Investigación

Se presentan los métodos de solución utilizados en los artículos de investigación revisados en el estado del arte.

2.1.1 *Enfoque Matemático*

Este enfoque hace referencia a las ramas de la matemática computacional, las cuales utilizan a las computadoras en las áreas que involucran actividades científicas y prácticas que son caracterizadas como el análisis de modelos matemáticos. Además, utilizan y proponen técnicas y algoritmos para resolver problemas matemáticos que surgen durante la investigación de estos modelos, dichos problemas son muy grandes para la capacidad numérica humana (Bradley, Magnant; 1977).

El análisis de modelos matemáticos incluye el estudio de la formulación del problema, la selección del modelo, el análisis y procesamiento de la información de entrada, la solución numérica del problema , el análisis de los resultados y finalmente los problemas que se conectan con la consecución de los resultados obtenidos (Bradley, Magnant; 1977).

Murray (2003) indica que para el diseño de una red de rutas eficiente es crucial determinar la cantidad y ubicación de los paraderos, mientras más de ellos se encuentren, más accesible será el diseño de rutas, pero menor será la velocidad de transporte. Murray estudia dos variaciones del problema. Para la primera variante utiliza el Location Set Covering Problem y el Maximal Coverage Location Problem para modelarlo y el objetivo es reubicar los paraderos para reducir la cantidad de estos en una red existente. La segunda variación busca obtener la óptima colocación de los paraderos para crear o extender la red, dado un número determinados de paraderos adicionales; con esto se logra incrementar la accesibilidad a la red de transporte. Para este problema utiliza un híbrido del Location Set Covering Problem.

Los estudios de Guan, J. F., Yang, H. y Wirasinghe, S. C. en el 2003, haciendo uso de un programa lineal binario entero, buscaron modelar en conjunto el planeamiento de las rutas y el proceso de transferencia de pasajeros, teniendo como objetivo minimizar la cantidad y longitud de rutas y la longitud total recorrida de los pasajeros.(apud. Guihaire; Hao 2008).

Dentro de este enfoque también se encuentran los modelos de programación no lineales que fueron desarrollados por Constantine y Florian 1995, Russo 1998, y por último Delle Site y Fillipi Francesco 2001, pero estos modelos no consideran las rutas como variables de decisión.(Kepaptsoglou, Karlaftis 2009).

Los modelos matemáticos logran resultados óptimos para restricciones específicas y en donde el tamaño del problema es pequeño. Sin embargo, estas formulaciones no logran representar el problema en un tamaño realista por la necesidad de variables discretas, no linealidades (para una función no lineal es extremadamente difícil predecir su comportamiento dado una variable) y la existencia de condiciones. Además, la programación matemática tiene serias limitaciones en cuanto la representación de transferencias, continuidad de la ruta, los pasajeros y el diseño de parámetros lógicas (Fan, Wei; B. Machemehl, Randy, 2004). La complejidad y dificultad en representación del problema, los extensivos requerimientos computacionales junto con la alta dificultad del problema hacen que esta metodología de resolución no sea la más adecuada.

2.1.2 *Búsqueda Exhaustiva (Exhaustive Search)*

La búsqueda exhaustiva es un método, el cual consiste en explorar todo el universo de soluciones posibles y de esta manera encontrar la mejor solución al problema (al recorrer todo el universo de búsqueda encontramos una solución global) (Fan, Wei; B. Machemehl, Randy, 2004).

La estructura de una búsqueda exhaustiva según Fan y Machemehl (2004) es la siguiente:

1. Generar una solución inicial i ∈ S
2. Evaluar la solución i y establecer la solución óptima objetivo O(i)
3. Repetir
4. Generar una nueva solución j ∈ S
5. Si O(j) < O(i), entonces i = j;
6. Hasta haber recorrido todas las soluciones del espacio de soluciones posibles.

Wei Fan y Randy B. Machemehl (2004) mencionan que si bien en principio el método de búsqueda exhaustiva siempre puede ser utilizado; si el espacio o universo de soluciones posibles es muy amplio, el tiempo que va toma realizar una exploración exhaustiva es muy extenso y los recursos que se requieren son demasiados, esto lleva a la conclusión de que, este método solo se va a poder aplicar en los problemas de diseño de rutas si la red de rutas posibles posee un tamaño pequeño. Para las aplicaciones del problema de diseño de rutas en escenarios reales, por lo general no se utiliza este método ya que las redes son de gran tamaño (una ciudad puede tener cientos de rutas posibles) (Fan, B. Machemehl 2004).

Wei Fan y Randy B. Machemehl (2004) desarrollaron este algoritmo para resolver el problema del BTRNDP (Bus Transit Route Network Design Problem). Este desarrollo se utilizó solo en redes pequeñas, ya que para redes muy grandes o complejas este método no es viable por el tiempo de ejecución, con el objetivo de tener un marco de comparación al momento de evaluar la eficiencia de otros algoritmos como Búsqueda Tabú y Algoritmo Genético al resolver el mismo problema (ya que este algoritmo arroja una solución exacta y por ello sirve como marco comparativo para las soluciones aproximadas).

2.1.3 *Búsqueda Aleatoria (Random Search)*

Este es un método de optimización que se basa en ir escogiendo para cada paso una solución de manera aleatoria del universo de soluciones posibles y luego evaluar la solución comparando el valor de la función objetivo con el valor que tenía en los pasos anteriores (Fan, Wei; B. Machemehl, Randy, 2004).

Wei Fan y Randy B. Machemehl (2004) desarrollan un algoritmo utilizando el método de Búsqueda Aleatoria para resolver el BTRNDP. El algoritmo presentado buscaba la minimización de los costos así como satisfacer la demanda establecida, este se basa en que, para cada paso en un conjunto de rutas de tamaño específico, el método escoge de manera aleatoria un conjunto solución de todo el universo de soluciones posibles y luego, se evalúa la solución comparando el valor de la función objetivo con el valor que tenía en los pasos anteriores (Fan, Wei; B. Machemehl, Randy, 2004). Se alcanza la solución óptima cuando se encuentra una solución cuyo valor de la función objetivo sea el menor. Para dicho problema, las variables de decisión involucradas en esta solución fueron el diseño de rutas y distribución de paraderos.

2.1.4 *Recocido Simulado*

Zhao y Gan (2003) proponen un algoritmo Recocido Simulado Integrado. El objetivo era lograr que las rutas sean más directas y minimizar el número de transferencias y maximizar la cobertura del servicio de transporte. Su algoritmo fue probado con data del Condado de Miami, Florida. Sus estudios determinaron que la metodología utilizada otorgaban significativas mejoras en una red existente en términos de transferencias directas definidas por la cantidad de transbordos y que tan directas son las rutas y la longitud promedio de un viaje definido por número de viajes cubiertos por la longitud de una ruta (Zhao, Gan, 2003).

Wei Fan y Randy Machemehl (2006) usaron un algoritmo Recocido Simulado para un modelo multiobjetivo para el TRNDP. La metodología de los autores se componía de un proceso de generación de rutas candidatos y seleccionar el conjunto de rutas óptimo, un procedimiento de análisis que asignaba los viajes y la frecuencia del servicio de transporte y el algoritmo de Recocido Simulado que combina ambas partes, guía la generación de soluciones candidatas y selecciona un óptimo conjunto de rutas. El objetivo era minimizar los costos de usuario y de operador que se representan por el costo de demanda no

satisfecha, costo por hora de operación del bus, horas de operación de un bus recorriendo una ruta y valor del tiempo de operación. Sus resultados indicaron que mientras el conjunto de rutas aumentaba en tamaño, inicialmente la solución mejoraba porque mayor demanda era cubierta y el costo de demanda insatisfecha disminuía. Sin embargo, al alcanzar la menor función objetivo, teniendo cierta cantidad de rutas para la red estudiada, el crecimiento del tamaño de la flota causaba una baja utilización de las rutas y no causaba una solución de calidad (Fan, Wei; B. Machemehl, Randy, 2006)..

2.1.5 *Búsqueda Tabú (Tabu Search)*

Fan y Machemehl (2004) explican el procedimiento de este algoritmo. La Búsqueda Tabú empieza con una solución mínima local, luego explora cada solución del conjunto de soluciones vecinas llamado vecindad. El principal enfoque es evitar entrar en ciclos, prohibiendo o penalizando movimientos que lleven a la posible solución (de la próxima iteración), a puntos del espacio de soluciones previamente visitados. Para evitar repetir o revertir pasos, estos son guardados en listas de soluciones Tabú. Una solución tabú o movimiento tabú es aquel movimiento que ya fue encontrado en alguna iteración anterior y se considera Tabú porque ya no deberá ser considerado nuevamente. La búsqueda Tabú utiliza un criterio de aspiración, el cual puede cambiar el estado de una solución tabú para que vuelva ser considerada. Esto ayuda a que soluciones de buena calidad halladas en iteraciones anteriores puedan volver a ser consideradas y lograr un mejor resultado.

Este método ha sido utilizado para resolver problemas de diseño de ruteos considerando una demanda fija o una demanda variable. Wei Fan y Randy B. aplicaron tres versiones de la búsqueda tabú en TRNDSP con demanda fija y variable, los cuales son los siguientes:

1. Búsqueda Tabú sin procedimientos de reorganización y con una cantidad determinada de tiempo o iteraciones que un movimiento es tabú, a lo que se llama Tabu Tenure.
2. Búsqueda Tabú con procedimientos de reorganización y con “tabu tenure” fija.
3. Búsqueda Tabú con procedimientos de reorganización y con “tabu tenure” variable.

Sus estudios dieron como conclusión que para representar el TRNDP con demanda fija usando redes pequeñas, la variación de búsqueda tabú que otorgó mejores resultados fue la segunda. Mientras las redes van incrementando de tamaño los resultados son muy similares en términos de calidad de solución, dando como conclusión que cualquier tipo de búsqueda tabú puede ser utilizada. Cuando la demanda es variable todas las variaciones los resultados tampoco mostraron grandes diferencias, al utilizarlos en redes pequeñas y de medianas (Fan, Wei; B. Machemehl, Randy, 2004).

Zhao y Gan (2003) y Zhao et.al (2005) utilizaron una combinación de la Búsqueda Tabú y el Recocido Simulado para obtener y configurar un óptimo diseño de rutas. Adicionalmente Zhao y Zen (2007) también utilizaron una combinación de los dos algoritmos, y reportaron que obtuvieron buenos resultados para una red de tránsito de gran tamaño en un tiempo razonable.

2.1.6 *Algoritmo Genético*

Un algoritmo genético es un tipo de algoritmo meta-heurístico bioinspirado que busca resolver problemas de optimización numérica simulando tanto el proceso selección natural y genética natura (Goldberg, Holland; 1988). Los elementos principales de este tipo de algoritmos por Coley AM, David (2003) son los siguientes:

a. **Población inicial:** Es el conjunto inicial de soluciones al problema, esta va a ir evolucionando mediante nuevas generaciones.
b. **Cromosoma:** Es una solución que pertenece a una generación.
c. **Función de Fitness o de Evaluación:** Es aquella que me indica, de manera cuantitativa, lo buena que es una solución.
d. **Operadores de selección:** Son aquellos que van a escoger a los individuos más aptos para formar una nueva generación.
e. **Operador de cruzamiento:** Es el proceso de “cruce” entre dos cromosomas, para formar uno nuevo. Imita el proceso de apareamiento entre animales.
f. **Operadores de mutación:** Estos operadores buscan imitar el proceso de mutación genética cambiado la configuración del ADN. Sus objetivos principales son evitar pérdida de diversidad en las soluciones, explorar áreas posiblemente no abordadas.
g. **Elitismo:** Consiste en conservar un grupo de la población para la siguiente generación.

Cabe resaltar que los Algoritmos genéticos son métodos adaptativos y se enfocan en resolver los problemas de búsquedas y de optimización. Como ya se mencionó, se basan en el proceso genético de los organismos vivos, esto implica que a lo largo de las "generaciones" desarrolladas por el algoritmo, estas van evolucionando mediante los principios de selección natural y supervivencia de los más fuertes, postulados por Darwin (Coley, 2003).

Este tipo de algoritmos ha sido un método de solución utilizado por diversos investigadores para resolver el problema del diseño de rutas de transporte público. Un trabajo presentado por Mohan, Pattnaik y Tom (1988) implementaba un algoritmo genético para resolver el problema del diseño de rutas para buses urbanos. Al año siguiente, una investigación presentada por Yang, Chien y Hou (1999) implementaba un algoritmo genético para resolver el problema de encontrar una ruta óptima de bus con el objetivo de minimizar los costos (tanto de los usuarios como de los proveedores), tomando en cuenta variables como la geografía, la capacidad y el presupuesto. Concluyeron que mientras el número de conexiones aumentaba, la cantidad de posibles rutas aumentaban dramáticamente, convirtiendo el problema TRNDSP computacionalmente intratable. Por esta razón, eligieron utilizar un algoritmo genético para determinar las ubicaciones óptimas de las rutas y terminales, y determinaron que el algoritmo elegido logra otorgar resultados de buena calidad. Dos años más adelante, en el Ngamachi y Lovell (2001) presentaron una investigación en la cual desarrollaban un algoritmo genético para resolver el problema de diseño de rutas de transporte público, el método presentado utilizaba 3 fases (generación de rutas, evaluación y mejora) y su objetivo era minimizar el tiempo de viaje de las personas, las variables de decisión involucradas eran tanto las rutas como la frecuencia de los buses. Aplicaron su propuesta para una red de tamaño pequeño logrando buenos resultados. Tres años después, Cipriani y Petreli (2004), con el objetivo de minimizar los costos de usuario y operador de la red de transporte implementaron este algoritmo y este otorgaba un conjunto de rutas, frecuencias y un tamaño apropiado de la flota de vehículos.

En general los algoritmos genéticos enfocados al diseño de rutas, usados en los trabajos pasados, antes de ejecutar el algoritmo genético se construye un conjunto de potenciales rutas que conforman el espacio de solución. Cada ruta es representado por un cromosoma, de esta manera el operador de selección consiste en seleccionar rutas, el operador de cruzamiento cruza particiones de las rutas para formar nuevas y el de

mutación modifica una ruta sea en un elemento de esta o más. De esta forma, se van construyendo nuevas soluciones conformadas por un conjunto de rutas.

2.1.7 *Algoritmo de Colonia Hormigas*

Poorzahedy y Safari (2011) explican, este algoritmo al igual que el genético es un algoritmo meta-heurístico bioinspirado, en este caso el algoritmo imita el comportamiento de las hormigas al encontrar el camino más corto. Para poder encontrarlo las hormigas dejan un rastro de feromonas, y como estas se mueven aproximadamente a una velocidad constante, el camino más corto tendrá una mayor densidad de feromonas, es de esta manera como pueden identificar cual es el camino óptimo.

En este algoritmo una “hormiga” es un procedimiento de construcción estocástico que incrementalmente construye una solución, cada hormiga construye una. Cuando una hormiga completa una solución las hormigas la evalúan y modifican el valor de la ruta con los componentes utilizados en la solución (Poorzahedy, Safari ; 2011). Un algoritmo de colonia de hormigas puede ser aplicado para cualquier problema de optimización en donde una heurística constructiva puede ser definida (Dorigo, Stützle 2004).

La optimización mediante el algoritmo de colonia de hormigas ha sido utilizada tanto para construcción como para la mejora de rutas. Hossain Poorzahedy y Farshid Safari (2011) buscaron minimizar el tiempo total de viaje de los usuarios, seleccionando un subconjunto de rutas interconectadas de un conjunto de rutas dadas. En este problema es necesario utilizar tantas colonias para las hormigas como par de origen-destino existan, así las hormigas de cada colonia realizarán la búsqueda de las rutas con menor costo para cada par. Yu, Yang y Liu (2005) utilizaron un algoritmo de colonia de hormigas paralelo para moldear el problema, con el objetivo de minimizar las transferencias y maximizar el flujo de personas por unidad de longitud, teniendo como restricciones la longitud de la ruta y velocidades variables. Ese mismo año una investigación desarrollada por Hu en conjunto con otros colegas presentó una forma de resolver este problema utilizando este método. Un año después, Yu y Yang (1988) buscaron incrementar la densidad de los viajes directos entre los puntos utilizando este algoritmo.

2.2 Software

2.2.1 *Sistema TRANSCAD*

Actualmente TRANSCAD es uno de los software más potentes para el cálculo de rutas óptimas, diseño de rutas de transporte, análisis y fácil cambio de las rutas. Forma parte de los sistemas de información geográfica, dentro de sus funcionalidades está diseñar rutas, este diseño de rutas se realiza especificando uno o más pares de puntos origen-destino para luego utilizar algoritmos de camino más corto como el algoritmo Shorthest Path y Multiple Shorthest Path para generar el camino. La herramienta también permite hacer modificaciones manuales a la ruta y establecer paraderos en la ubicación deseada. Asimismo, actualmente cuenta con extensiones especializadas en dicha área, enfocados en análisis de transporte de pasajeros. Este producto cuenta con dos versiones, la básica y la estándar, cuyos precios son 4 000 y 12 000 dólares respectivamente.

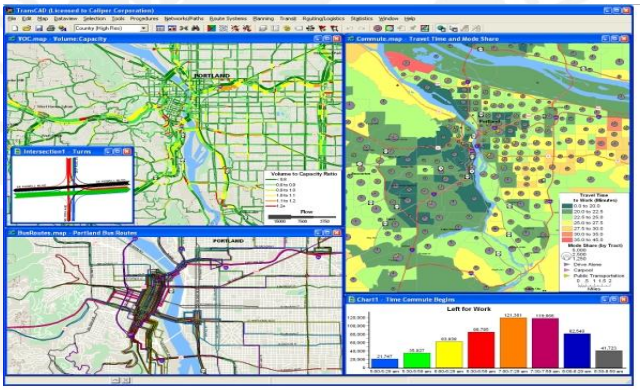


Figura 2 Imagen recuperada de Sistema Transcad página web

2.2.2 *WAY localizador GPS*

Esta aplicación para dispositivos Android es una herramienta que además de mostrar la locación de otros dispositivos y lugares vía GPS, también puede notificar al usuario la ruta más corta para llegar a un lugar definido. Su limitación es que solo puede proveer una ruta corta para llegar de un punto a otro; por esta razón, esta aplicación no es lo más adecuado para realizar un sistema de rutas de transporte.

2.2.3 *Google Maps*

Servicio gratuito de Google, es un servidor de aplicaciones en la web que tiene la funcionalidad de mostrar la ruta entre diferentes ubicaciones indicándote la distancia recorrida y el tiempo que tomaría recorrerlas. Sin embargo su principal función es la de ofrecer imágenes de mapas desplazables y fotografías por satélite. Por las características mencionadas este software no es el más adecuado para resolver un problema de diseño de ruteos.

2.3 Comparaciones

Un análisis de sensibilidad es necesario para aplicar mejor cada uno de los algoritmos mencionados. Con dicho análisis se logra definir óptimamente los parámetros de aleatoriedad y número de iteración que necesitan estos algoritmos, y de esta manera lograr mejores resultados. Según los estudios de Wei Fan y Randy B. Machemehl (2004), al considerar la variable de demanda fija, los valores de una óptima definición de parámetros definida para redes pequeñas no cambiaron significativamente para una red mediana, sugiriendo que esa definición de valores se puede generalizar para distintas redes. Por el contrario, al considerar la demanda variable, es necesario definir óptimamente los parámetros para cada red que se utilice dependiendo de su configuración y tamaño. Sin embargo, para cualquier tipo de demanda los resultados de los algoritmos mencionados fueron resultados de casi el mismo nivel y buena calidad. Es decir, cuando alguien quiere aplicar un algoritmo a este tipo de problemas cualquiera de los algoritmos pueden ser utilizados. Por otro lado, los resultados del problema con demanda variable son distintos a los resultados considerando la variable fija; por lo tanto, considerar la demanda variable para futuros estudios es importante.

Entre las metodologías utilizadas se distingue que una gran parte centra sus funciones objetivo en minimizar los costos sea de operador, de usuario o ambos a la vez y también en maximizar la cobertura de demanda. Las restricciones que generalmente se han utilizado son: máximo y mínimo de longitud de rutas, máximo número de rutas, máximo volumen de pasajeros por rutas; para problemas que incluyen el establecimiento de frecuencias, entre las restricciones más utilizadas se han considerado también la cantidad mínima de frecuencias, máximo factor de carga de pasajeros, y máximo tamaño de la flota

de buses. Por último, las variables de decisión más utilizadas han sido primordialmente las rutas para este problema, y en variaciones las rutas y frecuencias.

En la Tabla 3 se muestran algunos autores, los objetivos de sus trabajos y los algoritmos que utilizaron para resolver el TRNDSP o variantes.

2.4 Conclusiones sobre el estado del arte

Se puede concluir que diversas metodologías han sido aplicadas al resolver el TRNDP. Entre las variables que han sido consideradas por distintos autores pueden variar en cantidad y tipo. La mayoría de autores han enfocado el problema con la variable de demanda fija para simplificar el problema y decidieron resolver el problema considerando un solo objetivo, generalmente el de disminuir tiempo, cuando el problema por su naturaleza es multiobjetivo. Por otro lado, el aplicar un enfoque matemático no es la mejor opción, debido a que se vuelve muy complicado representar el problema mientras su tamaño no sea pequeño. Los algoritmos meta-heurísticos son los que mejor se adaptan a este problema de gran dificultad, logrando resultados buenos en un tiempo de ejecución razonable.

Se identifica que los trabajos anteriores enfocados al diseño de rutas, se ha utilizado el algoritmo Recocido Simulado para minimizar transferencias como en los trabajos de Zhao y Gan (2003), Zhao y Ubaka (2004) y Zhao et. al (2005). Por otro lado el minimizar el tiempo de viaje en trabajos anteriores, Yang, Chien y Hou (1999) y Ngamachi y Lovell (2001), ha sido tomado como único objetivo. El lograr maximizar la demanda cubierta ha sido tomado en conjunto con otros objetivos distintos al reducir el tiempo de viaje y centrándose más en el estableciendo de frecuencias, entre estos trabajos se identifican los de Wei Fan y Randy B. Machemehl (2004 y 2006). El algoritmo genético ha sido utilizado en enfoques que tomaron como objetivos la minimización de costos y el tiempo de viaje; sin embargo, estos enfoques fueron tomados por separado. Tomando en cuenta el cuadro anterior y las observaciones, se desarrollará un algoritmo Recocido Simulado que genere un sistema de rutas, que optimice el tiempo de viaje y el cubrimiento de la demanda de transporte, ya que estos objetivos han sido estudiados por separado o tomando otros enfoques en conjunto y utilizando otros algoritmos.

Año	Autor	Objetivos	Variables de Decisión	Metodología
1999	Yang, Chien y Hou	Minimizar Tiempo de viaje	Rutas y frecuencia	A. Genético
2001	Ngamachi y Lovell	Minimizar Tiempo de viaje	Rutas y frecuencia	A. Genético
2003	Murray	Incrementar acceso a la red y su accesibilidad	Paraderos	E. Matemático
2003	Guihaire, Jin-Kao Hao	Minimizar cantidad y longitud de rutas	Rutas	E. Matemático
2003	Zhao y Gan	Minimizar transferencias	Rutas	R. Simulado
2004	Fan y B. Machemehl	Maximizar cobertura de demanda y minimizar costos	Rutas y paraderos	B. Exhaustiva
2004	Zhao y Ubaka	Maximizar cobertura de demanda y minimizar transferencias	Rutas	R. Simulado
2004	Fan y B. Machemehl	Satisfacer la demanda y minimizar costos	Rutas y paraderos	Búsqueda Random
2004	Cepreni et al.	Minimizar costos de usuario y operador	Rutas y frecuencia	A. Genético
2004	Petrelli	Minimizar costos de usuario y operador	Rutas y frecuencia	A. Genético
2005	Fan y B. Machemehl	Satisfacer la demanda y minimizar costos	Rutas y paraderos	B. Tabú
2005	Zhao et.al	Minimizar transferencias	Rutas	B. Tabú–R. Simulado
2005	Yu, Yang y Liu	Minimizar transferencias y maximizar el flujo de personas por unidad de longitud	Rutas	C. de Hormigas
2005	Hu et. al	Minimizar costos de usuario y operador	Rutas y frecuencias	C. de Hormigas
2006	Fan y B. Machemehl	Minimizar costo de usuario y de demanda insatisfecha	Rutas y frecuencias	R. Simulado
2006	Yu y Yang	Densidad de viajes directos	Rutas y paraderos	C. de Hormigas
2007	Zhao y Zeng	Minimizar costos de usuario y operador	Rutas y frecuencia	B.Tabú–R. Simulado
2011	Poorzahedy; Safari	Minimizar tiempo de viaje total	Rutas	C. de Hormigas

Tabla 3 Estudios del problema.

Capítulo 3 Estructuras de Información y del Algoritmo

En este capítulo se presentarán los resultados obtenidos para el primer objetivo específico. Se determinó como primer objetivo específico el establecer las estructuras de almacenamiento de la información y del resultado. Para este objetivo se cuentan con dos resultados. El primer resultado muestra la recolección de la información, una descripción del diseño de la base de datos y los procedimientos usados para procesar la información obtenida y cargarla a la base diseñada. En el segundo resultado se presentan el diseño de las estructuras de datos y de la solución que se utilizan en el algoritmo.

1 R1O1: Formato de almacenamiento de la demanda de transporte por zona, vías transitables y solución. Diseño de base de datos para almacenar la información recolectada.

1.1 Recolección de Información

Para este proyecto de fin de carrera fue necesario conseguir información sobre la demanda de transporte público. Al revisar el informe final de la Encuesta de Recolección de Información Básica del Transporte Urbano en el Área Metropolitana de Lima y Callao (JICA, 2013) se encontró que existe información guardada y digitalizada sobre zonas de tránsito y la demanda de transporte levantada en base a esas zonas. Por esta razón, se contactó con diversas entidades encargadas de la gestión del transporte público para solicitarles tal información requerida. Los archivos que se lograron conseguir son los siguientes:

- **Archivos Zonas de Transito:** Otorgado por el Ministerio de Transportes y Comunicaciones. Este conjunto de archivos guardan la distribución de zonas de tránsito en la provincia de Lima. Los archivos relevantes son los de formato .shp y el .dbf, tales archivos se pueden visualizar con programas como QGis o Arcguis. El primero es un formato de imagen donde se muestra la composición de Lima en zonas de tránsito utilizando polígonos, el segundo guarda información sobre las zonas de tránsito. La información más relevante es su identificador (ID) y a qué distrito pertenece la zona (Distrito). El archivo .shp necesita del archivo .dbf y del resto de archivos para poder pintar el conjunto de zonas de tránsito en un mapa de Lima. Fue necesario añadir la información de polígonos para cada Zona de Tránsito, para esto se utilizó la información guardada en el archivo de zonas de

tránsito en formato .shp. El archivo obtenido es un archivo en formato .csv La estructura final del archivo obtenido se muestra en la Figura 3.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
1	WKT	RefNa	ID	Area	ID_c	Area_1_	Zone	Traffi	LandU	Distrito	Zconb	Zconbi	Regi	Zona5	Zona53	Zona6	Dist	Centroi	DestAut	DestPubl
2	POLYGON ((-77.02373 -11.68162,-77.0	B1	1	142.17	355	142.17	48	601	NO	6	20	9	5	48	48	48	CARABAYLLO	6582	429	382
3	POLYGON ((-77.02373 -11.68162,-77.0	B2	2	323.03	342	323.03	23	201	NO	2	10	8	5	23	23	23	ANCON	6581	561	1827
4	POLYGON ((-77.18867 -11.7862,-77.17	B3	171	12.825	28	12.825	322	3901	RDM	39	125	8	5	322	322	322	SANTA ROSA	6186	105	27
5	POLYGON ((-76.71871 -12.00019,-76.7	B4	3	226.6	253	226.6	81	901	NO	9	31	12	1	81	81	81	CIENEGUILLA	7352	1307	1848
6	POLYGON ((-76.71871 -12.00019,-76.7	B5	4	26.794	157	26.794	60	702	NO	7	23	11	4	60	60	60	CHACLACAYO	8095	858	1586
7	POLYGON ((-76.75747 -11.97128,-76.7	B6	5	44.446	199	44.446	169	1806	NO	18	63	11	4	169	169	169	LURIGANCHO	7363	329	525
8	POLYGON ((-76.7351 -11.96115,-76.73	B7	6	30.951	335	30.951	168	1805	NO	18	62	11	4	168	168	168	LURIGANCHO	8107	233	669
9	POLYGON ((-76.71418 -11.99266,-76.7	B8	7	17.509	333	17.509	167	1804	NO	18	62	11	4	167	167	167	LURIGANCHO	8104	194	2091

Figura 3 Archivo Zonas de Tránsito

Las columnas relevantes son:

- **WKT:** Guarda los polígonos de la zonas de tránsito.
- **ID:** Identificador de la zona de tránsito.
- **Area:** Área total de la zona de tránsito.
- **Dist:** Nombre del distrito donde pertenece la zona de tránsito.
- **Distrito:** Guarda el código del distrito donde pertenece la zona de tránsito.

El resto de columnas guardan información que no es necesaria en el proyecto.

Los pasos para obtener este archivo son los siguientes:

1) Abrir la Terminal y ubicar la carpeta donde se encuentra el archivo .shp y .dbf.
2) Ejecutar el comando: **ogr2ogr -f CSV output.csv NOMBRE.shp –lco GEOMETRY=AS_WKT -lco SEPARATOR=SEMICOLON**. Donde NOMBRE es el nombre del archivo y SEPARATOR = SEMICOLON indica que los campos del archivo estarán separados por el signo “;”.
   Se genera un archivo con extensión .csv de nombre output.

- **Archivo Matriz de Demanda de Transporte Público:** Otorgado por el Ministerio de Transportes y Comunicaciones. El archivo con formato .xlsx muestra la demanda de transporte público y de transporte privado por cada zona de tránsito definida en los distritos de Lima. Estos datos han sido recolectados en base a zonas de tránsito y reflejan la demanda de transporte en el intervalo de tiempo entre las 7:00 am y 8:00 am (hora pico de la mañana) para el año 2012 (JICA, 2013). El archivo está compuesto por un sector de columnas donde se muestra la información de demanda

(columna VIAJES) por cada zona origen (columna ORIGEN) y destino (columna DESTINO) y una segunda donde se muestra la misma información en una matriz.

La demanda del transporte público está expresada en cantidad de viajes por persona y la del transporte privado en cantidad de autos. Para fines del presente proyecto se transforman las unidades de ambas demandas en cantidad de personas. Los valores de la demanda de transporte público se transformaron con un factor cuyo valor es de 1.5 viajes/persona y para los valores del transporte privado con el factor cuyo valor es de 1.2 personas/auto. Tales factores fueron tomados como referencia del Plan Maestro de Transporte Urbano para el Área Metropolitana Lima y Callao en la República del Perú (JICA et al. , 2005).

	A	B	C	D	E	F	G	H	I
1	ORIGEN	DESTINO	VIAJES						
2	1	1	2.80744						
3	1	2	4.13057		Suma de VIAJES	Rótulos de columna			
4	1	3	12.276034		Rótulos de fila	1	2	3	4
5	1	4	13.912971		1	2.80744	4.13057	12.276034	13.912971
6	1	5	92.482521		2	1.40039	3.99504	22.3351	20.65023
7	1	6	2.651332		3	10.2809	5.17645	33.309391	17.797693
8	1	7	16.08696		4	2.13028	5.32109	17.28932	15.968026
9	1	8	0.396067		5	1.01215	1.78429	25.69631	7.873637
10	1	9	2.844897		6	4.37315	2.25437	15.80502	6.970222
11	1	10	99.758301		7		5.06822	23.22744	17.11598
12	1	11	4.32612		8	0.41676	0.800202	1.614742	2.147441
13	1	12	145.060394		9	1.56537	6.31426	13.56051	17.465958
14	1	13	74.832848		10		2.93207	26.103001	7.229542
15	1	14	10.776907		11	2.67695	4.35533	21.08946	11.848105
16	1	15	24.18256		12			10.49327	1.8303
17	1	16	11.24063		13		1.40058	19.31139	6.602096
18	1	17	62.304131		14	6.11778	7.19648	86.763733	34.266541
19	1	18	18.011499		15	3.44752	1.24838	5.81759	3.021921
20	1	19	3.209882		16		0.619452	1.710227	3.54009
21	1	20	9.37757		17	6.30326	9.57095	48.344528	31.719423

Figura 4 Archivo Matriz Demanda OD

1.2 Diseño de Base de Datos

Además de recolectar la información se creyó conveniente crear una base de datos PostgreSQL, en la cual se guarde la data recolectada como la matriz de demanda, las zonas de tránsito y la estructura vial del área determinada en el proyecto de tesis. Además se almacenan las zonas del área para las cuales se correrá el algoritmo y también todos los datos y parámetros que son necesarios para correr el algoritmo. Por último dicha base también permite guardar los resultados obtenidos por el algoritmo, que se determina por el sistema de rutas y demanda cubierta por tal sistema. Fue

necesario añadir la extensión PostGuis a la base de datos, ya que permite almacenar datos de georreferenciación como los polígonos y linestring añadidos en algunas de las tablas. Esta extensión permite guardar la estructura vial mencionada, por otro lado permite graficar las rutas obtenidas como resultado del algoritmo. La tabla "spatial_ref_sys" y las tablas que se encuentran en el módulo "raster_overviews" son propias de la extensión PostGuis y se generan como resultado de añadir tal extensión a la base de datos. El diseño de la Base de datos se presenta en la Figura 5 y el diccionario de datos se presenta en el Anexo 1 Diccionario de Datos:

1.3 Carga de Datos

El flujo de la carga de datos a la base de datos diseñada se presenta en la Figura 6 muestra. Los procedimientos indicados en el flujo se explican a continuación:

- **Procedimiento de mapeo de estructura vial:** Gracias a herramientas como OpenstreetMaps y Pgrouting se logra conseguir la estructura vial en una base de datos llamada Pgouting-workshop, esta base no es la descrita anteriormente. El procedimiento para lograr dicha base de datos se visualiza en el anexo 2 Flujo de Carga de Información red vial. En esta base cada calle es una arista y cada nodo es una esquina. Sin embargo para un mejor manejo de esta información se realizaron procedimientos que migren esta data a la base de datos diseñada. Así se tiene la información clasificada en las tablas de nodo, arista, vías y tipo de vía. Las aristas tienen asociado un campo LineString, el cual servirá para poder graficar cada arista en un mapa. Este proceso inicia con el *procedimiento Lectura y el Procedimiento de los Datos* de la estructura vial almacenada en la base Pgrouting-workshop, logrando procesar los registros leídos para modificarlos y generar los registros a guardar en las tablas indicadas de la base de datos diseñada, y finaliza el procedimiento Carga de Datos de Estructura Vial, la cual almacena los registros generados en el proceso anterior en la base de datos.

- **Procedimiento Lectura y Procesamiento de archivos:** Se realizaron procedimientos que leen los datos de los archivos de Zonas de Transito y Demanda, estos datos son procesados para ser cargados en la base de datos finalmente en el proceso de Carga de Datos.

Figura 5 Diseño de Base de Datos

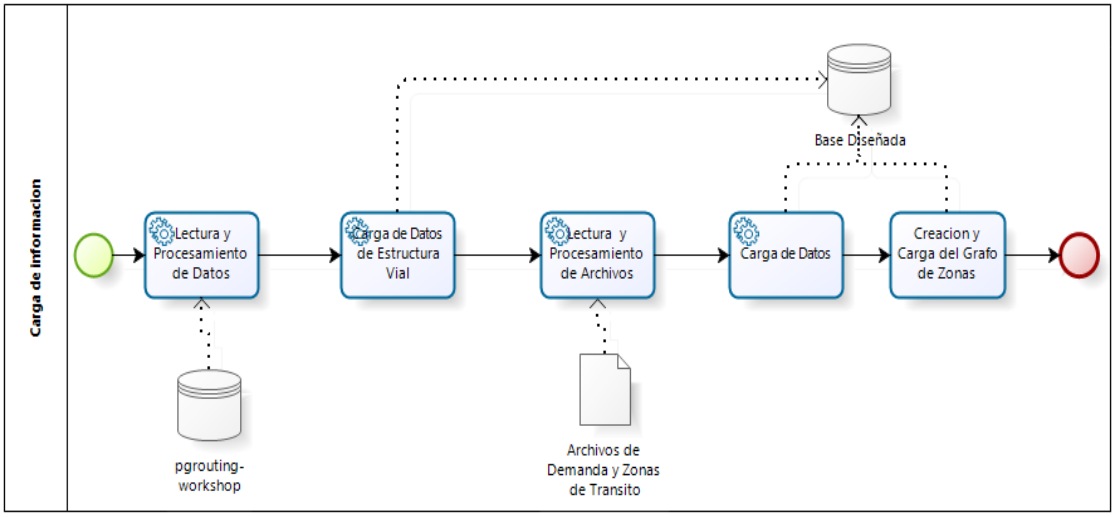


Figura 6 Carga de Información.

- **Procedimiento de creación de grafo de zonas:** Teniendo las zonas de transito almacenadas en la base de datos, se desarrolló un procedimiento que automáticamente verifique, para cada zona con cuales otras tiene conexión o son adyacentes, y guarde esa información de adyacencia en la tabla arista_minizona. Tal tabla es de suma importancia, ya que permitirá crear el grafo de zonas. Este grafo servirá para la elaboración de las rutas a nivel de zonas. Para esto el uso de los polígonos de cada zona fue de suma importancia, ya que a partir de ellos se hallan las adyacencias entre las zonas. Para realizar esta operación fue necesarios funciones de la extensión PostGis para base de datos PostgreSQL. Las funciones usadas son las siguientes:

    - *ST_Touches*: Este comando verifica si entre dos polígonos comparten algún punto pero sus interiores no se intersectan. En simples palabras si son adyacentes. Esta función permite obtener las aristas del grafo de zonas.
    - *ST_Centroid*: Este comando calcula el centro geométrico del polígono como un punto. Retorna la longitud y la latitud del centro geométrico.
    - *ST_Distance*: Con este comando se calcula la distancia mínima que existe entre dos puntos. Así con la ayuda de la función ST_Centroid se logra obtener la distancia entre cada zona de tránsito.

2 R2O1: Diseño de las estructuras de datos y de la solución del algoritmo.

Para este resultado se realizaron dos etapas. Como primera etapa se tuvo que definir los datos de entrada, variables de decisión, restricciones y la función objetivo a utilizar en el algoritmo. Los valores y explicación de los datos se presentan en la sección Consideraciones. Finalmente como segunda etapa se definieron las estructuras a utilizar en el algoritmo.

2.1 Datos

Se muestra la lista de datos de entrada que requiere el algoritmo para resolver el problema.

- N Número de zonas.
- E Número de aristas.
- $M_{i;j}$ Matriz de demanda de viajes de zona *i* a *j*.
- Lmin Distancia mínima de cualquier ruta.
- Lmax Distancia máxima de cualquier ruta.
- VpromP Velocidad promedio de transporte público en Lima Metropolitana.
- VpromPr Velocidad promedio de transporte privado en Lima Metropolitana.
- Fl Flota, cantidad de buses por ruta.
- Bcap Capacidad de un bus, cantidad de personas que puede transportar un bus.
- CAP Proporción de la capacidad total de la ruta (Rcap).
- P % Porcentaje mínimo de la demanda satisfecha a cubrir en la solución.
- CHH Costo de horas hombre o Valor Social del Tiempo para usuarios del transporte urbano en Lima Metropolitana.
- CHHP Costo de horas hombre o Valor Social del Tiempo para usuarios del transporte privado.

- $C2_{i,j}$ Costo unitario por persona con demanda insatisfecha expresado en unidades monetarias. Se puede traducir como el costo constante de transporte privado (taxi) por distancia entre las zonas *i* y *j*. Entre zonas no adyacentes se toma la menor distancia entre ellas.

Los costos de la matriz $C2_{i,j}$ se componen utilizando el tiempo total de viaje expresado en valores monetarios y el costo del transporte privado. Asumiendo un costo de 1.25 soles/km, que representa el costo por kilómetro de movilizarse en un transporte privado para una persona.

$$C2_{i,j,} = (Tv) * CHHP + Dist*1.25$$

Ecuación 1 Matriz de costos de demanda insatisfecha

Donde:

- Tv es el tiempo de viaje
- Dist es la distancia entre las zonas i y j

- Rmax Número máximo de rutas.
- $D_{i,j}$ Matriz de demandas de transporte entre las zonas i y j.
- W1; W2 Son introducidos para reflejar la importancia de cada parte de la función objetivo.

2.2 Variables de Decisión

Se muestra las variables de decisión cuyos valores serán explorados y determinados por el algoritmo.

- R Conjunto de rutas.
- N_{rutas} Número de rutas.
- $C1_{i,j,k}$ Matriz de costos por viaje satisfecho de la zona i a la zona j por medio de la ruta k de forma directa por persona expresado en unidades monetarias. Para su cálculo se toma en cuenta el tiempo de viaje por el costo de horas hombre.

Los costos de las matriz $C1^{d}_{i,j,k}$ se componen por el tiempo total del viaje expresados en costo monetario para el usuario y el costo de pasaje promedio (S/. 1.5). Se define:

$$C1^{d}_{i,j,k} = (Tv) * CHH + 1.5$$

Ecuación 2 Matriz de costos de demanda satisfecha

Donde:

- Tv es el tiempo de viaje

- $D_{i,j,k}$ Matriz de demanda de viajes satisfechos de la zona i a la zona j por medio de la ruta k de forma directa. Dado que cada ruta puede transportar CAP pasajeros y que se asigna la capacidad completa de la ruta hasta completar la demanda total:

$$D_{i,j,k} = \begin{cases} CAP.RCAP, si\ k \leq \dfrac{M_{i,j}}{CAP.RCAP} \\ M_{i,j} - CAP.RCAP \cdot \dfrac{M_{i,j}}{CAP.RCAP}, si\ k = \dfrac{M_{i,j}}{CAP.RCAP} \\ 0, si\ k > \dfrac{M_{i,j}}{CAP.RCAP} \end{cases}$$

Ecuación 3 Matriz de demanda satisfecha

La asignación de demanda cubierta reflejada en las ecuaciones anteriores se dan de la siguiente forma: Para el par *i-j* se resta la demanda con el valor CAP.RCAP, mientras ese valor sea mayor a la demanda. Si no lo es se actualizará con el valor de la demanda sin cubrir, logrando cubrir toda la demanda y el resto de rutas que pasen por el par de zonas *i-j* no cubrirán más demanda.

Para cada par *i-j* de la matriz C1 el valor de los costos por la demanda cubierta por cada ruta *k* se empieza a cubrir por las rutas que recorran con menor distancia de la zona i a la zona j. Es decir que la ruta con mayor distancia entre las zonas *i,j* será la que se cubrirá menor demanda.

- $\forall\, k \in R$, está representado de la forma: $k = \{e_1, e_2 \dots e_{tam}\}$, tal que *tam* es la cantidad de aristas de la ruta *k* y la arista e_i une dos nodos.
- $U_{e,k}$ Matriz de aristas usadas e para cada ruta *k*: 0 (No usada) , 1 (Usada).

2.3 Función Objetivo y Restricciones

Se presenta la Función Objetivo y restricciones a utilizar.

Función Objetivo:

$$\boldsymbol{min\ Z = W1.} \sum_{i\in N} \sum_{j\in N} \sum_{k\in R} \boldsymbol{D_{i,j,k} \cdot (C1_{i,j,k})} + \boldsymbol{W2.} \sum_{i\in N} \sum_{j\in N} \boldsymbol{(C2_{i,j}) \cdot (M_{i,j} - \sum_{k\in R} D_{i,j,k})}$$

Sujeto a:

$$\textbf{1.}\ \forall k \in R:\ Lmin \leq \sum_{i=1}^{tam} e_i \leq Lmax$$

$$\textbf{2.}\ \boldsymbol{P\%.} \sum_{i\in N} \sum_{j\in N} \boldsymbol{M_{i,j}} \leq \sum_{i\in N} \sum_{j\in N} \sum_{k\in R} \boldsymbol{D_{i,j,k}}$$

$$\textbf{3.}\ Nrutas \leq Rmax$$

Ecuación 4 Función objetivo y restricciones

El primer término de la función objetivo representa el costo de usuarios. Es decir, el costo total de transportar a las personas por una ruta *k* de i a j (C1), tal costo se conforma por el tiempo de viaje transformado a valores monetarios. El segundo término representa el costo de la demanda insatisfecha (C2), este término representa el costo de transportar a las personas cuya demanda no logro ser cubierta por las rutas de transporte público.

Restricciones:

1. La restricción 1 limita la longitud máxima y mínima de una ruta en kilómetros.
2. La restricción 2 establece el porcentaje mínimo de demanda a ser cubierto.
3. La restricción 3 limita la máxima cantidad de rutas de la solución.

2.4 Consideraciones

Tomando como referencia las observaciones encontradas en el informe final del Plan Maestro de Transporte Urbano para el Área Metropolitana Lima y Callao en la República del Perú (JICA et al. ,2005) sobre las condiciones y características del sistema de transporte actual, se identifican que la velocidad promedio de viaje es por debajo de 20km/h en diversos tramos, por otro lado la longitud promedio de las rutas es de 64.3 km en ida y vuelta. Teniendo como referencia estos datos se establecen los valores para los siguientes datos:

- Lmin = 5 km.
- Lmax = 30 km.
- Mmax = 500 rutas.
- VpromP = 17 km/h.
- VpromPr = 23 km/h.

En el estudio a cargo del Banco Central de Reserva del Perú (2011) el Valor Social del Tiempo para usuarios del transporte urbano en Lima Metropolitana tiene un valor de S/. 8.8 por hora y el valor social del tiempo para usuarios de transporte privado se considera en aproximadamente el doble. Bajo esta información se establece:

- CHH = S/. 8.8
- CHHP = S/. 17.6

La capacidad total de la ruta Rcap se calcula con la multiplicación de la cantidad de la flota con la capacidad del bus. Se define:

$$Rcap = Fl * Bcap$$

Se toma como capacidad del bus, la misma capacidad de los buses de los corredores azules de 80 pasajeros (Municipalidad Metropolitana de Lima, 2014), asumiendo que los buses de cada ruta parten cada 90 segundos, para la hora pico de la mañana (total de 1 hora) se necesita una flota de 40 buses por ruta. Por último se toma una proporción de capacidad de la ruta del 10%, mínima demanda cubierta en solución inicial de 60% y como demanda mínima cubierta P% un valor de 85%.

- Bcap = 80
- Fl = 40
- CAP = 10%
- P% = 80%

En el proyecto el disminuir el tiempo de viaje tiene una ligera mayor importancia que cubrir la mayor cantidad de demanda por tanto se establecen los pesos W1 y W2 para los términos de la función objetivo, cuyos valores son los siguientes:

- W1 = 0.6
- W2 = 0.4

2.5 Estructuras

En la segunda etapa se diseñaron las estructuras a utilizar. Se definieron estas estructuras orientadas al trabajo con grafos, y de manera que sea sencillo utilizarlas en los procedimientos del algoritmo. Se buscó también que guarden la mayor relación mutua posible con el diseño de la base de datos para simplificar el manejo de esta en cuanto a extracción y almacenamiento de los datos. Cada estructura será implementada como un objeto clase.

Arista: Estructura para representar las conexiones entre los vértices del grafo. Tiene los siguientes atributos:

- VO: Vértice origen
- VD: Vértice destino
- distancia: Distancia entre nodos

Demanda: Estructura que representa la celda de la matriz de demanda O-D. Tiene los siguientes atributos:

- zonaO: Zona Origen
- ZonaD: Zona Destino
- demanda: Demanda de transporte de la zona origen a la zona de destino

- dirCub: Demanda cubierta por viajes directos.
- DemSinCubrir: Demanda que no ha sido cubierta por ningún tipo de viaje.

Grafo: Estructura que representa el grafo a nivel de zonas. Este grado es un grafo no dirigido. Tiene los siguientes atributos:

- V: Cantidad de vértices.
- E: Cantidad de aristas del tipo aristaZona.
- G: Relación de vértices y aristas (Vector de vectores).

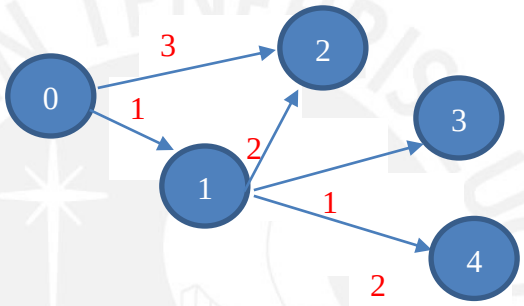


Figura 7 Ejemplo de Grafo.

Para el grafo mostrado en la figura anterior hay un total de 5 vértices y 5 aristas. La estructura Grafo tendría como atributo V = 5 y A = 5. La relación de vértices y aristas se estructura en un vector de vectores de aristas. Se puede entender así: cada fila representa un vértice (primera fila - primer vértice, segunda fila - segundo vértice y así sucesivamente) y cada columna de la tabla representa las aristas del vértice. Similarmente para el GrafoZona.

A continuación en la Figura 8 se muestra la estructura que representa el grafo mostrado anteriormente.

V = 5

E = 5

G =

0	1	1	0	2	3			
1	2	2	1	3	1	1	4	2

Figura 8 Ejemplo Estructura Grafo.

En la figura mostrada se aprecia los valores de los atributos V y E, los cuales ambos son de valor 5, ya que se tienen 5 vértices y 5 aristas en el grafo 1. El atributo G solo contiene dos filas porque en el grafo 1 los únicos vértices con aristas son el 0 y 1. El vértice 0 tiene como aristas al vértice 1 con peso 1 y vértice 2 con peso 3 y estas son expresadas en la primera columna de la tabla como 0 1 1 y en la segunda como 0 2 3.

Ruta: Estructura que representa las rutas diseñadas del transporte público a nivel zonas de demanda. Tiene los siguientes atributos:

- nodos: Vector de nodos (zona) que conforman la ruta.
- costo: Distancia total del recorrido de la ruta.

Se presenta un ejemplo de ruta generado en el grafo de ejemplo.

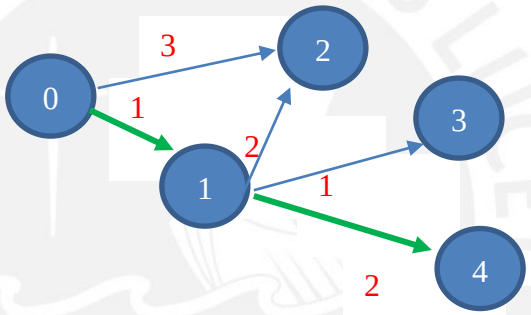


Figura 9 Ejemplo de Ruta.

Ruta 1:

0	1	1	1	4	2

En este ejemplo de ruta, se observa que la ruta está conformada por los vértices 0, 1, y 4 con las aristaVia 0-1, 1-4 con una distancia total de 3 km.

Solución: Estructura que representa la solución del algoritmo. Tiene los siguientes atributos:

- rutas: Vector de rutas seleccionadas.
- valor: Resultado de la función objetivo en la solución.

A continuación se presenta un ejemplo para el cálculo de la función objetivo.

Suponiendo que se tiene la siguiente tabla matriz de demanda:

Demanda O-D 0	0 0	1 1
1	1	0

Tabla 4 Matriz ejemplo de demanda de transporte

Se observa que solo existen dos zonas, la cuales tienen un valor de demanda igual a 1 y la distancia entre la zona 0 y 1 es de 20 km. El algoritmo obtiene como resultado que es necesaria una sola ruta y sin demanda insatisfecha resultante. Se obtendría como valor objetivo el valor de:

Función Objetivo = (1*((20)/(VpromP))*8.8 +1*1.5) + (1*20*1.25+1*(20/VpromPr)*(2*8.8))= S/.8.8

Ecuación 5 Ejemplo Función Objetivo

La estructura de la solución sería la siguiente:

Solucion1:

Ruta 1
8.8

Para casos donde se encuentren más de una ruta el atributo rutas contendrá la siguiente lista:

Lista de rutas de solución:

Ruta 1	Ruta2	Ruta3	...	Ruta n

Capítulo 4 Diseño del Algoritmo

En este capítulo se presentan el resultado obtenido para el objetivo específico número 2. En este objetivo se planteó el implementar un algoritmo Recocido Simulado que diseñe un sistema de rutas para disminuir el tiempo de viaje total y la demanda insatisfecha. Como resultado se presenta el pseudocódigo desarrollado de los procedimientos que generan la solución inicial y del algoritmo Recocido Simulado. Estos algoritmos serán implementados en lenguaje C++.

1 R3O2: Pseudocódigo del algoritmo Recocido Simulado implementado, que diseñe el sistema de rutas para el área a considerar

El algoritmo Recocido Simulado tiene entre sus características la necesidad de una solución inicial para empezar su proceso. Esta solución inicial se construye utilizando una variación del algoritmo PIA. Con este algoritmo se construirán las rutas de la solución inicial. Recordando que el algoritmo PIA realiza la creación de rutas en base a un grafo de zonas que representan las zonas del área geográfica, y que la información de demanda de transporte obtenida es en base a zonas de tránsito, las rutas generadas por el algoritmo desarrollado en este proyecto son rutas generadas a nivel de zonas. Este enfoque de creación de rutas no solo ha sido usado en el algoritmo PIA (Mauttone, Urquhart, 2009), sino también en trabajos como los de Yu, B et. al. (2005) y Sadrsadat, Hadi et. al. (2003).

1.1 Algoritmo de creación de Solución Inicial

El algoritmo para generar la solución Inicial se basa en el algoritmo PIA, sin embargo no es exactamente igual al original y se le nombrará como PIA variado (PIA V). La variación consiste en cómo se seleccionan los pares de zonas origen-destino a evaluar en cada iteración y en el método de inserción de pares. El algoritmo original siempre selecciona el de mayor demanda global de la lista de pares origen-destino, en el PIA desarrollado se selecciona aleatoriamente un par el cual se ubica dentro de un porcentaje (P%lista) de toda la lista donde se encuentran los pares con mayor demanda. A partir de este punto sigue el procedimiento original de crear una ruta nueva (función *creaRutaDirecta*), si no excede el máximo de rutas propuesto o insertar el par origen - destino en una ruta ya existente (función *insertarEnMejorRuta*). El costo de una ruta es el valor de la distancia total de la ruta, para evaluar cual opción elegir, se compara el costo de generar la ruta nueva, y la diferencia de costo entre la ruta

generada al insertar el par O-D en la ruta X y el costo de la ruta X, la opción con el menor valor generado será la elegida. Si ya se alcanzó el máximo de rutas solo queda la opción de insertar en las rutas ya existentes. Para crear una ruta nueva o realizar una inserción se encuentran las rutas más cortas utilizando el algoritmo Dijkstra. Este algoritmo finaliza cuando la demanda cubierta cubre el porcentaje mínimo establecido (P%). Se presenta los procedimientos utilizados en la variación del algoritmo PIA propuesto.

```
crearSolucionInicial ()
  R lista de rutas;
  MIENTRAS no se cumpla la demanda Minima
        [M1,M2] = seleccionarParZona (R);
        Si no se excede el Rmax entonces
              rutaDirecta = creaRutaDirecta(M1,M2);
        rutaInsercion = insertarEnMejorRuta(M1,M2, rutaSeleccionada,R);
        SI (costo(rutaDirecta)<costo(rutaInsercion)-costo(rutaSeleccionada))
        entonces R = R U {rutaDirecta};
       SINO R = R U {rutaInsercion} - {rutaSeleccionada}


seleccionarParZona( lista de rutas R )
        Ordenar la lista de mayor a menor demanda sin cubrir;
        rango = (tamaño de la lista R) / 3; // tercera parte de la lista
        posición = numero aleatorio entre 0 y rango;
        parOD = lista[posición];
        retornar parOD
```

Pseudocódigo 4 Algoritmo Creación Solución Inicial

Función InsertarEnMejorRuta: Esta función recibe el par de zonas origen-destino seleccionado y el conjunto de rutas generadas hasta el momento. Para cada ruta de un subconjunto de un número de rutas (numRutas) seleccionadas aleatoriamente esta función evalúa si las zonas se encuentran en la ruta, esta es la segunda variación ya que el algoritmo original busca en cada una de las rutas generadas. Si una zona del par ya se encuentra en la ruta, insertará la zona que no se encuentre aún. En caso ninguna de las zonas se encuentren en la ruta, entonces se insertarán ambas empezando por la zona origen del par O-D seleccionado. Buscando la mejor posición para la zona origen y la mejor posición para la zona destino. En cada inserción se evalúa la mejor posición de la ruta donde convenga insertar la nueva zona, es decir la posición donde genere menor costo (menor distancia). Como la función evalúa la mejor posición para cada una de las rutas donde convenga realizar la inserción, al final se obtiene una nueva ruta (*rutaResultado*) y la mejor ruta seleccionada que generaba

la inserción con menor costo (*rutaSeleccionada*). Se muestra el procedimiento de la función en Pseudocódigo 5.

1.2 Algoritmo Recocido Simulado

Función Principal: La función principal del algoritmo Recocido Simulado se presenta según lo explicado en el Capítulo 2- Sección 1, basándonos en el modelo del algoritmo Recocido Simulado que usa la función de Metrópolis que establece una velocidad de enfriamento del tipo geométrico. El algoritmo concluirá cuando la temperatura llegue a 0 o se produzcan un determinado número de intentos sin lograr actualizar la solución actual con una solución nueva generada por el movimiento (cambio) realizado en la solución actual. Para que no se logre un cambio de solución, la nueva solución generada por el movimiento, no ha resultado mejor que la actual y la función de probabilidad de aceptación que define si realizar el cambio a una solución no mejor a la actual, dio como resultado que no se realice el cambio. Esta probabilidad depende de la temperatura, mientras menos temperatura menor probabilidad de realizar el cambio a una solución de menor calidad. Las rutas generadas en la solución inicial con el algoritmo PIA y las que se modificarán durante las iteraciones del algoritmo se conforman por zonas y todo cambio se realizara a nivel de zonas también. La función principal se muestra en Pseudocódigo 6.

```
InsertarEnMejorRuta(M1,M2,R, rutaSeleccionada)
        // rutaSeleccionada es la mejor ruta candidata a la inserción.
   Por cada ruta en R hacer
        SI (M1 pertenece a r) entonces
          por cada posición de un nodo en toda la ruta hacer
            rAux = insertar(p,M2,rutaSeleccionada) //insertar en cada posición (p) de
ruta
           Si( costo(rAux) < costo(rutaResultado)) entonces rutaResultado = rAux

        SINO SI (M2 pertenece a r) entonces
          por cada posición de un nodo en toda la ruta hace
            rAux = insertar (p,M1,rutaSeleccionada)
           Si( costo(rAux) < costo(rutaResultado) entonces rutaResultado = rAux

        SINO
          por cada posición de un nodo en toda la ruta hacer
            rAux = insertar (p1,M1, rutaSeleccionada)
            por cada posición de un nodo en rAux hacer
                  rAux = insertar (p2,M2, rutaAux)
                 Si( costo(rAux) < costo(rutaResultado) entonces rutaResultado = rAux
RETORNAR rutaResultado
```

Pseudocódigo 5 Método de Inserción de rutas

```
ProcedimientoPrincipal()
    Solución= crearSolucionInicial ( );
        REPETIR   //temperatura
            Repeticiones = 0;
            REPETIR
                (M1,M2) = seleccionarMinizonas;
                SolucionNueva = realizarMovimiento(M1,M2,solucion);
                &= FunObj(solucionnueva)- FunObj(solucion);
                SI(&<0) entonces solucion= solucionNueva;
                SINO
                    SI (random(0,1)<exp(-&/T) entonces
                        solucion=solucionNueva;
                    SINO contNoHayCambios =  contNoHayCambios
    + 1
            HASTA llegar al máximo de repeticiones
             T' = funcionMetropolis(T1);
        HASTA cumplir condición de parada//temperatura o no más cambios
```

Pseudocódigo 6 Algoritmo Recocido Simulado

Realizar Movimiento: Esta función evalúa que rutas pasan por las zonas del par O-D de zonas seleccionado aleatoriamente. Se procede a seleccionar aleatoriamente una de esas rutas para extraer ambas zonas de la ruta (función *modificarRutadeTraspase*) e insertar ese par de zonas en una ruta de un subconjunto de rutas (función *traspaseInserción)* seleccionadas aleatoriamente, este conjunto de rutas tiene un tamaño a determinar en la calibración del algoritmo (nRutasSim).

```
realizarMovimiento(M1,M2,R)
    rutaInsercion=traspaseInsercion (M1,M2, rutaSeleccionada, rutaDeTraspase);
    rutaModificada=modificarRutadeTraspase(M1, M2, rutaDeTraspase);
    R = R – {rutaDeTraspase} – {rutaSeleccionada} U {rutaInsercion } U
    {rutaModificada};
Retornar R;
```

Pseudocódigo 7Método de movimiento del algoritmo

TraspaseInsercion: Esta función verifica que rutas incluyen en su recorrido el par de zonas evaluadas anteriormente, aleatoriamente elije una de ellas (*rutaDeTraspase*) y luego elige aleatoriamente también un ruta del resto de rutas actuales (*rutaSeleccionada*) para insertar ese par en esa última ruta seleccionada. Finalmente se inserta el par de zonas en la ruta seleccionada (*rutaSeleccionada*) utilizando la

función *insertarEnMejorRuta*, la ruta generada luego de la inserción es el resultado de la función.

traspaseInsercion*(M1,M2,rutaSeleccionada,rutaDeTraspase)*
rutaDeTraspase = seleccionar ruta que pase por las zonas M1 y M2;
rutaSeleccionada = seleccionar ruta a insertar las zonas M1 y M2;
rutaResultado = insertarEnMejorRuta(M1,M2, rutaSeleccionada,rutaDeTraspase);
Retornar rutaResultado;

Pseudocódigo 8 Método 1 de modificación de solución

ModificarRutadeTraspase: Esta función, extrae el par de rutas evaluado anteriormente de la ruta seleccionada (*rutaSeleccionada*) en la función *traspaseInserción* y retorna la ruta modificada.

modificarRutadeTraspase*(M1,M2,rutaDeTraspase)*
rutaAux = Extraer M1 de rutaDeTraspase;
rutaModificada = Extraer M2 de rutaAux;
Retornar RutaModificada;

Pseudocódigo 9 Método 2 de modificación de solución

1.3 Ejecución del Algoritmo Recocido Simulado

A continuación se muestra una ejecución de los procedimientos, donde se muestra dos iteraciones y las dos formas de movimiento. Tal ejecución inicia desde que se obtuvo el resultado del algoritmo de creación inicial.

Datos de Entrada:

- CAP = 10%
- capBus = 80
- FL = 40
- distMin = 1km;
- distMax = 5km;
- P% = 60%
- Mmax = 10
- Vprom = 17km/h
- Vpromp = 23km/h

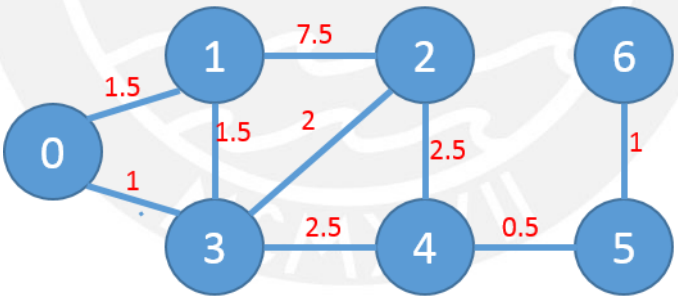
- CHH = S/.8.8.
- W1 = 0.6
- W2 = 0.4

Matriz de Demanda:

O-D	0	1	2	3	4	5	6
0	0	220	520	0	0	0	0
1	0	0	220	0	0	320	0
2	0	0	0	0	220	0	0
3	0	0	0	0	0	620	0
4	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0

Demanda Total: 2120

Grafo:



Resultado Solución Inicial:

Se muestran las rutas con su costo (distancia total), la demanda total cubierta y el valor de la función Objetivo obtenidas por el algoritmo PIA.

RutaZonas:

	Ruta 1	Ruta 2
nodosRZ	0 1 3 4 5	0 3 2 4 5
costo	6	6

Demanda Cubierta = 1700 personas

Función Objetivo = 1422.96 + 685 = s/.2107.96

Recocido Simulado: Se presenta una iteración del algoritmo Recocido Simulado.

- Temperatura Inicial = 1000
- Alpha = 0.905

El proceso inicia generando la lista de pares de zonas origen ordenada, de esta lista se selecciona un par aleatoriamente.

Se reinicia el proceso empezando con seleccionar un nuevo par O-D de la lista

3 – 5 : 620	0 – 2 : 520	1 – 5 : 320	0 – 1 : 220	1 – 2 : 220	2 – 4 : 220

Par O-D seleccionado = 0-1: 220

Se identifica cual ruta contiene a ambas zonas. La ruta seleccionada es:

Ruta 1

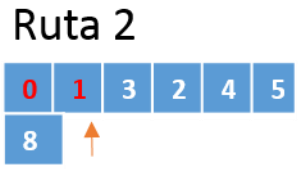
0	1	3	4	5
6				

Seleccionamos aleatoriamente la ruta a insertar el par. Al tener solo dos rutas la única opción es la ruta:

Ruta 2

0	3	2	4	5
6				

Se inician el proceso de inserción. La ruta al tener la zona 0 ya dentro de su recorrido sólo se busca la mejor posición para insertar la zona 1, dando como resultado la siguiente ruta.



Luego se extrae el par de zonas de la primera ruta seleccionada.

Al final se obtienen las siguientes ruta, demanda cubierta y nueva función objetivo.

Demanda Cubierta = 1920 personas

Función Objetivo = 2056.56 + 300 = S/. 2356.56

En este caso a pesar que se reduce la demanda sin cubrir, la nueva solución presenta un incremento del valor de la función objetivo, debido a que el costo de transportar a las personas a sus destinos se incrementa. Para estos casos se utiliza el criterio de Metrópolis para definir si la solución actual se actualiza con el nuevo resultado.

Capítulo 5 Experimentación Numérica

Realizar la experimentación numérica que compare los resultados del algoritmo PIA variado y Recocido Simulado es definido como el Objetivo Específico 3. La documentación, diseño y resultado de la experimentación numérica se muestran en el presente capítulo.

1 R4O3: Documentación, diseño y resultados de la experimentación numérica

La experimentación numérica se realizó siguiendo el Método de Múltiples Tratamientos y la Metodología de Contraste de Hipótesis explicados en la sección 2 del Capítulo 1. Con este método se logra comparar diferencias entre dos o más poblaciones de datos. Los datos a comparar en la experimentación son los valores de la función objetivo obtenidos por los algoritmos PIA variado y Recocido Simulado.

1.1 Diseño de la Experimentación.

Objetivo de la Experimentación: Determinar si la media de los valores de la función objetivo obtenidos por el algoritmo Recocido Simulado es mayor a la media de los valores de la función objetivo obtenidos por el algoritmo PIA.

Datos de Entrada: Se obtendrán 40 juegos de 10 resultados por cada algoritmo a comparar, de cada juego se utilizará el mayor valor obtenido, por lo tanto se obtendrán dos muestras (una por cada algoritmo) de 40 resultados.

Hipótesis: Las hipótesis a comprobar por cada prueba establecida son las siguientes:

- *Prueba de Kolgomorov*: Esta prueba debe realizarse para ambos algoritmos individualmente.
    - **Hipótesis Nula (H0)**: Los valores de la muestra siguen una distribución normal
    - **Hipótesis Alternativa (H1)**: Los valores de la muestra no siguen una distribución normal.

- *Prueba F*: Esta prueba debe realizarse entre ambos algoritmos.
    - **Hipótesis Nula (H0)**: Las varianzas son significativamente homogéneas.
    - **Hipótesis Alternativa (H1)**: Las varianzas son significativamente diferentes.

- *Prueba Z*: Esta prueba debe realizarse entre ambos algoritmos.
    - **Hipótesis Nula (H0)**: La media del Recocido Simulado es menor igual a la media del PIA variado.
    - **Hipótesis Alternativa (H1)**: La media del Recocido Simulado es mayor que la media del PIA variado.

1.2 Ejecución y Resultados de la Experimentación.

Elaboración de Datos de Entrada: Se seleccionó un área de Lima metropolitana, para el grafo formado por esta área se obtuvo 1 matriz de demanda de transporte. De esta muestra se obtendrán los resultados de los algoritmos para elaborar los 40 juegos de resultados.

Los valores de los Datos:

- CAP = 10%
- capBus = 80
- FL = 40
- distMin = 0km;
- distMax = 30km;
- P% = 80%
- Mmax = 500
- Vprom = 17km/h
- Vpromp = 23km/h
- CHH = S/.8.8.
- W1 = 0.6
- W2 = 0.4

Calibración de Algoritmos: Se definieron distintos valores para cada uno de los parámetros de los algoritmos, con los cuales se obtuvieron conjuntos de 10 resultados por cada distinto valor de parámetro, y para seleccionar el mejor valor de cada

parámetro se compararon las medias de los valores de la función objetivo de los conjuntos obtenidos. El valor del parámetro con el que se obtuvo el menor valor de la media es el escogido. A continuación se detallan los parámetros calibrados.

Algoritmo PIA variado:

- P%lista: Porcentaje de lista de pares O-D con mayor demanda.
- Nrutas: Tamaño del subconjunto de rutas para el método de inserción.

Algoritmo Recocido Simulado:

- Temperatura (T): Temperatura inicial del Algoritmo.
- Alpha: Factor de Enfriamiento.
- MaxNoCambios: Número máximo de intentos sin lograr cambios de solución.
- NRutasSim: Tamaño del subconjunto de rutas para el movimiento.
- Cambios por T: Cambios por temperatura.

Los valores obtenidos en la calibración se muestran a continuación:

Algoritmo PIA variado:

- P%lista = 30%
- Nrutas = 15

Algoritmo Recocido Simulado:

- Temperatura = 50000
- Alpha = 0.955
- MaxNoCambios = 400
- NRutasSim = 25
- Cambios por T = 10

La calibración de los algoritmos se muestra a detalle en el Anexo 3 Calibración de Algoritmos.

Resultados de la Experimentación:

Todas las pruebas se realizaron con un nivel de confianza de 95% y los resultados se detallan en el anexo 4 Experimentación Numérica.

Prueba de Kolmogorov: Esta prueba es realizada para ambos algoritmos por separado. En esta prueba se halla la máxima diferencia entre la probabilidad acumulada esperada y la probabilidad esperada de las 40 funciones objetivos y se compara con el valor crítico sacado de la tabla de distribución normal. Si la máxima diferencia es menor que el valor crítico se acepta la hipótesis nula y por tanto se concluye que los resultados del algoritmo siguen una distribución normal. Se muestra el resumen de los resultados obtenidos.

Algoritmo PIA variado

Media	850,297.67
Desviación Estándar	6,024.68
Varianza	36,296,749.13
Máxima Diferencia	0.094
Significancia de la prueba	0.05
Valor Crítico	0.21012

Algoritmo Recocido Simulado

Media	829,533.42
Desviación Estándar	6,739.86
Varianza	45,425,748.11
Máxima Diferencia	0.090
Significancia de la prueba	0.05
Valor Crítico	0.21012

Tabla 5 Resumen de Prueba Kolmogorov

La tabla 5 muestra que para ambos algoritmos la máxima diferencia es menor que le valor crítico. Por esta razón, se acepta la hipótesis nula para ambos algoritmos y se obtiene como resultado de esta prueba que ambos algoritmos cumplen una distribución normal.

Prueba F: Para esta prueba se halla la varianza de los resultados de ambos algoritmos, se halla la proporción entre ambas varianzas y se compara con el valor crítico de la tabla de distribución F con 39 grados de libertad. Si la proporción hallada es mayor al valor crítico se niega la hipótesis nula y se concluye que las varianzas de los algoritmos son distintas.

Prueba F	PIA V	Recocido
Media	850,297.67	829,533.42
Varianza	36,296,749.13	45,425,748.11
F	1.252	
Valor crítico para F (una cola)	1.704	

Tabla 6 Resumen de Prueba F

La tabla 6 muestra que le valor F obtenido es menor que el valor crítico para F. Por esta razón, se acepta la hipótesis nula y se obtiene como resultado de esta prueba que las varianzas de los algoritmos tienen varianzas distintas.

Prueba Z: Finalmente en esta prueba se calcula la función Z usando las medias de ambos algoritmos, este resultado es comparado con el valor crítico de Z de una cola. Si el valor calculado es mayor que el valor crítico se rechaza la hipótesis nula y se concluye que la media del algoritmo Recocido Simulado es menor.

Prueba Z	**PIA V**	**Recocido Simulado**
Media	850,297.67	829,533.42
Z	14.527	
Valor crítico de Z (una cola)	1.645	

Tabla 7 Resumen de la prueba Z

La tabla 7 muestra que el valor obtenido Z es mayor que el valor crítico Z. Por tanto, se rechaza la hipótesis nula y se obtiene como resultado de esta prueba que la media del Recocido Simulado es menor que la media del algoritmo PIA variado.

2 Conclusiones del Capítulo 5

El presente proyecto de fin de carrera propone implementar un algoritmo Recocido Simulado para solucionar el problema de Diseño de Rutas de Transito. Para comprobar su eficiencia se realizó la experimentación numérica comparándolo con el algoritmo PIA variado.

Según los resultados de la experimentación, al comparar las medias de los resultados obtenidos por ambos algoritmos, se concluye que la mejor media se obtiene por el algoritmo Recocido Simulado, es decir da mejores resultados.

Por lo tanto, es posible afirmar que el algoritmo Recocido Simulado propuesto es eficiente al resolver el problema de Diseño de Rutas de Transito.

Capítulo 6 Implementación de Aplicación

En el presente capítulo se presenta el Resultado correspondiente al Objetivo Número 5, implementar una aplicación donde se ejecutará el algoritmo. Este resultado está compuesto por el diseño de las vistas de la aplicación y explicación de su funcionalidad.

1 R5O4: Aplicación, la cual mostrará el conjunto de rutas resultado.

Las pantallas desarrolladas de la aplicación serán pantallas Web y se usará el API de Google Maps para realizar el graficado de las rutas en un mapa. A continuación se muestran las pantallas utilizadas.

1.1 Pantalla de Ejecución del Algoritmo

En esta vista se ingresarán los valores de los datos y parámetros que requiere el algoritmo para ser ejecutado. Se divide en dos áreas, en la primera se ingresan los valores de los parámetros del algoritmo ,detallados en el Capítulo 2 Sección 1.2.8, y en la segunda los valores de los datos que requiere el algoritmo explicados en el Capítulo 3 Sección 2.1. A continuación se presenta el prototipo de la vista.

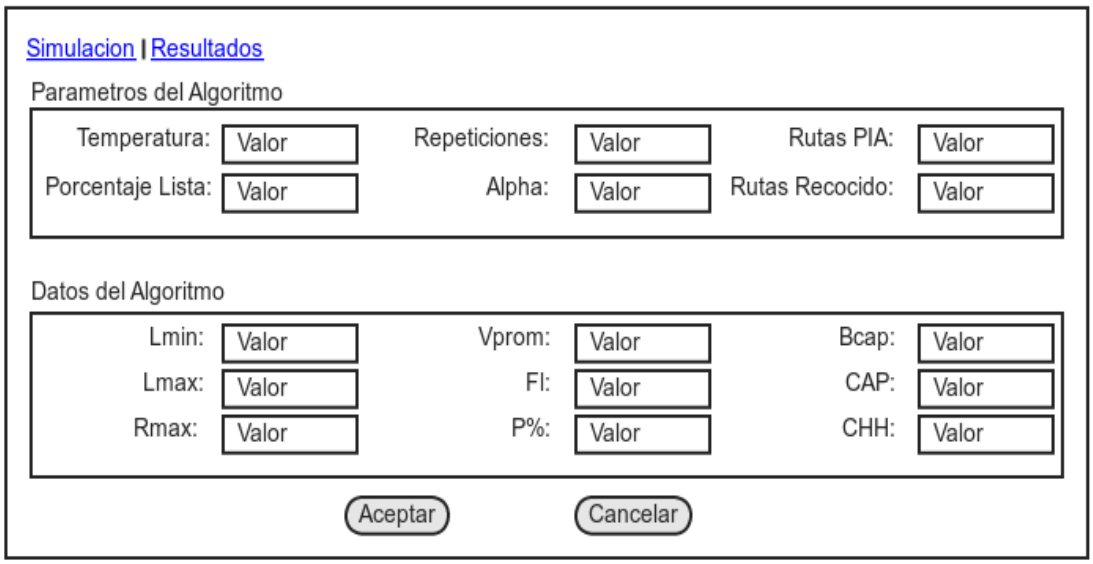


Figura 10 Prototipo Pantalla Simulación

1.2 Pantalla de Resultados

En esta pantalla se presentan las rutas obtenidas por el algoritmo, estas rutas son a nivel de zonas y por tanto en el mapa se mostraran el recorrido de zonas por donde pasan las rutas. Además se mostrarán el valor de la función objetivo de la solución, el número total de las rutas y la información de una ruta en específico, en caso se seleccione una de ellas. Todos los valores de la pantalla y las rutas a mostrar son leídos desde la base de datos. A continuación se presenta el prototipo de la vista.

Figura 11 Prototipo Pantalla Resultados

2 Ejecución del Algoritmo

El procedimiento de ejecución el algoritmo y mostrar los resultados se producen en 3 etapas. La primera etapa es ingresar y almacenar los valores de parámetros y datos que se requieren, la segunda es el proceso del algoritmo desarrollado y finalmente la etapa de mostrar la solución. Toda consulta a base de datos que se realice desde las pantallas web se realizan utilizando lenguaje PHP y las funcionalidades de las pantallas y las gráficas son realizadas utilizando lenguaje HTML y Javascript.

En la primera etapa, se ingresan los valores de los datos y parámetros mostrados en la pantalla de Ejecución del Algoritmo, estos son cargados en las variables definidas en el archivo .html correspondiente de la pantalla de Ejecución del Algoritmo. Teniendo los valores necesarios, al presionar el botón Aceptar, haciendo uso de funciones en php, se inicia una función que envía los valores a la base de datos mediante una sentencia para ser almacenados. Inmediatamente después, mediante una función en javascript, se ejecuta el algoritmo desarrollado, el cual se encuentra empaquetado en un archivo de extensión .exe. De esta manera se da inicio a la

segunda etapa. Esta etapa empieza leyendo desde la base de datos todos los valores previamente ingresados y almacenados, junto con la información del grafo de zonas y la matriz de demanda de transporte, esta información es cargada en las estructuras definidas para este propósito.

Luego se inician los procedimientos diseñados del algoritmo Recocido Simulado. Al finalizar este, se obtiene una solución compuesta por el conjunto de rutas a nivel de zonas obtenidas y el valor de la función objetivo de la solución, esta solución se almacena en la base de datos y se da inicio a la última etapa. En la pantalla de resultados, se consulta desde la base de datos la solución obtenida y las zonas de tránsito de Lima Centro, las consultas a base de datos, y mediante las funciones de la API de Google Maps es posible mostrar el mapa, su composición en zonas de tránsito y graficar las rutas obtenidas. Cada zona tiene asociado un punto llamado centroide y es partir de estos centroides que se van uniendo las zonas que incluyen cada ruta para ir mostrando el recorrido de la ruta, gracias a esto se puede pintar las aristas que componen cada ruta en el mapa. El valor de la función objetivo y la cantidad de rutas obtenidas son consultados a la base de datos para mostrarlos en la pantalla de Resultados. Adicionalmente si se desea observar una ruta en especial, se utiliza el combobox “Escoger Ruta”, selecciona la ruta que se desea mostrar y la composición de la ruta junto con sus datos son leídos desde la base de datos, para luego ser mostrados en la pantalla de Resultados. Además se realiza una consulta en la base de datos, para que indique cuales son los pares de la zona seleccionada que poseen la mayor cantidad de demanda de transporte, cada par de zonas es mostrado con un marcador.

Capítulo 7 Recomendaciones, trabajos fututos y conclusiones

1 Recomendaciones y Trabajos Futuros

En esta sección se presentan las recomendaciones y trabajos futuros con la finalidad de proponer mejoras a futuro para solucionar el problema planteado.

En primer lugar, se mencionó que la información de demanda de transporte obtenida fue obtenida en base a zonas de tránsito y que las rutas obtenidas por el algoritmo también serían en base a zonas. Sería recomendable tener esta información de demanda de transporte en base a zonas de área más pequeña, el algoritmo podría construir rutas de transporte con un recorrido más detallado (a la vez con un tiempo de viaje más exacto) y cubriendo más eficientemente la demanda de transporte. Además si se transformaran estas rutas a rutas conformadas por calles, se obtendría al máximo nivel de detalle las rutas de transporte con el tiempo de viaje total real. Esta transformación podría incluirse dentro del algoritmo para que el cálculo de la función objetivo sea más real respecto al primer término de la función que mide el tiempo de viaje total. Sin embargo ya que el área geográfica de Lima es extensa, el grafo de calles que se obtendría tendría una enorme cantidad de nodos y aristas, los cual implicaría gran cantidad de tiempo el procesar estas rutas a nivel de calles sea dentro o fuera del algoritmo.

El presente proyecto de tesis se concentra en construir rutas de transporte de público. Futuros proyectos pueden utilizar como base este proyecto o complementarse con este para implementar un algoritmo que intente resolver el problema de localización de paraderos (BTLP), así como también para los distintos problemas que también son parte de la familia del problema padre BTRNDP. Entre estos problemas se identifican al establecimiento de frecuencias de salida de buses, determinación de horarios y planificación de buses y choferes. Entre estos, el más directo a aplicar es el de establecimiento de frecuencias de salida de los buses para rutas. Además la base de datos también ha sido diseñada para almacenar y trabajar con datos que se requieren para resolver estos dos problemas y puede ser utilizada también. Por otro lado, se ha

establecido todo un flujo de carga para guardar la estructura vial del distrito de Lima y Callao, esta información puede ser utilizada para los proyectos que requieran de esta.

En trabajos futuros también puede realizarse modificaciones al algoritmo para mejorarlo y obtener aún mejores resultados, modificando el movimiento utilizado en el algoritmo o añadiendo nuevos movimientos para que trabaje con más de un solo movimiento, incluyendo otras restricciones a considerar o tratando de obtener otros objetivos, como el minimizar el número de transferencias (utilizar más de una ruta) para llegar a un destino. También se puede buscar comparar el algoritmo con otro algoritmo meta-heurísticos como el algoritmo genético para medir aún más su eficiencia.

2 Conclusiones

En esta sección se presentan las conclusiones del presente proyecto de fin de carrera al haber completado todos los objetivos del proyecto.

Durante el proyecto una de las primeras cosas que se identificó es que si bien el ciudadano peruano tiene el derecho del libre acceso a la información, el acceso a esta información puede ser dificultoso en sentido que no se tiene un lugar de acceso sencillo donde se pueda encontrar la información requerida y que esta podría no estar integrada y actualizada. Un ejemplo es la información requerida en este proyecto, para obtener la información de demanda de transporte y las zonas de transito utilizadas, fue necesario preguntar y solicitar a diversas entidades hasta encontrar las entidades que tuvieran la información y las peticiones realizadas fueran recibidas y atendidas. Una vez que se obtuvo la información, luego de analizarla se notó que las entidades tenían versiones distintas.

A raíz de esto, la base de datos diseñada donde se almacena esta información obtenida junto con la información de la estructura vial de Lima Metropolitana, podría ser de mucha ayuda para los futuros proyectos que necesiten de esto tipo de información.

Por otro lado el problema de Diseño de Rutas de Tránsito, es un problema muy complejo que puede incluir muchas variables y restricciones, por lo cual ha sido muy importante analizar cuales considerar de manera que sean las más importantes para el contexto del proyecto y contribuyan más con el objetivo del mismo. Además durante la

formulación de un problema de este nivel de complejidad, es importante determinar los factores a considerar y sobretodo establecer una función objetivo que logre expresar de manera correcta el objetivo deseado, para lo cual es necesario un exhaustivo análisis y tener bien claro lo que se desea optimizar. También es importante si se desea resolver un problema en un escenario real, obtener los valores de los datos que usan las restricciones y supuestos planteados para seguir siendo fieles al escenario real.

Finalmente se compararon las soluciones obtenidas por el algoritmo Recocido Simulado y el algoritmo PIA variado y como resultado de la experimentación numérica las soluciones del Recocido Simulado son mejores que las obtenidas por el algoritmo PIA variado. Por lo tanto se puede concluir que el algoritmo Recocido Simulado presenta una mejor solución que el algoritmo PIA variado para el presente problema.

Referencias bibliográficas

Agencia de Cooperación Internacional del Japón (JICA)

2013 Encuesta de Recolección de Información Básica del Transporte Urbano en el Área Metropolitana de Lima y Callao, Informe Final (Resumen). Perú.

Agencia de Cooperación Internacional del Japón (JICA); Consejo de Transporte Lima y Callao; Ministerio de Transportes Y Comunicaciones de la República del Perú.

2005 Plan Maestro de transporte Urbano para el Área Metropolitana Lima y Callao en la República del Perú. Perú.

Alsuwaiyel, M. H.

1998 “Algorithms: Design Techniques and Analysis”. Londres: World Scientific Pub Co Inc.

Antonio Mauttone; María E. Urquhart.

2009 “A route set construction algorithm for the transit network design problem” Montevideo: Computers & Operations Research, Vol. 36, pp. 2440 – 2449.

Banco Central de Reserva del Perú

2011 El tiempo es dinero: Cálculo del valor social del tiempo en Lima Metropolitana para usuarios del transporte Urbano. Estudios Económicos. Vol. 20, pp. 73-86.

Berenson, Mark; Levine, David; Krehbiel, Timothy
2006 "Estadística para Administración". Pearson Educación. USA.

Bielich Salazar, Claudia
2009 La Guerra del Centavo una mirada actual al transporte público en Lima Metropolitana. Lima.

Blum, Christian; Roli, Andrea.
2003 "Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison".

Bradley, Hax; Magnant
1997 Applied Mathematical Programming. Addison-Wesley

Code::Blocks. Consulta 12 de Julio de 2014
<http://www.codeblocks.org/>

Coley A, David.
2003 "An introduction to genetic algorithms for scientists and engineers" World Scientific.

Comparación de Múltiples Tratamientos. Consulta: 6 de noviembre de 2013.
<http://dta.utalca.cl/estadistica/ejercicios/interpretar/Metodos/anova.pdf>

Consejo Metropolitano de Lima.
2012 Ordenanza N° 1599. 17 de Abril.

Cormen, Thomas; Leirsen, Charles ; Rives, Ronald; Stein, Clifford
2009 Introduction to Algorithms. Thid Edition. United States Of America. Massachussets Institute of Tecnology.

Curso General de Estadistica. Consulta 30 de septiembre de 2014.
<http://virtual.uptc.edu.co/ova/estadistica/>

Dorigo, Marco; Stüzle, Thomas
2004 Ant Colony Optimization.England: Massachusetts Institute of Tecnology.

Eugene L. Lawler
1976 Combinatorial optimization: Networks and Matroids. New York: Holt, Rinehart and Winston.

Fan, Wei, B. Machemehl, Randy
2004 "Optimal Transit Route Network Design Problem: Algorithms, Implementations, and Numerical Results",Texas.

Fan, Wei; B. Machemehl, Randy
2006 Using a Simulated Annealing Algorithm to Solve the Transit Route Network Design Problem, Journal of Transportation Engineering , Vol. 122, pp. 122-132.

Free dictionary. Consulta: 22 de Septiembre del 2013
<http://encyclopedia2.thefreedictionary.com/Computer+Mathematics>

Goldberg D. E.; Holland J. H.
1988 Genetic Algorithms and Machine Learning, Machine Learning, Vol 3, pp. 95-99. Holanda.

Guihaire ,Valérie; Hao, Jin-Kao
2008 Transit Network Design And Scheduling: a Global Review. Transportation Research Vol 42, Num 10, pp 1251-1273

Hossain Poorzahedy; Farshid Safari
2009 "Transit Network Design Problem: Review". Journal of Transportation Engineering. Vol 135, Num. 8, pp. 491-505

Huapu, Lu
2007 "Complexity of Public Transport Networks" TSINGHUA SCIENCE AND TECHNOLOGY

IBM – SPSS Statistics Standard. Consulta: 29 de agosto de 2013.
<http://www-03.ibm.com/software/products/pe/es/spss-stats-standard/>

Jens Vygen
2005 Approximation Algorithms for Facility Location Problems. Research Institute for Discrete Mathematics, University of Bonn Lenn´estraße 2, Germany.

Karthryn A. Dowsland; Berlamino Adenso Díaz
2003 Diseño de Heurística y Fundamentos del Simulated Annealing. Inteligencia Artificial. Revista Iberoamericana de Inteligencia Artificial. Vol 7, Num 19, pp. 93-102.

Kepaptsoglou, Konstantinos; Karlaftis, Matthew
2009 Transit Route Network Design Problem: Review, Journal of Transportation Engineering, Vol. 135, Num. 8, pp. 491-501.

Kirkpatrick, S.; Gelatt, C. D.; Vecchi, M. P.
1983 Optimization by Simulated Annealing. Science Vol 220 Num. 4598 pp. 671–680.

Mc Carthy, John
2007 “What is artificial intelligence”. Consulta: 30 de agosto de 2013. <http://www-formal.stanford.edu/jmc/whatisai/node1.html>

M. Grotschel; L Lovász
1995 Handbook of Combinatorics. Amsterdan: Elsevier Science B.V.

Ministerio de Transportes y Comunicaciones
2007 Decreto Supremo N° 017-2007-MTC.

Municipalidad Metropolitana de Lima
2005 Ordenanza N°1613. 26 de Junio.

Municipalidad Metropolitana de Lima, Reforma de Transporte, Consulta 12 de Septiembrede 2014.
<http://www.munlima.gob.pe/reformatransporte/ejes/sistema-integrado-de-transporte>

Open GIS Consortium
1999 OpenGIS Simple Features Specification for SQL. 5 de Mayo.

OpenStreetMap. Consulta 10 de noviembre de 2013
< http://www.openstreetmap.org/about >

Papadimitriou, Christos; Steiglitz, Kenneth
1982 Combinatorial optimization, Algorithms and complexity. USA: Prentice-Hall

Perú 21. Consulta 22 de octubre de 2013
<http://peru21.pe/actualidad/lima-279-accidentes-transito-fatales-lo-que-va-ano-2141165>

PgAdmin III Consulta 12 de julio de 2014
< http://www.pgadmin.org/>

Poorzahedy, Hossain; Safari, Farshid
2011 An Ant System application to the Bus Network Design Problem: an algorithm and a case study.

Reza Ghanbari- Nezam Mahdavi-Amiri
2011 Solving bus terminal location problems using evolutionary algorithms. Applied Soft Computing Vol. 11, Número 1, pp. 991–999.

Rolón, Rocío
2009 Manejo del Software TRANSCAD especializado en transporte, logística y operaciones. LEMAC Centro de Investigaciones Viales

Ruiz, Arturo; Rojas, Falcó
2009 "Herramientas Estadísticas" Madrid. Universidad Pontificia Icai-Ica de Comillas.

Russell, Stuart J.; Norvig Peter
2004 Inteligencia Artificial: Un Enfoque Moderno. 2da Edición, Madrid: Pearson Education S.A.

Sadrsadat, Hadi; Poorzahedi, Hossein; Haghani, Ali; Sharifi, Elham
2003 Bus Network Design Using Genetic Algorithm. Tesis de Maestría de Ciencias, Ingeniería de Transporte. Sharif University of Tecnology.

Schöbel, Anita
2011 "Line planning in public transportation: models and methods", OR Spectrum. Vol. 32, Número 3, pp. 491-510.

Sistema Transcad. Consulta: 19 de Septiembre del 2013
<http://www.caliper.com/transcad/analisis.htm>

Thompson, Ian; Bull, Alberto
2002 "La congestión del tránsito urbano: causas y consecuencias económicas y sociales", Revista de la Cepal, pp.109-121.

Yu, B.; Yang, Z.; Cheng, C.; Liu, C.
2005. Optimizing bus transit network with parallel ant colony algorithm. Proceedings of the Eastern Asia Society for Transportation Studies. Vol.5, pp. 374-389.

Zhao, Fan; Gan, Albert
2003 Optimization of Transit Network to Minimize Transfers, Research Office Florida Department of Transportation. Miami