UNIVERSIDAD AUTÓNOMA DE NUEVO LEÓN

FACULTAD DE INGENIERÍA MECÁNICA Y ELÉCTRICA

MÉTODO DE SOLUCIÓN DE RUTA ENTRE CIUDADES PARA REPARTO DE MUESTRAS

POR

AZCARIE MANUEL CABRERA CUEVAS

COMO REQUISITO PARCIAL PARA OBTENER EL GRADO DE MAESTRÍA EN LOGÍSTICA Y CADENA DE SUMINISTRO

SEPTIEMBRE, 2018

UNIVERSIDAD AUTÓNOMA DE NUEVO LEÓN

FACULTAD DE INGENIERÍA MECÁNICA Y ELÉCTRICA

SUBDIRECCIÓN DE ESTUDIOS DE POSGRADO

MÉTODO DE SOLUCIÓN DE RUTA ENTRE CIUDADES PARA REPARTO DE MUESTRAS

POR

AZCARIE MANUEL CABRERA CUEVAS

COMO REQUISITO PARCIAL PARA OBTENER EL GRADO DE MAESTRÍA EN LOGÍSTICA Y CADENA DE SUMINISTRO

SEPTIEMBRE, 2018

Universidad Autónoma de Nuevo León

Facultad de Ingeniería Mecánica y Eléctrica

Subdirección de Estudios de Posgrado

Los miembros del Comité de Tesis recomendamos que la Tesis «Método de solución de ruta entre ciudades para reparto de muestras», realizada por el alumno Azcarie Manuel Cabrera Cuevas, con número de matrícula 1883868, sea aceptada para su defensa como requisito parcial para obtener el grado de Maestría en Logística y Cadena de Suministro.

El Comité de Tesis

Jania Astrid Saucedo

Dra. Jania Astrid Saucedo Martínez

Asesor

Dr. José A. Marmolejo Saucedo

Revisor

MLyCS R. Igor Sánchez Castro

Revisor

Vo. Bo.

Dr. Simón Martínez Martínez

Subdirector de Estudios de Posgrado

San Nicolás de los Garza, Nuevo León, septiembre 2018

A mi familia: Abigail, Ayari, Jessica, Lucía, Manuel y Vielka.

Índice general

Índice de figuras

Índice de tablas

Índice de pseudocódigos

Agradecimientos

Agradezco a la Comisión Nacional de Ciencia y Tecnología, a la Universidad Autónoma de Nuevo León y a la Facultad de Ingeniería Mecánica y Eléctrica, que apoyaron y financiaron la presente investigación, y sin quienes la anterior no habría sido posible. De la misma manera es imprescindible agradecer el apoyo y guía de la Dra. Jania Astrid Saucedo Martínez durante el proyecto, así como la ayuda del Dr. José Antonio Marmolejo Saucedo para la divulgación del mismo y la contribución de ideas para el desarrollo, y al M. en L. y C.S. Rogelio Igor Sánchez Castro por el aporte de información para el caso de estudio.

Me permito agradecer a mi familia: a mis padres que siempre cuentan con las palabras y los medios adecuados para estimular mi búsqueda de crecimiento profesional y personal, a mis hermanas Abigail y Vielka quienes siempre serán mi ejemplo a seguir y motivación, y, por último, pero no menos importante, a mi esposa Jessica, por sus acciones, que llevaron a esta investigación a denodarse durante cada etapa.

Resumen

Azcarie Manuel Cabrera Cuevas.

Candidato para obtener el grado de Maestría en Logística y Cadena de Suministro.

Universidad Autónoma de Nuevo León.

Facultad de Ingeniería Mecánica y Eléctrica.

Título del estudio: Método de solución de ruta entre ciudades para reparto de muestras.

Número de páginas: 115.

Objetivos y método de estudio: En el presente documento se propone un método de solución aproximado para resolver problema de distribución de una empresa de mercadotecnia, que busca posicionar cinco vehículos para visitar ciento cincuenta puntos de interés repartidos en veintidós ciudades de la República Mexicana, generando una ruta que cumpla con lo anterior, para dar a conocer un nuevo producto de una embotelladora. Una vez que un vehículo se encuentre en algún punto de interés, este será abastecido con muestras de un producto promocional, fungiendo como imagen de la marca que promoverá, en otras palabras, los vehículos no transportarán las muestras promocionales.

Se estudiaron algunos posibles planteamientos de solución del problema anterior, acotando el enfoque hacia la perspectiva matemática, que resultó presentar un

reto significativo debido a que el modelo del problema del agente viajero múltiple se encuentra catalogado dentro del conjunto **NP**. Se propuso usar un método de solución asignación primero, ruta segundo; resolviendo el problema de asignación mediante una versión modificada del modelo del problema de las p medianas para calcular cada ruta mediante un algoritmo metaheurístico fundamentado en los algoritmos de optimización basados en colonia de hormigas.

Contribuciones y conclusiones: Se concluyó que el método propuesto es una alternativa viable y asequible para las pequeñas empresas que cuenten con un presupuesto limitado, ya que requiere de una inversión mínima para su uso, con resultados aceptables en un tiempo rápido comparado con los demás algoritmos que atacan el mismo problema. El primero reportó tiempos totales menores a treinta y cuatro minutos para instancias de cuatrocientos cuarenta y tres nodos (y menores), para la solución de tales instancias con el número de vehículos indicado en el caso de estudio (cinco). Asimismo, se identificó que si no se cuenta con una licencia de solver, es factible resolver el modelo exacto en «la nube» (para cinco vehículos) si el problema es menor a 65 nodos aproximadamente, pero, para instancias mayores el método propuesto resulta una mejor opción. Con respecto a la cadena de suministro, se aporta una alternativa con las características anteriormente mencionadas para los participantes de esta, que podrán contar con una herramienta que coadyuve en el posicionamiento de los productos (o recursos) desde donde son producidos (o desde donde están están ubicados) hasta donde son requeridos, en este caso para fines publicitarios.

Firma del asesor: _____

Dra. Jania Astrid Saucedo Martínez

Capítulo 1

Introducción

La presente investigación se cimienta sobre el enfoque de **cadena de suministro**, que considera distintos aspectos de la satisfacción de las órdenes provistas por las empresas hacia sus clientes, tomando en cuenta que para lograr ofrecer productos o servicios al cliente final de una manera eficiente es necesaria una visión holística de todos los componentes que participan, medían e **injieren colateralmente en el flujo** financiero, **de productos**, información y servicios, tangibles así como intangibles desde los extractores del recurso en su estado natural **hasta su término fungible**.

Los elementos bajo la perspectiva anteriormente mencionada, comprenden el aprovisionamiento para la fabricación del producto o servicio, su transformación, reparación o manufactura y la distribución de estos a través de una red de distribuidores y/o vendedores a detalle, entregando el producto o servicio hasta el extremo en el que está el usuario final, existiendo en este ciclo un flujo bidireccional (ilustrado en la Figura 2.1).

Por otra parte, la logística obedece al cumplimiento en tiempo y forma del producto o servicio; y aunque se maneja en distintas áreas de una empresa en este trabajo se revisará una problemática específica: la distribución con fines de mercadotecnia, y específicamente de promoción, a sabiendas de que el enfoque aludido

primeramente, que agrupa distintos procesos pertenecientes a los anteriores (aunados a otros más), contempla la correlación directa que tienen dichos actos en la **satisfacción de la demanda y en su propagación**: labor imprescindible en la búsqueda de la ventaja competitiva a nivel cadena de suministro.

Se detalla el caso de una empresa productora de bebidas que desea dar a conocer un nuevo producto mediante el reparto de muestras gratuitas **a través de una empresa de mercadotecnia**. La empresa previamente mencionada elabora un plan de trabajo para la distribución de producto promocional, que precisa el ruteo de los vehículos que servirán como puntos de reparto del producto, con la imagen de la empresa a publicitar.

Es importante señalar que la empresa ya cuenta con rutas previamente diseñadas para aprovisionar diferentes puntos de venta con su variedad de productos «habituales», considerando una demanda específica, y que para avenir las demandas de sus productos existentes y nuevos recurren, a través de la tercerización, a proyectos específicos de promoción, especificando con poco tiempo de anticipación los puntos dónde se desea generar la misma y que, además, los anteriores pueden estar sujetos a cambio sin previo aviso, volviendo complejo el proceso de planeación.

Es en el punto anterior en el que se propone una herramienta de trabajo, con la que se pretende minimizar la distancia de los recorridos en apoyo a la elaboración del plan de trabajo, por medio de una interfaz útil y fácilmente manipulable para el ingreso de parámetros y obtención de una respuesta ágil.

1.1 Objetivo

Diseñar una ruta entre puntos de interés para resolver el problema de distribución de una empresa de mercadotecnia mediante una herramienta matemática que logre minimizar la distancia recorrida.

1.2 Hipótesis

Se minimizará la distancia recorrida mediante el desarrollo de una herramienta matemática para el diseño de una ruta entre puntos de interés para resolver el problema de una empresa de mercadotecnia.

1.3 Justificación

Con el aumento en la diversificación de segmentos de mercado, cada vez con mayor grado de especificaciones en razón de patrones y preferencias de consumo, cerca de la mitad del precio que paga un cliente por un producto o servicio es a cuenta de las actividades que conllevan hacer llegar este último hasta el cliente (Dent, 2008).

La distribución de los productos tangibles es uno de los tópicos más importantes dentro de la cadena de suministro, Chopra y Meindl (2013) indican que los costos relacionados a la distribución forman el 10.5 % de la economía norteamericana además de representar el 20 % del costo de fabricación, luego entonces, lograr la ubicación de manera eficiente con los recursos disponibles se vuelve una tarea compleja, en función de su correcto planteamiento acorde a los objetivos de la empresa.

Para atenuar el costo y mejorar la distribución directamente, se atenderá la problemática del transporte, sin embargo, de acuerdo al tipo de empresa que se analice, distintas opciones, como lo es la adquisición y/o arrendamiento de equipo y/o personal que se encargue de la investigación y desarrollo de las mejoras en este rubro, pueden llegar a ser impensables en términos de inversión para algunas empresas.

La Secretaría de Economía (SE, 2008) categoriza las empresas nacionales en cinco segmentos con respecto a su tamaño, como se muestra en la Figura 1.1; em-

prendedores (que son personas que están desarrollando, creando o consolidando una empresa), microempresas, PyMEs (Pequeñas y Medianas Empresas), Gacelas (microempresas y PyMEs que presentan actividad mayor a la media) y empresas Tractoras (empresas de gran tamaño consolidadas en el mercado).

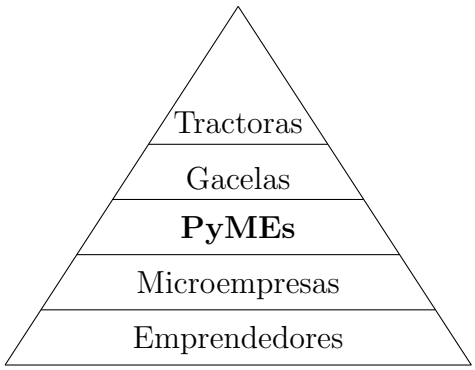


Figura 1.1: Clasificación de empresas en México
Fuente: Secretaría de Economía (2008)

La empresa que se verá en el caso de estudio se encuentra en el rango de las PyMEs, y como lo mencionan Olivos *et al.* (2015), representan el 4.2 % de las empresas en México, siendo consideradas como la piedra angular de la economía del país, porque contribuyen con aproximadamente 31.5 % y 37 % del empleo y Producto Interno Bruto, respectivamente.

Los autores anteriores identifican también la omisión de planificación bajo una estructura organizada en dicha categoría. Es este aspecto lo que limita el presupuesto financiero para los proyectos como los mencionados en el presente documento, y que por lo tanto deben de ser desarrollados considerando no solamente las mejores opciones, si no también, las que resulten alcanzables en los términos mencionados precedentemente.

Se propone una metodología para el análisis de mejores alternativas de planeación en el transporte, alcanzable para las empresas que cuenten con un presupuesto limitado (PyMEs).

1.4 Metodología de investigación

Como primer paso para este trabajo se revisarán las condiciones iniciales del proyecto; los activos disponibles y el alcance deseado, para después explorar las alternativas de solución, tanto las ofrecidas en un principio por la empresa interesada, así como las generadas por el análisis del caso.

Una vez identificado el problema, se buscarán los métodos usados desde el estado del arte en la materia hasta las investigaciones más prometedoras, en función de los resultados obtenidos en las mismas, para contrastar las distintas aproximaciones e inferir la(s) mejor(es) para considerar las mejores prácticas en el presente trabajo.

Como etapa final se considera el análisis de los resultados obtenidos y las posibles áreas de oportunidad para una investigación futura.

1.5 Estructura de la tesis

En el presente capítulo se detalló el objetivo de investigación y la hipótesis, seguidos de la justificación que alentó la investigación, para después explicar la metodología de la misma.

El capítulo 2 se referirá a los conceptos clave del enfoque de cadena de suministro, delimitando su aplicación para el transporte y la mercadotecnia, revisando la problemática de diseño de rutas estudiando la reseña de las investigaciones existentes más representativas bajo la perspectiva matemática, para pasar con algunos métodos de solución vigentes (exactos y aproximados), hasta la fecha de publicación, para la solución del problema en cuestión en el capítulo 3.

Una vez efectuada la revisión de literatura pertinente se describirán los pasos para resolver el problema central de la investigación en el capítulo 4, describiendo las condiciones en las que se puede aplicar. Efectuando experimentación computacional

y reportando los resultados de la metodología en el capítulo 5, con el fin de validarlo. El caso de estudio se desarrollará en el capítulo 6, concluyendo con un análisis final de todo lo anterior en el capítulo 7.

Capítulo 2

Marco Teórico

Con la intención de delimitar el alcance de la investigación, en este capítulo se comenzará por definir el concepto de cadena de suministro y la relación que tiene esta con la distribución, así como la del transporte y la mercadotecnia con la misma, para después detallar los métodos de diseño de ruteo de vehículos, aludiendo los métodos cuantitativos para el desarrollo de la propuesta de solución, recurriendo específicamente a la modelación matemática para la solución del problema descrito en el caso de estudio, su planteamiento, y análisis de factibilidad para obtener el mejor resultado posible, analizando los modelos del problema del agente viajero (y su variación con múltiples agentes) y el problema de ruteo de vehículos. Para finalizar examinando los métodos de solución para el modelo matemático y su factibilidad computacional de solución.

2.1 Logística y cadena de suministro

La Real Academia Española (2014) define **logística** partiendo del griego *logistikós*, que es «lo relativo al cálculo», y menciona que este concepto gira en torno a la organización de una empresa o un servicio, enfatizando la distribución. De manera que pareciera que la connotación de la palabra es estrecha, no obstante, logística implica colocar el recurso en el lugar, al momento, condiciones y costo correctos. Ésta

concepción ha madurado a través del tiempo y es común evocar su matiz militar como talante estratégico, originando tratados como el expuesto por Jomini (1838), quien fuera asesor marcial en la época napoleónica.

Una de las principales tergiversaciones es que **cadena de suministro** es un concepto más para referirse a logística, siendo que la primera está formada por un conjunto de procesos presentes a lo largo de todo el ciclo de un producto o servicio, desde que este es extraído en su estado natural hasta su consumo por el usuario final, abarcando etapas como el abastecimiento, manufactura, **distribución** y venta, por mencionar algunas (Ayers, 2000).

Es el usuario final el que dicta los requerimientos que darán forma a la cadena, en otras palabras, las características demandadas por los consumidores son la razón de ser de la cadena de suministro. En este sentido, la definición que brindan Chopra y Meindl (2013) acerca de este concepto es que está formada por todas aquellas partes involucradas en la satisfacción de una solicitud de un cliente, de manera directa o indirecta y que incluyen, además de otros aspectos y procesos, al desarrollo de nuevos productos, **la mercadotecnia**, y el servicio al cliente.

Por su parte Sosa (2012) menciona, sobre la concepción anteriormente mencionada, que se genera **valor agregado** a los usuarios finales a través de la **integración** de los procesos desde los proveedores iniciales, entregando productos y servicios. El flujo en la cadena es **bidireccional**, constituyéndose de flujo físico, de información y financiero (Figura 2.1).

Bajo este tenor Christopher (2011) sostiene que la cadena de suministro es una red de organizaciones conectadas e interdependientes, y que por medio del **trabajo cooperativo** les es posible controlar, administrar y mejorar los flujos previamente mencionados desde los proveedores hasta los usuarios finales. Una manera tentativa de visualizar esta última aseveración se muestra en la Figura 2.2, en donde los distintos participantes pueden estar dentro de cadenas de suministro ajenas a la industria o en conjunto con empresas que compiten por un mercado común.

Por lo que podemos concluir que la cadena de suministro es un sistema complejo con abundantes áreas de oportunidad (y de riesgo) que atienden a preguntas como: ¿Cómo medir el sistema? (métricas), ¿quién terminará ganando? (políticas y poder de negociación de los participantes), ¿quién puede ver qué y con qué rapidez? (visibilidad), **¿qué pasará si...? (incertidumbre)**, y esto sin mencionar la complejidad (unidades de medida, **demandas divergentes** por parte de los clientes, nuevos canales de distribución), etc. y las operaciones globales.

Durante el diseño de la cadena de suministro, y en concreto de la **distribución**, es necesario considerar la configuración del ruteo del producto para apoyar los objetivos estratégicos (Chopra y Meindl, 2013).

2.2 Transporte y mercadotecnia

Dentro del ámbito de la distribución, el transporte (del latín *trans*: más allá y *portare*: llevar) es una de las directrices más importantes debido a que los productos rara vez se consumen en el lugar donde son producidos, en consecuencia se busca trasladar mercancías desde aquellos puntos en los cuales su utilidad marginal para su uso o consumo es relativamente baja a aquellos puntos donde su utilidad es mayor, en otras palabras busca abastecer puntos geográficos que tienen escasez de ciertos

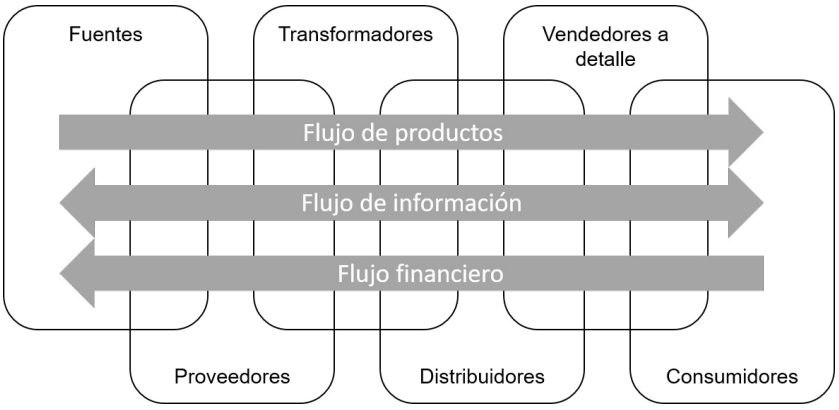


Figura 2.1: Vertiente de cadena de suministro
Fuente: Elaboración propia con información de Christopher (2011)

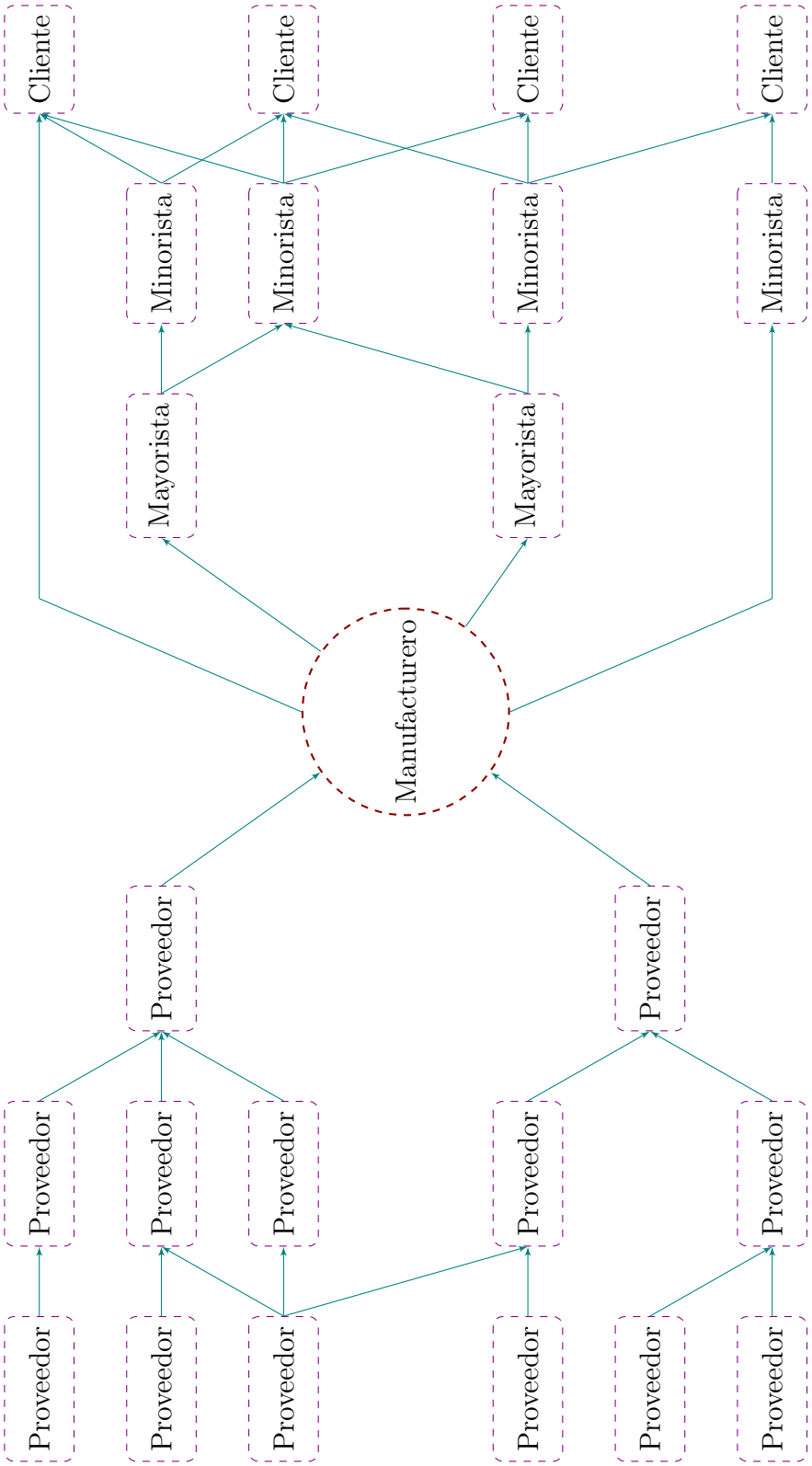


Figura 2.2: Visualización de la red de cadena de suministro
Fuente: Elaboración propia con base en la definición de Christopher (2011)

productos o servicios. Bonavia (1956) asevera que el transporte cumple su misión cuando beneficia al consumidor extendiendo el mercado. Por su parte Rodrigue *et al.* (2013) señalan que busca vencer las restricciones del espacio, conformadas por elementos tanto físicos como humanos, siendo algunos de estos: la **distancia**, el tiempo, la topografía y las divisiones administrativas. Según los autores mencionados la capacidad (del modo de transporte), la infraestructura y la naturaleza de lo que se trasporta, desempeñan un rol significativo, no obstante, el caso de estudio que se trabajará considera solamente la movilidad de los vehículos.

A pesar de que la distribución pudiera tratarse como un problema distinto a la creación de la demanda, la misma es parte importante de un procedimiento trascendental dentro de la cadena de suministro: la mercadotecnia, ya que a través de la primera se hace llegar el producto o servicio a los clientes potenciales identificados previamente por la segunda (Basheer, 2017).

La mercadotecnia destaca como uno de los procesos que agregan valor al cliente final mediante el posicionamiento del producto, y con respecto a su finalidad busca identificar, cuantificar y anticipar las necesidades del mercado para satisfacer tales requerimientos (Marr, 1984). El plan estratégico de mercadotecnia incluye el posicionamiento, la publicidad y la promoción de un producto, actividades que generan valor a largo plazo (Russell *et al.*, 2005) y permite el posicionamiento del producto o servicio.

2.3 Diseño de ruta y métodos de planteamiento

Indudablemente una de las primeras acciones que se consideran al momento de diseñar una ruta es tomar un mapa y ubicar los puntos de interés para dimensionar el problema y posiblemente, se considere la lógica empírica de quien o quienes estén interesados en trazar un recorrido bajo ciertos criterios de decisión, como lo pueden

ser el tiempo, el costo financiero y/o la distancia a recorrer por el o los vehículos y demás variables contextuales a la implementación, como por ejemplo: la naturaleza de lo que se transporta (de ser el caso), la variabilidad (y el tipo) de la demanda, la cantidad (y características) de los vehículos con los que se dispone, divisiones administrativas y políticas, topografía, infraestructura, por mencionar algunas.

Además del método empírico-analítico existe una perspectiva cuantificable para la solución de distintos tipos de problemas: la modelación. Se analizarán estas opciones a continuación.

El método de solución empírico se basa en que el responsable de diseñar la(s) ruta(s), tiene el conocimiento suficiente para lograr llegar a un resultado eficiente y que satisfaga los requerimientos del problema, sin embargo, no existe ninguna garantía de que este logre el mejor resultado posible o, en su defecto, uno de buena calidad, en cuanto se traslade a cifras calculables respecto al desempeño en términos de costo, es por esto que se optará por la inclusión de la experiencia empírica para la solución del problema del caso de estudio y diseño de la herramienta dentro de una metodología sustentada por antecedentes en su investigación previa buscando una perspectiva mensurable.

En el presente documento se detalla el caso de estudio de una empresa de mercadotecnia (véase capítulo 6) que requiere hacer llegar unidades promocionales a ciertos puntos de interés, es decir, lo que se busca es la secuencia a seguir para cubrir todos y cada uno de los puntos, por los cinco vehículos promocionales disponibles, las alternativas al uso de la experiencia empírica para el ruteo de vehículos pueden ser: el uso de software especializado o el uso de un modelo[1].

El uso de modelación abre una serie de categorías de las que se puede inquirir una herramienta que mejor se adecue al problema, en la Tabla 2.1 se muestran algunos ejemplos de modelación. Se busca generar una respuesta que ofrezca una solución con antelación al problema, es por tal motivo que se explorará la modelación

[1]Entendiendo por modelo, una representación matemática de la realidad.

matemática prescriptiva.

Tabla 2.1: Categorías y características de algunas técnicas de modelado

Categoría	Características del modelo		Técnica de gestión
	Forma de la función dependiente	Valores de las variables independientes	
Descriptivo «¿Qué pasó?»	Conocida, bien definida	Desconocidos o inciertos	Simulación, técnicas de revisión y evaluación de proyecto, teoría de colas, modelos de inventario
Predictivo «¿Qué podría pasar si...?»	Desconocida, no definida	Conocidos o bajo el control del tomador de decisiones	Análisis de regresión, series de tiempo o discriminatorio
Prescriptivo «¿Qué se debería hacer?»	Conocida, bien definida	Conocidos o bajo el control del tomador de decisiones	Programación lineal, programación entera mixta, método de la ruta crítica, lote económico, programación no lineal

Fuente: Ragsdale (2012)

Las variables independientes[2] son conocidas, en otras palabras, se sabe el costo en el que incurre cada vehículo de ir de un punto a otro, y la forma de la función dependiente, de la misma manera, es conocida, es decir, se tiene entendimiento de en qué medida impactará recorrer determinado trayecto, mediante la adición de cada uno de los costos que lo componen. Debido a esto se descaminarán las técnicas de modelado descriptivas y predictivas, quedando únicamente las prescriptivas.

De las mencionadas anteriormente el método de la ruta crítica o CPM (*Critical Path Method*) busca calcular costo en el que incide un proyecto a lo largo de un determinado periodo de tiempo (Vanhoucke, 2013) y no tiene relación con el problema en cuestión, así que será descartado. De modo similar el lote económico o EOQ (*Economic Order Quantity*) se excluirá, puesto que considera la reciprocidad entre el costo de ordenar y de almacenar, siendo una herramienta de modelado *ad hoc* a los problemas de inventario (Schwarz, 2008). De las técnicas mencionadas,

[2]También conocidas como parámetros, que son los datos considerados como imprescindibles para la evaluación de una situación

las restantes se encuentran catalogadas dentro de la programación u optimización matemática, que es una rama de las matemáticas que se encarga de minimizar o maximizar una función objetivo sujeta a restricciones que pueden contener variables del tipo lineal, no lineal, enteras y/o binarias (Dantzig y Thapa, 1997), características que se analizarán a detalle en el apartado 2.4.

Observando los modelos matemáticos de optimización con respecto a las características de sus variables, como lo menciona García Travieso (2014), es posible dividir los modelos en: programación continua (lineal), entera y mixta, esto basándose en el tipo de dato que puede tomar cada incógnita; en los primeros pueden tomar valores reales, en los segundos solamente pueden tomar valores enteros, y en el tercero una mezcla de los dos que le anteceden. Con respecto a los valores que pueden tomar los parámetros del modelo, se identifica la programación determinística (fijos), paramétrica (varían de forma sistemática) y estocástica (aleatorios).

La programación lineal, como su nombre lo indica está formada por funciones lineales tanto en la función objetivo como en sus restricciones, caso contrario para la programación no lineal, que requiere técnicas avanzadas de modelación matemática en la fase de formulación (en comparación con la lineal), y convirtiendo a la primera opción en una técnica ampliamente usada por tal motivo (Luenberger y Ye, 2016). Es así, que la programación no lineal representa una alternativa tortuosa para el objetivo del presente documento, además de que existen herramientas (definidas en la siguiente sección) que no requieren de esta técnica de modelación y por tales motivos se excluirá.

Desdeñando desde las técnicas de modelado presentadas anteriormente, se considerará como derrotero del planteamiento la modelación matemática prescriptiva.

2.4 Modelación matemática

Según Ragsdale (2012), los modelos matemáticos se componen de tres elementos esenciales: las variables de decisión, pueden representar números enteros, continuos u opciones binarias[3] (x); las restricciones, que están en función de las anteriores y que deben ser cumplidas en términos de sus operadores racionales (2.2, 2.3 y 2.4); y el objetivo[4], función dependiente de las variables de decisión y que el tomador de decisiones desea maximizar o minimizar (2.1).

$$\text{máx (o mín): } f_0(x_1, x_2, \ldots, x_n) \tag{2.1}$$

$$\text{sujeto a: } f_1(x_1, x_2, \ldots, x_n) \leq b_1 \tag{2.2}$$

$$f_k(x_1, x_2, \ldots, x_n) \geq b_k \tag{2.3}$$

$$f_m(x_1, x_2, \ldots, x_n) = b_m \tag{2.4}$$

A propósito de representar una forma general simplificada de visualizar los modelos de optimización, Sun y Yuan (2006) sugieren el siguiente arquetipo:

$$\text{máx (o mín): } f(x) \tag{2.5}$$

$$\text{s.a: } x \in X \tag{2.6}$$

Por otra parte, con la intención de discernir entre los métodos de solución para los modelos matemáticos, se definirá como solución **factible**, la que satisface todas las restricciones (Taha, 2012) y por **óptima** que, además de ser factible, produce el mejor valor buscado, ya sea máximo o mínimo. De esta manera, se cuenta con métodos de solución «exactos» (que pueden encontrar el óptimo), como lo son: ramificación y acotación, hiperplanos de corte y ramificación y corte. Además de métodos de solución que se aproximan al óptimo, a cambio de ofrecer un menor

[3] Solo pueden tomar uno de dos valores: 0 o 1.

[4] Es posible buscar más de un objetivo mediante la programación multiobjetivo donde la función se vuelve vectorial.

esfuerzo computacional: los métodos «heurísticos» (García Travieso, 2014).

Conforme a la categorización de los métodos de solución a los modelos matemáticos, es posible identificar en la literatura diversas opciones como se ejemplifica en la Tabla 2.2.

Tabla 2.2: Métodos de solución

Técnicas exactas	Ramificación y corte
	Hiperplanos de corte
	Ramificación y acotamiento
	Programación lineal
Heurística	Procedimiento de ahorros
	Procedimiento de inserción
	Mejora
	Partición-primero ruta-segundo
	Ruta-primero partición-segundo
Metaheurística	Enjambre de partículas o PS (*Particle Swarm*)
	Colonia de hormigas, *ant systems* o ACO (*Ant Colony Optimization*)
	Búsqueda tabú o TS (*Tabu Search*)
	Recocido simulado o SA (*Simulated Annealing*)
	Algoritmos genéticos o GA (*Genetic Algorithms*)
	Algoritmos de optimización basados en hierba invasiva o IWO (*Invasive Weed Optimization*)
	Búsqueda cuckoo o CS (*Cuckoo Search*)
Interactivo	Aproximación basada en la preferencia
	Aproximación intuitiva
	Simulación

Fuente: Elaboración propia con información de Anbuudayasankar *et al.* (2014)

Cuando se pretende abordar un problema atinente a uno o varios itinerarios de viaje, es imprescindible detallar la ruta a seguir por las unidades que transitarán el recorrido. Una perspectiva objetivamente mensurable hacia este tipo de problemas es la del flujo de redes, que como lo indican Ahuja *et al.* (1993) incluye aportaciones por parte de distintas disciplinas, lo que favorece al análisis desde diversos puntos de vista, pero discerniendo en notaciones. Debido a esto se tomarán como **nodos** los lugares a visitar por los vehículos (agentes), y como **arcos** los caminos que pueden seguir estos últimos para llegar de un nodo a otro.

Los modelos empleados comúnmente para la solución de problemas de ruteo (Gutin y Punnen, 2007) son: el problema del agente viajero o TSP (*Traveling Salesman Problem*) y el problema de ruteo de vehículos o VRP (*Vehicle Routing Problem*). Estos planteamientos pertenecen a una rama de la optimización matemática que se denomina optimización combinatoria, en la que existen un número finito de posibles soluciones (García Travieso, 2014).

2.4.1 El problema del agente viajero

El TSP es un modelo de optimización combinatoria, en el que un agente desea visitar cada una de las ciudades de su territorio, de manera en que cubra la totalidad de las mismas, visitando cada ciudad sólo una vez, regresando a su punto de origen (Figura 2.3), minimizando el costo[5] total del viaje (Schlapfer, 1997).

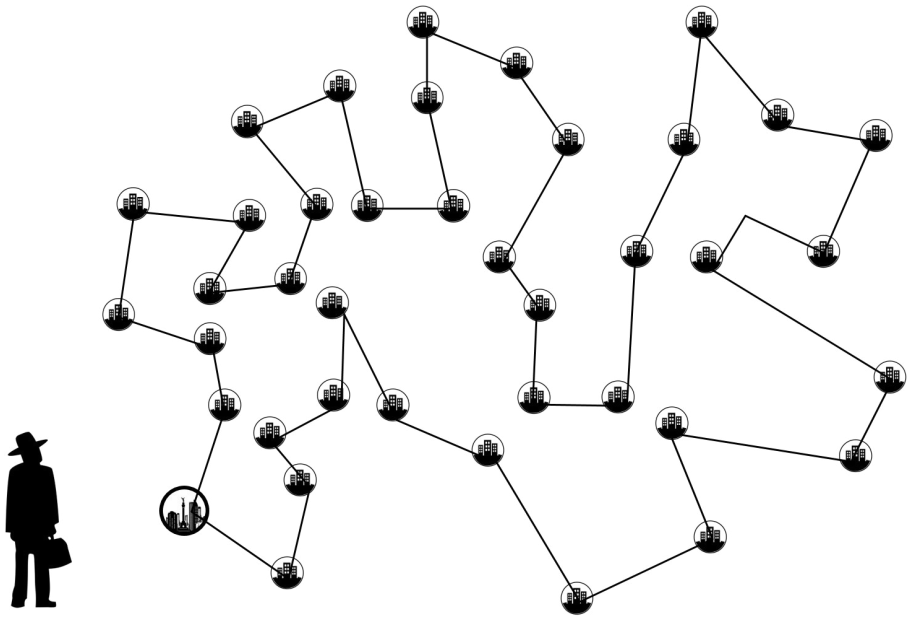


Figura 2.3: Representación de un TSP
Fuente: Elaboración propia basado en Cook (2012)

Con respecto al modelo mencionado anteriormente, Cirasella *et al.* (2001) indican que el objetivo del mismo es encontrar la permutación del orden en el que se

[5]Puede referirse a un costo financiero, al tiempo de recorrido, a la distancia entre los nodos, etc.

visitarán todas las ciudades minimizando el costo.

Lawler *et al.* (1985) consideran el trabajo de B.F.Voigt (1832) como uno de los pioneros de este tópico, en el que entre otras cosas, menciona que el aspecto más importante es cubrir tantas ubicaciones como sea posible sin visitar una ubicación más de una vez.

Para dar inicio al planteamiento del modelo matemático del TSP, es necesario ceñirse hacia la teoría de grafos: definiendo un grafo G como la pareja ordenada ($\mathcal{N}$,$\mathcal{A}$), entonces $G = (\mathcal{N}, \mathcal{A})$, donde $\mathcal{N}$ se compone de subconjuntos de dos elementos de $\mathcal{A}$. Los elementos que componen $\mathcal{N}$ serán los nodos (o puntos) sobre el grafo y los elementos de $\mathcal{A}$ son sus arcos (o líneas) que conectan algunos (o todos los) nodos (Diestel, 2005), siendo $\mathcal{N} = \{1, 2, 3, \ldots, n\}$. En la Figura 2.4 se muestra un ejemplo de un grafo donde $\mathcal{N} =$ { 1, . . . , 7 } y el conjunto de arcos $\mathcal{A} =$ {{ 1, 2 }, { 1, 5 }, { 2, 5 }, { 3, 4 }, { 5, 7 }}.

En este sentido, un **circuito** (o ciclo) se compone de un conjunto de nodos, dentro de un grafo, de forma que sea posible conectar los nodos a través de arcos adyacentes regresando al nodo de partida, sin recorrer el mismo arco dos veces. Si un circuito contiene todos los nodos de un grafo se denomina ciclo hamiltoniano[6] (Lawler *et al.*, 1985). De ahí que el TSP (de un grafo con longitudes de arcos definidas) consista en encontrar el ciclo hamiltoniano con la menor longitud total.

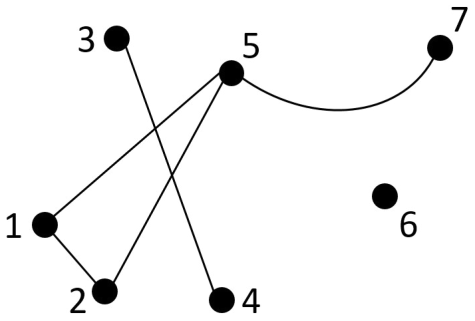


Figura 2.4: Ejemplo de un grafo.
Fuente: Diestel (2005)

Sean I el conjunto de nodos a visitar por el agente, $I = \{1, 2, \ldots, n\}$, c_{ij} el costo de ir desde el nodo i hasta el nodo j, x_{ij} una variable binaria con valor de 1,

[6]Nombrado por el matemático irlandés, William Rowan Hamilton.

si el agente viaja desde el nodo i hasta el nodo j y con valor de 0, de otro modo; y $S \subset I = \{1, 2, \ldots, n\}$, manifestando la cardinalidad como $|\cdot|$.

$$\text{mín} \sum_{i \in I} \sum_{j \in I} c_{ij} x_{ij} \tag{2.7}$$

$$\text{s.a:} \sum_{i \in I} x_{ij} = 1 \qquad \forall j \in I \tag{2.8}$$

$$\sum_{j \in I} x_{ij} = 1 \qquad \forall i \in I \tag{2.9}$$

$$\text{restricción(es) de eliminación de sub-ciclos} \tag{2.10}$$

$$x_{ij} \in \{0, 1\} \qquad \forall i, j \in I, i \neq j \tag{2.11}$$

donde la distancia total del circuito, que se busca minimizar, estará dada por (2.7), las restricciones (2.8) y (2.9) aseguran que el agente salga y llegue una sola vez a cada nodo, respectivamente. La restricción (2.10) (cuya formulación se verá con escrutinio más adelante), también llamada SEC (*Subtour Elimination Constraints*), descarta la posibilidad de que la solución esté compuesta por más de un circuito, hecho que podría suceder si solamente se toman en cuenta las restricciones (2.8) y (2.9), como se muestra en la Figura 2.5, en la cual se satisfacen las restricciones mencionadas con la solución $x_{12} = x_{21} = x_{34} = x_{43} = 1$ para un problema de cuatro nodos (Lawler *et al.*, 1985).

Figura 2.5: Ejemplo de sub-ciclos
Fuente: (Lawler *et al.*, 1985)

La restricción (2.10) en la forma (2.12) genera un número de restricciones del orden de: $2^n - 2$, aunque es conveniente precisar que es posible usar expresiones alternas que busquen minimizar el número de restricciones provenientes de la primera,

logrando hacer el método de solución más eficiente.

$$\sum_{i \in S} \sum_{j \in S} x_{ij} \leq |S| - 1 \qquad \forall S \subset I \tag{2.12}$$

La formulación mencionada previamente se basa en el trabajo de Dantzig y Fulkerson (1954) y representa una forma «simple» de eliminar sub-ciclos, pero con alto costo de ejecución en recursos computacionales. Aguayo *et al.* (2017) identifican algunas aportaciones en el desarrollo de SEC, siendo un planteamiento destacable el de (Miller *et al.*, 1960) quienes las representan de la siguiente manera:

$$u_i - u_j + (|\mathcal{N}| - 1)x_{ij} \leq |\mathcal{N}| - 2 \qquad \forall i, j \in \mathcal{N} - \{1\}, i \neq j \tag{2.13}$$

$$1 \leq u_i \leq |\mathcal{N}| - 1 \qquad \forall i \in |\mathcal{N}| - \{1\} \tag{2.14}$$

donde la variable u determina con un número real el orden en el que cada ciudad es visitada dentro del circuito. La restricción de estado (2.11) termina el modelo.

Existen trabajos que se basan en el modelo del TSP para la solución exitosa del ruteo de vehículos, como el expuesto por Raffo y Ruiz (2005) que, en Lima, Perú, desarrollaron un modelo de optimización de una determinada ruta de entrega diaria, optimizando la ruta de cinco vehículos que abastecían 80 cajas cada uno a un costo de 300 soles peruanos por día, bajo la aplicación del ruteo con base en la experiencia de los operadores, para dar paso al análisis de la aplicación de una ruta basada en un TSP que reportó un costo de 175 soles por día, generando un ahorro para la empresa relativo del 58 %.

Del mismo modo hay diferentes variaciones del TSP, las cuales se diferencian por el tipo de restricciones atinentes al problema que se busque resolver, verbigracia, el problema del agente viajero blanco y negro o BWTSP (*Black and White Travelling Salesman Problem*) considera el conjunto de nodos $\mathcal{N}$ dividido en dos conjuntos de la manera $\mathcal{N} = B \cup W$ donde B es el conjunto de nodos «negros» y W el conjunto de

nodos «blancos» y se busca encontrar el circuito con la mínima distancia y que en su solución contenga dentro de los segmentos de dos nodos negros consecutivos hasta R número de nodos blancos y una distancia máxima de L (Gouveia *et al.*, 2017).

Este modelo tiene diversas aplicaciones dependiendo del contexto de la disciplina que lo implemente, por ejemplo, la reducción de movimientos de una grúa Huang y Wong (2018). Derivado de este planteamiento, es conveniente mencionar el problema del agente viajero múltiple o m-TSP (*Multiple Traveling Salesman Problem*), en el que se incluye más de un agente.

2.4.2 Complejidad de solución

El planteamiento del TSP es claro en cuanto al objetivo, pero el espacio de soluciones posibles presenta un serio problema en lo que a la búsqueda del óptimo respecta.

Por ejemplo, podemos partir de un TSP de cuatro nodos, en el que se irá construyendo una posible solución como se ilustra en la Figura 2.6, al comienzo (Figura 2.6a) se tendrán tres opciones para ir desde el primer nodo hacia el segundo, ahora bien, desde el segundo nodo (Figura 2.6b) existen dos opciones para llegar al tercero, restando una sola opción para el penúltimo movimiento y posteriormente cerrar el circuito (Figura 2.6c); en otras palabras, para las primeras tres opciones (del problema anteriormente mencionado) hay dos opciones por cada una, y para las dos opciones mencionadas hay una sola opción restante, y así completar cada circuito posible.

En resumen, el espacio de soluciones obedece la proporción de: $(n-1)\cdot(n-2)\cdot(n-3)\cdot\ldots\cdot 3\cdot 2\cdot 1$ (Cook, 2012). Simplificando, surge la expresión $(n-1)!$, que representa el posible rango de soluciones para un problema en el que el costo de ir de un nodo a otro es diferente bidireccionalmente. Si el costo para llegar de un nodo a otro es el mismo en ambas direcciones entonces se trata de un TSP simétrico

($c_{ij} = c_{ji}$), para tal caso el espacio de soluciones estará dado por la expresión (2.15) (Lin, 1965).

$$\frac{(n-1)!}{2} \tag{2.15}$$

Con el fin de lograr que al lector le sea posible escrutar el escalamiento del espacio de soluciones, se muestra la Figura 2.7 en la que se detalla el número de posibles rutas para un TSP simétrico, dado n número de nodos.

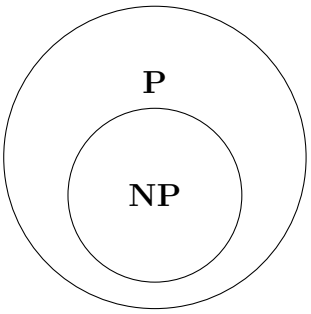
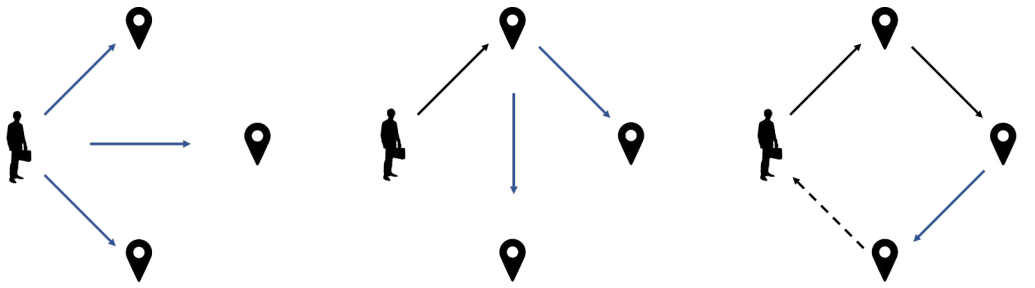


Figura 2.8: Una perspectiva tentativa del conjunto NP

Fuente: Garey y Johnson (1979)

Un vistazo sucinto hacia el análisis de complejidad lleva la presente investigación hacia el trabajo que desarrollaron Díaz *et al.* (1996), en el que identifican distintos conjuntos de problemas según sus características para llegar a una solución óptima, entre los que destacan los pertenecientes a **P** y a **NP** (Figura 2.8). Para los problemas que pertenecen a **P** es posible llegar a la solución óptima de manera eficiente, por otro lado para los del tipo **NP** no hay un algoritmo conocido en el que la solución esté dada por un tiempo polinomial, es decir, para llegar a la solución óptima, el tiempo aumenta exponencialmente conforme lo hacen las dimensiones del problema (en este caso el número de nodos). Garey y Johnson



(a) Primer movimiento (b) Segundo movimiento (c) Tercer movimiento

Figura 2.6: Posible solución de un TSP de cuatro nodos
Fuente: Elaboración propia con información de Cook (2012)

n	$\frac{(n-1)!}{2}$
4	3
5	12
6	60
7	360
8	2,520
9	20,160
10	181,440
11	1,814,400
12	19,958,400
13	239,500,800
14	3,113,510,400
15	43,589,145,600
16	653,837,184,000
17	10,461,394,944,000
18	177,843,714,048,000
19	3,201,186,852,864,000
20	60,822,550,204,416,000

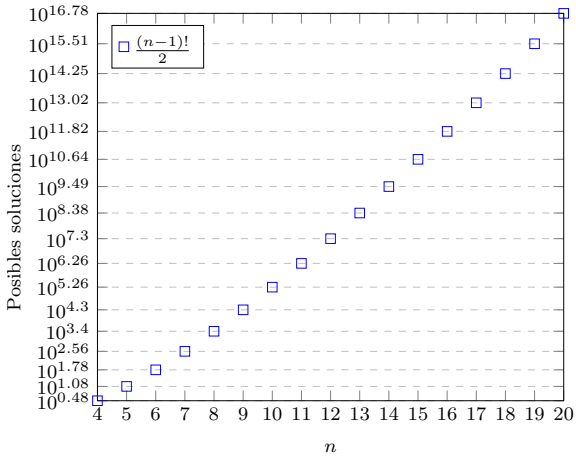


Figura 2.7: Número de posibles soluciones para el TSP simétrico
Fuente: Elaboración propia con información de Cook (2012) y Lin (1965)

(1979) comprueban la pertenencia del TSP al conjunto **NP**.

En otras palabras, para el TSP no existe un algoritmo conocido que pueda llegar al mejor resultado posible (el óptimo) en un tiempo de cómputo realista para las aplicaciones sensibles al anterior, en el que el tiempo mencionado no rebase el límite del horizonte de planeación programado (Figura 2.9).

Aunque existen aproximaciones que buscan nivelar, en términos de tiempo, el trabajo entre el esfuerzo computacional y la implementación del TSP, que depende estrechamente de la capacidad tecnológica actual (y que varía constantemente), como lo es el caso del CIP (*Computation-Implementation Parallelized*), que tiene el supuesto de poder realizar el cálculo por fases mientras se está efectuando la implementación (Çavdar y Sokol, 2015), en el presente trabajo solamente se buscará reducir lo más posible el tiempo de respuesta para el ruteo, de tal manera en que la herramienta pueda ser reproducible para proyectos sensibles al tiempo.

2.4.3 El problema del agente viajero múltiple

Para el caso con más de un agente viajero, y suponiendo que todos los agentes deben salir y volver al mismo nodo, sin que ningún nodo (excepto el inicial) sea visitado más de una vez por indeterminado agente, el problema representa una noción diferente al TSP, como se muestra en la Figura 2.10.

En un análisis realizado por Bektas (2006), sobre la formulación del m-TSP y algunas variaciones originadas debido a restricciones peculiares inherentes a distintos tipos de aplicaciones, el autor puntualiza modelos modificados del mismo para problemas específicos, no obstante, en el presente trabajo se detallará exclusivamente el planteamiento general, señalado por el primero.

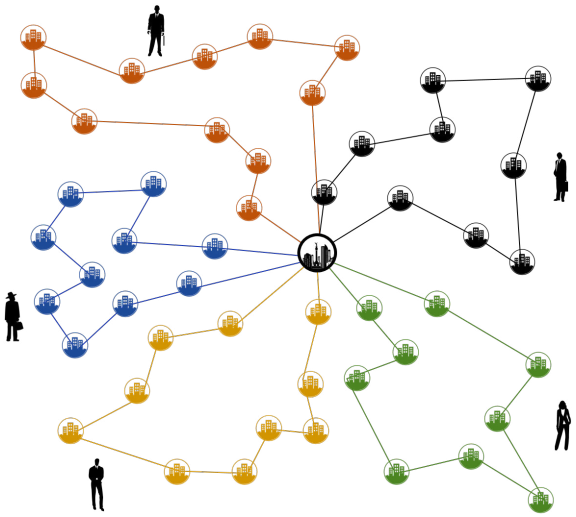


Figura 2.10: Esbozo de un m-TSP
Fuente: Elaboración propia con información de Bektas (2006)

Retomando el modelo del TSP expuesto en el apartado 2.4.1, es posible conservar el anterior, sustitu-

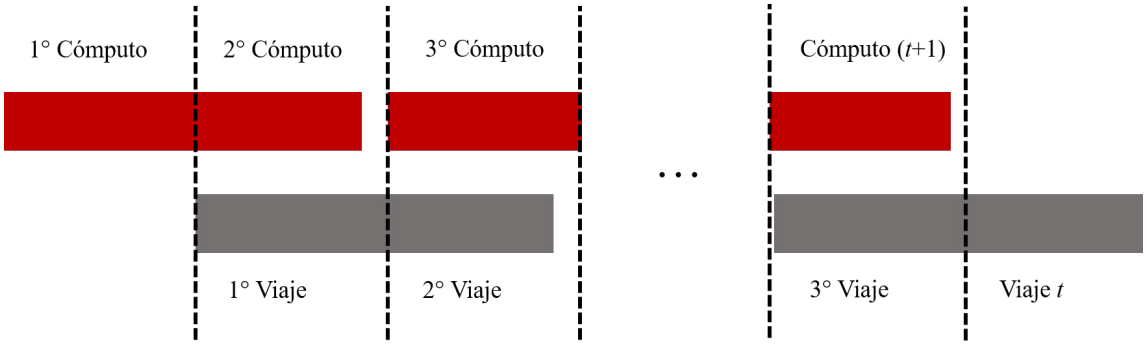


Figura 2.9: Paralelización del tiempo de cómputo y viaje
Fuente: Çavdar y Sokol (2015)

yendo las SEC por la desigualdad (2.16), y agregando además, las ecuaciones (2.17) y (2.18) a las restricciones como lo indica Bektas (2006), para modelar el m-TSP. Siendo m el número de agentes que transitarán los nodos.

$$u_i - u_j + px_{ij} \leq p - 1 \qquad \forall 2 \leq i \neq j \leq n \tag{2.16}$$

$$\sum_{j=2}^{n} x_{1j} = m \tag{2.17}$$

$$\sum_{j=2}^{n} x_{j1} = m \tag{2.18}$$

donde (2.17) y (2.18) se cercioran de que todos los agentes salgan y regresen al nodo de inicio, respectivamente, quedando las SEC a cargo de la restricción (2.16), preservando la función objetivo (2.7), la restricción de estado (2.11), así como las restricciones (2.8) y (2.9), esclarecidas en la subsección 2.4.1.

El objetivo usual del m-TSP es minimizar la suma de las distancias recorridas por todos los agentes, no obstante minimizar la distancia del recorrido de mayor longitud, también es un objetivo apropiado para su aplicación en problemas reales, puesto que genera soluciones con un mejor balance de carga de trabajo[7] asignada para cada agente (Soylu, 2017), aunque, esto no asegura que el número de nodos se equilibre en la solución final, y es ineludible incluir una restricción adicional (o modificar alguna existente) si se pretende lograr esto último.

Continuando con lo que respecta al trabajo de Soylu (2017), el autor aborda el m-TSP tomando en cuenta dos funciones objetivo distintas; minimizar la distancia del circuito más largo y minimizar la distancia total de todos los circuitos. Concluyendo que, con el primer objetivo, el algoritmo propuesto por el autor solamente puede resolver instancias muy pequeñas, por un lado, mientras que con el segundo objetivo, mostró ser útil para instancias de tamaño medio, llegando a soluciones cercanas al mejor resultado conocido.

[7] En términos de distribución del costo.

Posteriores modificaciones e innovaciones de este modelo resultaron en el planteamiento del VRP; mismo que incluye variables y parámetros en su modelación según la complejidad del problema (Anbuudayasankar *et al.*, 2014), como se muestra en la Tabla 2.3; cada singularidad intrínseca del problema a tratar, puede verse reflejada como una restricción durante la fase de modelación. En la siguiente subsección se realizará un estudio ingrávido.

Tabla 2.3: Dimensiones de los problemas de ruteo

Restricciones relacionadas a la flota	Variable
	Constante
	Capacidad
	Número
Condiciones estáticas	Multi-depósito
	Un depósito
Características del problema	Distancia simétrica/asimétrica
	Carga y tiempo estocásticos
	Carga y tiempo determinísticos
Restricciones operativas	Personal (cantidad y trabajo)
	Tiempo (máximo y/o ventanas específicas)
	Longitud del recorrido

Fuente: Anbuudayasankar *et al.* (2014)

2.4.4 El problema de ruteo de vehículos

Antes de agrupar las distintas dimensiones (mencionadas previamente) en un conjunto de problemas pertenecientes al VRP, se comenzará por definir una variante «elemental»; el problema de ruteo de vehículos capacitado o CVRP (*Capacitated Vehicle Routing Problem*): en este problema se debe cubrir una cantidad precisa de indistinto producto o servicio, desde un único depósito, mediante una flota conocida (en cantidad y cualidad) de vehículos con las mismas características y capacidades de carga, abasteciendo a un conjunto de clientes (nodos), retornando ulteriormente al depósito (Toth y Vigo, 2014). Un bosquejo tentativo se exhibe en la Figura 2.11.

El modelo del CVRP entonces, incluye la restricción de capacidad (denotando

tal parámetro con Q) de la siguiente manera:

$$\text{mín} \sum_{i \in I} \sum_{j \in I} c_{ij} x_{ij} \tag{2.19}$$

$$\text{s.a:} \sum_{i \in I} x_{ij} = 1 \qquad \forall j \in I \tag{2.20}$$

$$\sum_{j \in I} x_{ij} = 1 \qquad \forall i \in I \tag{2.21}$$

$$\sum_{j \in I} x_{1j} = |K| \tag{2.22}$$

$$u_i - u_j + Q x_{ij} \leq Q - q_j \qquad \forall i, j \in I \tag{2.23}$$

$$q_i \leq u_i \leq Q \qquad \forall i \in I \tag{2.24}$$

$$x_{ij} \in \{0, 1\} \qquad \forall i, j \in I i \neq j \tag{2.25}$$

Es posible notar la similitud con el m-TSP que, de hecho, es considerado como una relajación del CVRP en la que se elimina la restricción (2.24) de capacidad (Bektas, 2006).

Una definición amplificada del VRP es, como lo sugieren Toth y Vigo (2014), que dado un grupo de requerimientos de transporte y una flota de vehículos, se debe de definir de manera concisa un conjunto de rutas vehiculares que serán ejecutadas en su totalidad (o parcialmente), resultando en la combinación y secuencia de vehículos que logre satisfacer eficientemente dichos requerimientos. Los autores describen una serie de modelos para los siguientes casos agrupados en «familias», siendo sus principales criterios de agrupación por:

1. Estructura de la red (terrestre).
2. El tipo de requerimientos de transporte.
3. Las restricciones que inciden en cada ruta individualmente.
4. La composición de la flota y su ubicación.
5. Las restricciones entre rutas.
6. Los objetivos de optimización.

Es claro que el CVRP no es el único problema dentro de la «familia» de problemas procedentes del VRP, aun así, los pormenores de los modelos de VRP singulares exceden en dimensiones de modelación al caso de estudio que pretende solucionar el presente trabajo, y que por el grado de incertidumbre en los cambios efectuados durante la planeación, muestra como mejor opción el modelo del m-TSP.

Las soluciones arrojadas por el m-TSP (y por el CVRP) incluyen, la asignación de todos los nodos a visitar entre los agentes (vehículos), con el fin de generar rutas para cada uno de los mismos, pero este trabajo puede ser realizado en dos partes, generando en una primera etapa la asignación de los nodos, siguiendo con la solución del ruteo a seguir para cada vehículo (Ayu *et al.*, 2015; Dhanachandra *et al.*, 2015; Xu *et al.*, 2017).

2.4.5 Partición de nodos

A continuación se recalcarán aportaciones para la solución del m-TSP, generando la partición de los nodos en primer lugar, empleando métodos que se encarguen de dividir un conjunto de datos en una cantidad determinada de grupos, procesos también conocidos como «clasificación no supervisada» (Gimenez, 2010), *clustering* (Dhanachandra *et al.*, 2015) o *cluster analysis* (Wu, 2012).

El trabajo de Xu *et al.* (2017), emplea el método de k-medias para realizar una asignación a través de un algoritmo aproximado para resolver cada asignación como un TSP «aislado». De la misma manera es remarcable el trabajo de Yu *et al.* (2012) quienes utilizan el resultado de la asignación generada por el método de k-medias como solución inicial para explorar las demás, y así lograr el ruteo de los distintos vehículos.

2.4.6 El método de k-medias

Para comenzar, k-medias (o *k-means*) es un método de agrupamiento que se encuentra en el campo del análisis de grupos (*cluster analysis*), y que busca, de manera iterativa, realizar particiones, no traslapadas, de un conjunto de datos de tipo cuantitativo, representadas por un centroide: que se define como la media entre las desigualdades (costos, para el caso del m-TSP), para cada subconjunto. Para llegar a la mejor solución mediante este método la complejidad entra en el conjunto **NP** (Wu, 2012).

Para comenzar con la partición, el método de k-medias requiere definir la desemejanza entre los datos ubicándolos en un plano bidimensional en el que sea posible calcular la distancia en función de (2.26) (Gimenez, 2010). Acto seguido, el algoritmo coloca k número de centroides y asigna los nodos al centroide más cercano.

$$d(x_i, x_{i'}) = \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2 = \|x_i - x_{i'}\|^2 \tag{2.26}$$

$$W(C) = \sum_{k=1}^{K} \sum_{C(i)=k} \sum_{C(i')=k} \|x_i - x_{i'}\|^2 = \sum_{k=1}^{K} N_k \sum_{C(i)=k} \|x_i - \bar{x}_k\|^2 \tag{2.27}$$

Puesto que el método de k-medias representa una alternativa robusta y relativamente simple para la partición de conjuntos sigue vigente, como lo manifiesta Wu (2012), sin embargo, presenta algunas complicaciones para su aplicación a ciertos tipos de datos.

2.4.7 El problema de las p-medianas

El algoritmo de partición de conjuntos, k-medias comparte similitudes con los problemas de ubicación de instalaciones, en este trabajo se mencionará el problema de las p-medianas o PMP (*P-Median Problem*) que consiste en establecer una can-

tidad p de instalaciones, eligiendo su ubicación dentro de un conjunto de n nodos, minimizando[8] la distancia total entre las instalaciones y los nodos (también referidos como clientes) asignados a las mismas (Yakici y Yiğit, 2017).

El PMP se encuentra dentro de la clasificación de problemas clásicos como lo son el de los p-centros y el de cobertura (Daskin y Maass, 2015). Este problema, como lo menciona Segura *et al.* (2015), está dentro del conjunto **NP** y su complejidad de solución está dada por la expresión (2.28).

$$\binom{n}{p} = \frac{n!}{p!(n-p)!} \tag{2.28}$$

Con respecto a la modelación del PMP, habrá que definir: un conjunto N que contendrá los nodos (clientes o puntos de demanda), $I = \{1, 2, 3, \ldots, n\}$, y de la misma manera, un conjunto J de medianas (instalaciones), $J = \{1, 2, 3, \ldots, m\}$, así como una matriz de costos c_{ij}, para satisfacer un servicio hacia el nodo i desde la mediana j, para cada $i \in N$ y $j \in L$; y por otra parte, considerando las variables binarias de decisión

$$x_{ij} = \begin{cases} 1 & \text{si el nodo } i \text{ es atendido por la instalación } j \text{ y} \\ 0, & \text{de otro modo} \end{cases}$$

$$y_j = \begin{cases} 1 & \text{si la instalación } j \text{ se habilita y} \\ 0, & \text{de otro modo} \end{cases}$$

se desarrollará el modelo, de acuerdo al observado en el trabajo de Mladenović *et al.* (2007), de la siguiente manera:

$$\text{mín} \sum_{i \in I} \sum_{j \in J} c_{ij} x_{ij} \tag{2.29}$$

[8]Para el caso de maximización se trata de un PMP desagradable u OPMP (*Obnoxious P-Median Problem*) (Herrán *et al.*, 2018).

$$\text{s.a:} \sum_{j \in J} x_{ij} = 1 \qquad \forall i \in I \tag{2.30}$$

$$x_{ij} \leq y_j \qquad \forall i \in I, j \in J \tag{2.31}$$

$$\sum_{j \in J} y_j = p \tag{2.32}$$

$$x_{ij}, y_j \in \{0, 1\} \qquad \forall i \in I, j \in J \tag{2.33}$$

donde (2.29) representa la función objetivo que minimiza la distancia total desde cada nodo hacia su correspondiente mediana asignada, la restricción (2.30) se encarga de asignar cada nodo solamente una vez a indeterminada mediana, (2.31) asegura que los nodos sean asignados solamente a las medianas aperturadas, para seguir con la restricción (2.32) que señala el número exacto de medianas a ubicar, para finalizar con la restricción de estado (2.33).

Imperativamente se debe considerar el aprendizaje empírico para problemas tan complejos como lo es rutear vehículos, sin embargo, para lograr un resultado cuantificable y que garantice que el método de solución empleado busque aprovechar al máximo las ventajas de los métodos exactos y las capacidades de los algoritmos de aproximación, que han mostrado resultados eficientes, se empleará la programación lineal entera mixta en el planteamiento, siendo el problema que mejor se adapta a la situación: el m-TSP, por una parte, y con la intención de que el mecanismo de solución del ruteo para más de un vehículo cuente con la posibilidad de incluir en la decisión del reparto de nodos, a cada uno de los mencionados, la experiencia empírica del tomador de decisiones, se utilizará una aproximación de partición primero, ruta segundo, empleando el PMP para realizar dicha asignación, seguido del TSP para hacer frente al ruteo. Un ejemplo de esto es segmentar los puntos de interés por ciudad para la partición, para después computar cada circuito considerando todos los nodos.

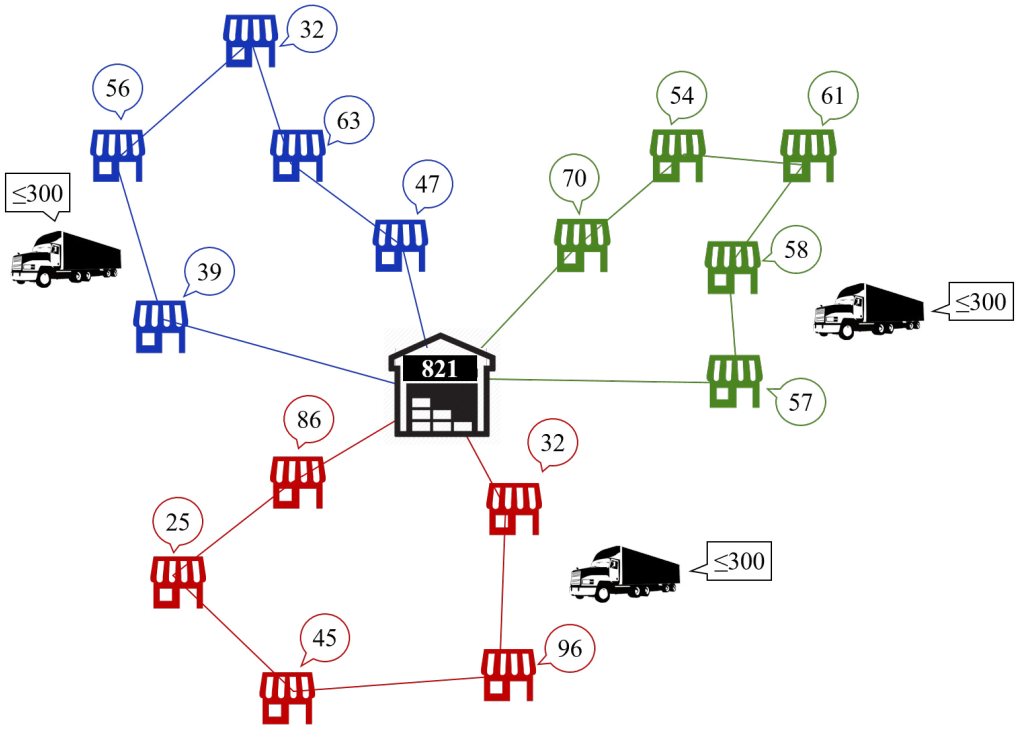


Figura 2.11: Ejemplo gráfico de un CVRP
Fuente: Elaboración propia con información de Toth y Vigo (2014)

Capítulo 3

Métodos de solución

Habiendo ya definido un método de dos fases para resolver el problema de ruteo, a través del PMP y el TSP, es menester de la presente investigación encontrar un algoritmo para llegar a la mejor solución posible a los problemas referidos, considerando la coyuntura de aplicación del problema raíz. Así que, en este capítulo se analizarán los algoritmos de solución exactos, que garantizan llegar al resultado óptimo (bajo ciertas consideraciones) y los algoritmos de aproximación, que no garantizan un resultado óptimo o factible, pero si una solución aceptable con la que se escudriñe un acercamiento al óptimo. Se comenzará con los algoritmos exactos. Para seguir con los algoritmos de aproximación, en razón de su grado de complejidad: heurísticos y metaheurísticos.

3.1 Algoritmos exactos

Para la solución de los modelos de programación lineal entera mixta, se puede hacer uso de algoritmos que resuelven los problemas agotando la gran mayoría de las soluciones posibles de una problema de diferentes maneras para llegar al mejor resultado.

3.1.1 Ramificación y acotación

El algoritmo de ramificación y poda o B&B (*Branch and Bound*) busca en todo el espacio de soluciones, evadiendo la enumeración categórica sirviéndose de límites, que dividen el espacio mencionado en subconjuntos de soluciones, comparando el límite menor obtenido (para el caso de la minimización) con las soluciones posibles que se obtendrán de cada subconjunto, mondando los que comprueben no contener la solución óptima, y así continuar fragmentando el espacio de soluciones en áreas (del espacio de soluciones) cada vez más pequeñas y con mejores resultados, que eventualmente conducirán el óptimo (Clausen, 1999).

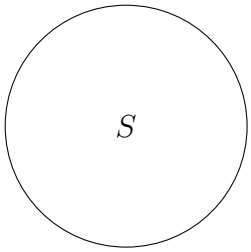
Un esquema, a modo de ejemplo, del funcionamiento de este algoritmo se muestra en la Figura 3.1; ilustrando la inicialización en 3.1a y 3.1b, la primera iteración en 3.1c y 3.1d, finalizando el ejemplo en la segunda iteración (3.1e y 3.1f).

El uso de este algoritmo es común para el TSP (Huang y Wong, 2018; Nilofer y Rizwanullah, 2017) y puede ser utilizado para resolver instancias de tamaños considerables (de 300 a 3000 nodos), siempre y cuando se planifique un sistema de solución que combine otras técnicas de solución, y la ejecución en paralelo de los cálculos (Pekny y Miller, 1992).

3.1.2 Ramificación y corte

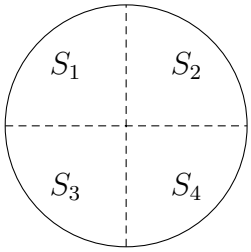
Este algoritmo, también conocido como B&C (*Branch and Cut*), para la solución de problemas de programación lineal, y particularmente de optimización combinatoria, utiliza relajaciones del problema original, por ejemplo, modificando las restricciones de variables enteras a continuas, generando planos de corte que delimitan el espacio de soluciones, con el objetivo de descartar las soluciones que no contengan el óptimo (Padberg y Rinaldi, 1991).

El uso de este algoritmo combinado con procedimientos de búsqueda del óptimo

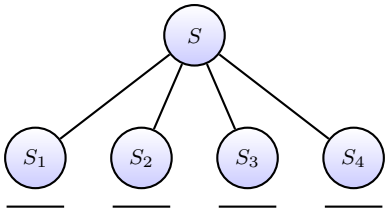


(a) Inicialización

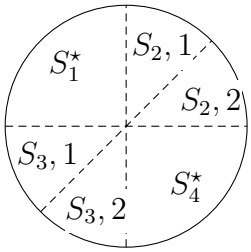
(b) Espacio completo de soluciones



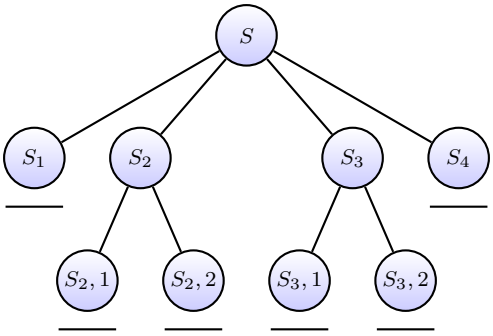
(c) Primera iteración



(d) Primer ramificación



(e) Segunda iteración



(f) Segunda ramificación

⋆No contienen la solución optima.

Figura 3.1: Ejemplo gráfico de B&B
Fuente: Clausen (1999)

es común para el TSP, como se presenta en el trabajo de Padberg y Rinaldi (1987) quienes emplean un algoritmo como se muestra en la Figura 3.2, o el concerniente a Hernández Pérez y Salazar González (2004), quienes resuelven sus respectivas experimentaciones llegando al óptimo.

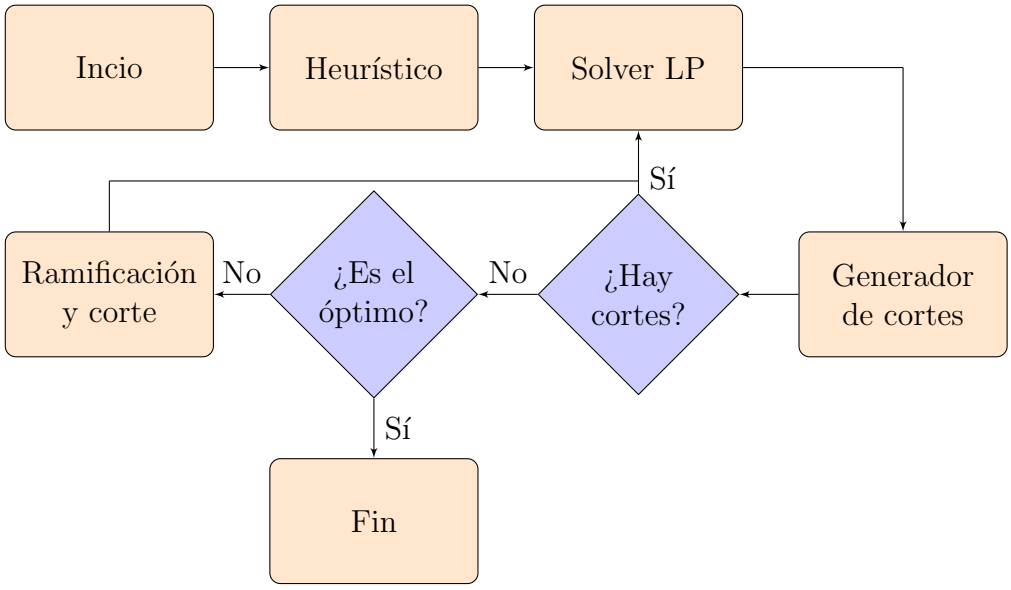


Figura 3.2: Algoritmo que incluye ramificación y corte
Fuente: Padberg y Rinaldi (1987)

Para el caso del PMP este método de solución resulta conveniente para el caso de estudio debido a que las posibles soluciones del mismo no «escalan» en la misma propoción que el TSP, atenuando las mismas en tanto se aumente el número de medianas (subgrupos), como se observa en la Figura 3.3.

3.2 Algoritmos heurísticos

En esta sección se abordarán algunos métodos heurísticos, que deben su presencia, en gran parte, a la complejidad computacional para la solución de los problemas catalogados dentro del conjunto **NP**. Se describirán los principales algoritmos, para pasar a los algoritmos metaheurísticos que son de mayor complejidad, pero diseñados

para poder ser aplicados a distintos tipos de problemas.

Derivado del vocablo griego *heuriskein* que significa «encontrar», el concepto de los algoritmos heurísticos, como lo indican Díaz *et al.* (1996), se basa en procedimientos simples, mismos que pueden estar basados en el sentido común y que, en teoría, deben ofrecer una solución (que no siempre es la óptima o inclusive factible) aceptable a problemas difíciles de una manera rápida. De la misma manera, el autor antes mencionado indica la flexibilidad de estos métodos para manipular las variables del planteamiento.

Reinelt (1994) propone una clasificación para los heurísticos (dedicados a la solución del TSP), para un grupo de los mismos, que trata procedimientos relativamente simples, dividiéndolos en heurísticos **de contrucción** y **de mejora**, a continuación se describirán algunos de ellos brevemente.

3.2.1 Heurísticos de construcción

Como su nombre lo índica, estos algoritmos se basan en un regla de construcción, conservando las partes exitosamente construidas de la ruta constantes durante el funcionamiento del algoritmo, hasta que se finaliza el ciclo hamiltoniano (Reinelt, 1994). Con el objetivo de favorecer la descripción de los algoritmos, se considerará la notación del TSP reseñada en el apartado 2.4.1, manteniendo los nodos y arcos como la pareja ordenada $(\mathcal{N},\mathcal{A})$ y denotando una solución parcial como $\mathcal{N}(s^p)$.

- **Vecino más cercano:** También conocido como *nearest neighbor*, consiste en iniciar la construcción de la ruta desde un nodo aleatorio, para añadir después del último agregado, el nodo más cercano a este (Hahsler y Hornik, 2007). Ahora bien, Gutin *et al.* (2002) corroboran que no es aconsejable el uso de este algoritmo para la construcción de soluciones del TSP, debido a la mala calidad de la solución obtenida, encontrandose en el mismo caso los algoritmos voraces.

- **Inserción:** Este procedimiento reside en elegir de forma aleatoria un par de nodos i y j, formando un subciclo $i - j - i$, agregando en cada iteración un nodo k, de tal manera en que la inserción (de todas las posibles) que resulte en el menor costo con respecto a las demás, será agregada a la solución (Huang y Yu, 2017), considerando que se debe «romper» indeterminado arco (i, j) siguiendo la pauta de (3.1), calculando la aptitud para la siguiente inserción con la ecuación (3.2), aunque cabe aclarar que podría emplearse un método de inserción arbitrario (Hahsler y Hornik, 2007).

$$k = \arg\min_{k \notin s^p} \{c(i, j, k), \forall (i, j) \in s^p\} \tag{3.1}$$

$$c(i, j, k) = c_{ik} + c_{kj} - c_{ij} \tag{3.2}$$

- **Algoritmos voraces:** Un algoritmo voraz elige el arco con el menor costo en cada iteración hasta concluir el ciclo hamiltoniano (Glover *et al.*, 2001). Como se mencionó anteriormente, Gutin *et al.* (2002) confirman que los algoritmos voraces deben ser eludidos para la construcción de soluciones del TSP, y de hecho, mencionan que en determinada instancia este algoritmo termina con la peor solución posible.

- **Ahorros:** Desarrollado para solucionar el VRP por Huang y Yu (1964), este heurístico también puede ser aplicado al TSP (debido a que puede considerarse como un caso especial con un sólo vehículo). Se basa en elegir un nodo base i_b, produciendo subciclos entre cada nodo restante y el anterior, es entonces que mientras existan subciclos se dispone de la ecuación (3.3), para evaluar si se incluye el arco (i, j) en la solución. El coeficiente más alto, representa el mayor ahorro (Altınel y Öncan, 2005).

$$s_{ij} = c_{i_b i} + c_{j i_b} - c_{ij} \tag{3.3}$$

3.2.2 Heurísticos de mejora

La solución que resulta al hacer uso de un heurístico de construcción para el TSP, cuenta con áreas de oportunidad debido a la baja calidad de la misma, por añadidura, los heurísticos de mejora ocupan una solución inicial y la modifican con cierto tipo de movimiento básico (Reinelt, 1994). Estos algoritmos también son conocidos como de búsqueda local (Helsgaun, 2009).

- **Inserción de nodo:** Cada nodo, se inserta en la mejor ubicación posible, considerando todas las opciones dentro del circuito (Reinelt, 1994), como se muestra en la Figura 3.4.

- **Intercambio 2-opt:** Consiste en remover dos arcos del circuito y reconectar los nodos con arcos diferentes[1], aceptando el resultado sólo si representa un costo menor (Woodruff, 1998).

  Partiendo de la solución inicial, la búsqueda local se efectúa como lo indica el pseudocódigo 1.

  A efectos de ejemplificar el algoritmo 2-opt, se considerará la ruta formada por los arcos {{1, 3}, {3, 4}, {4, 5}, {5, 8}, {8, 9}, {9, 2}, {2, 6}, {6, 7}, {7, 10}, {10, 1}}, como se muestra en la Figura 3.5.

  Si se deseara intercambiar los arcos {7, 10} y {5, 8}, por los {7, 5} y {10, 8}, o, por los {5, 7} y {8, 10} (Figura 3.6), en cualquiera de los dos casos mencionados habría que cambiar la dirección de la ruta, «invirtiendo» cuatro arcos (Figuras 3.7a y 3.7b), es por esto que se realiza el procedimiento presentado en la línea 7 del pseudocódigo 1.

  Los intercambios entre los mismos arcos son descartados, ya que no se efectuaría ningún cambio en la ruta y los intercambios entre arcos adyacentes tendrían un resultado similar como se muestra en la Figura 3.8, en consecuencia en el pseudocódigo 1 se incluye el condicional en la línea 4.

[1] Solamente hay una manera de hacerlo, sin volver a considerar los arcos removidos.

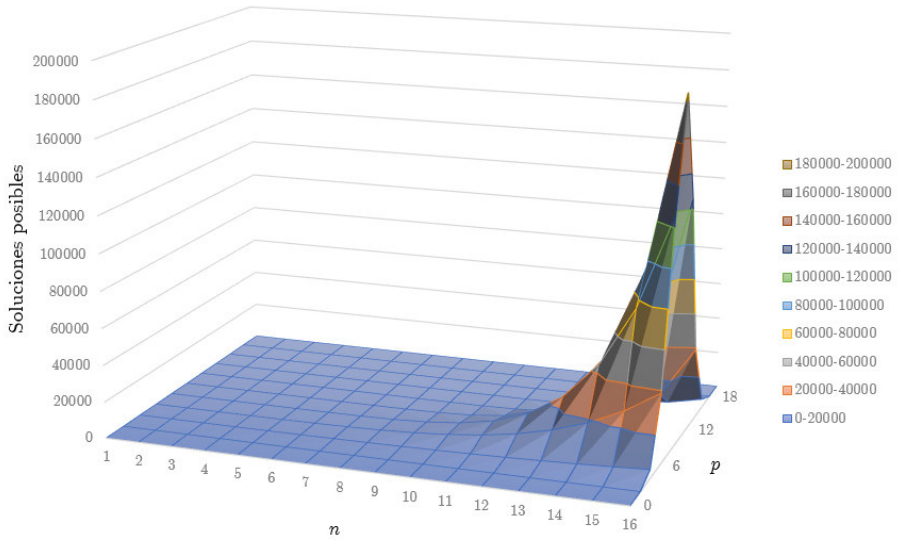


Figura 3.3: Comportamiento complejidad PMP
Fuente: Elaboración propia con información de Yakici y Yiğit (2017)

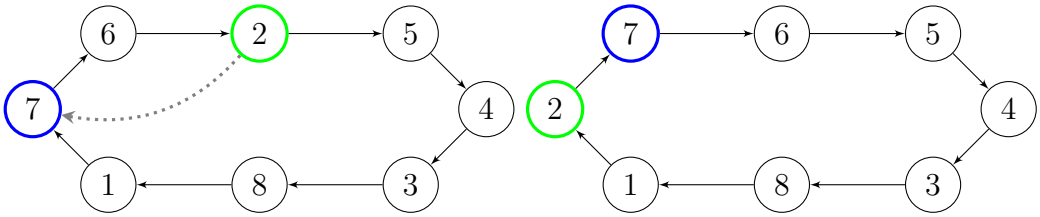


Figura 3.4: Ejemplificación de inserción de nodo
Fuente: Elaboración propia con datos de Woodruff (1998)

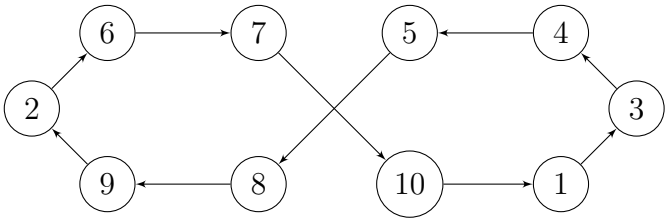


Figura 3.5: Ruta inicial, ejemplificación 2-opt
Fuente: Elaboración propia con datos de Woodruff (1998)

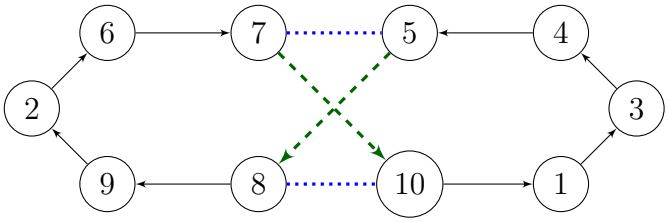


Figura 3.6: Intercambio ejemplificación 2-opt
Fuente: Elaboración propia con datos de Woodruff (1998)

Pseudocódigo 1 Búsqueda local 2-opt

```
1: Con una ruta factible previamente generada
2: Mientras no se hayan usado todos los intercambios posibles hacer
3:     Para cada combinación de arcos existentes hacer
4:         Si se trata del mismo arco o son adyacentes entonces
5:             Descartar intercambio
6:         De otro modo
7:             Realizar intercambio de arcos redireccionando el circuito
8:             Si distancia actual > distancia de intercambio entonces
9:                 Actualizar ruta
10:                Reiniciar algoritmo 2-opt
11:            De otro modo Si el intercambio ya fue usado entonces
12:                Finalizar algoritmo 2-opt
13:            De otro modo
14:                Agregar intercambio a lista de usados
15:            Fin Si
16:        Fin Si
17:    Fin Para
18: Fin Mientras
```

Fuente: Elaboración propia con información de Woodruff (1998)

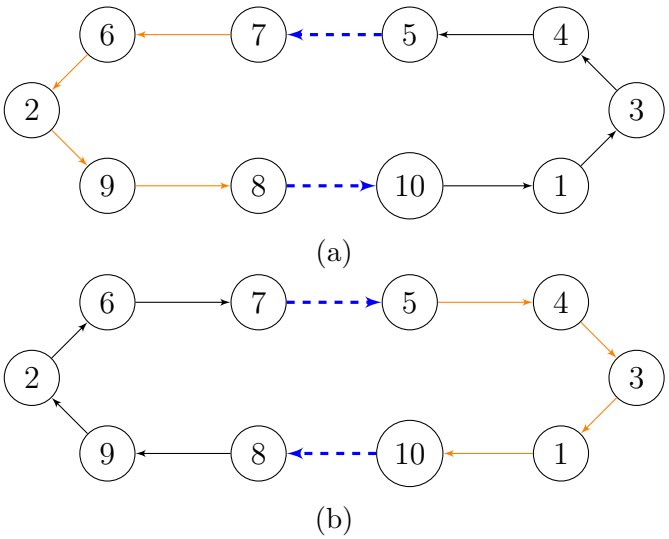


Figura 3.7: Opciones de intercambio 2-opt
Fuente: Elaboración propia con datos de Woodruff (1998)

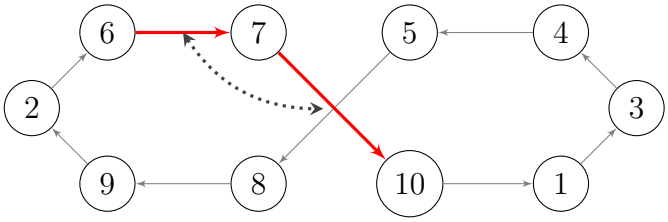


Figura 3.8: Intercambio 2-opt inadmisible; arcos adyacentes
Fuente: Elaboración propia con datos de Woodruff (1998)

- **3-opt:** Al igual que el heurístico anterior, 3-opt fue presentado por Lin (1965), y de una manera similar consta de intercambiar tres arcos, con ocho posibilidades de intercambio, conservando la posibilidad que represente el costo mínimo.

- **Lin-Kernighan:** De una manera desarrollada de los dos anteriores, el algoritmo Lin-Kernighan, propuesto por Lin y Kernighan (1973), realiza movimientos k-opt, así reduciendo el costo total del circuito, intercambiando k número de arcos (Helsgaun, 2009).

Adicional a las categorías mencionadas, Golden *et al.* (1980) definen la combinación de heurísticos de construcción y de mejora como «procedimientos compuestos».

3.3 Algoritmos metaheurísticos

Blum y Roli (2003) elaboraron una recopilación de las distintas concepciones sobre los algoritmos metaheurísticos, siendo aspectos destacables los que indican que este tipo de algoritmos corresponden a estrategias que se encargan de dirigir el proceso de búsqueda en el espacio de soluciones, con el propósito de realizar el anterior de una manera eficiente, persiguiendo el mejor resultado o, en su defecto, las soluciones lo más posiblemente cercanas al óptimo, incorporando mecanismos para evadir el estancamiento en óptimos locales y que cuentan con algún tipo de memoria. Los autores señalan, de la misma manera, que los algoritmos metaheurísticos no son específicos a un solo problema.

Por su parte Dorigo *et al.* (2006) y Bianchi *et al.* (2008) mencionan que un metaheurístico es un marco de referencia, que está compuesto por un conjunto de algoritmos usados para definir un sólo heurístico, y que puede aplicarse a distintos problemas de optimización con ciertas (usualmente pocas) modificaciones. En seguida se nombrarán algunos algoritmos sobresalientes dentro de esta categoría.

3.3.1 Búsqueda General en Entorno Variable

La Búsqueda General en Entorno Variable (*General Variable Neighborhood Search* o VNS) que define una estructura de búsqueda usada para la búsqueda local, evaluando puntos aleatorios de cada grupo, cuando una mejor solución es detectada se toma el grupo para continuar la búsqueda local (Mladenovic y Hansen, 1997).

Retomando el trabajo de Soylu (2017) (apartado 2.4.3), se aplica GNVS generando una solución factible inicial, una estructura de búsqueda de entorno (*one point move, or-opt move, two-point move y three point move*), una condición de término y un procedimiento «de sacudida» (cambia aleatoriamente un nodo dentro de un circuito) para lograr salir de un óptimo local si este no mejora con las iteraciones subsecuentes hasta que se cumple el tiempo límite ingresado por el usuario.

3.3.2 Algoritmos Genéticos

La idea principal de los Algoritmos Genéticos o GA (*Genetic Algorithms*) es operar sobre una población finita de individuos que representarán las posibles soluciones y que modifiquen su contenido en función de patrones selectivos que favorezcan los mejores resultados en un algoritmo que emula la evolución genética natural del más apto y, que como lo mencionan Osman y Laporte (1996), pueden ser vistos como métodos de búsqueda local que hacen uso de mecanismos generadores, que se enfocan en los atributos de un conjunto, en lugar de los atributos de una sola

solución, articulando por generaciones.

El número de iteraciones impacta directamente el algoritmo genético, así que para llegar a un resultado de calidad es necesario iterar el procedimiento para lograr mejorar sustancialmente el número de soluciones, por su parte Eldem y Ülker (2017) mediante la experimentación con un algoritmo ACO para el TSP, calculando la distancia mínima sobre una superficie esférica, los autores contrastan la calidad de los resultados de un GA contra los resultados del algoritmo ACO planteado por los primeros, en dónde es posible observar que usando ACO los resultados no dependen en tal magnitud del número de iteraciones para reportar un buen resultado.

Algunos trabajos para la solución del TSP mediante la inclusión de algoritmos genéticos incluyen las aportaciones siguientes.

Xu *et al.* (2017) proponen el uso de un algoritmo genético para darle solución al m-TSP, incluyendo una primera fase para la división de la carga de trabajo que se le asigna a cada agente, haciendo uso del método de k-medias y eligiendo el centro de cada grupo de ciudades que conformaran un circuito de manera aleatoria, logrando nivelar los recorridos, en términos de número de ciudades a visitar por agente. Para la segunda fase del algoritmo, los autores proponen emplear un algoritmo genético que después de evaluar los mejores valores de aptitud para cada individuo recurren a la regla de la ruleta bajo una estrategia elitista para generar y seleccionar a los individuos con mejores probabilidades de minimizar la distancia recorrida.

Para la solución del TSP también se hace uso de algoritmos genéticos multi-objetivo, como lo es el caso del trabajo propuesto por Changdar *et al.* (2014) en el que desarrolla un algoritmo modificado a través de la inclusión de un operador de refinación (que busca el cromosoma vecino dominante, cercano al estudiado, y si se encuentra, es remplazado el segundo por el primero), en dicho problema los dos objetivos a optimizar son el tiempo y el costo, con el uso de números triangulares (*fuzzy*) para tomar decisiones sobre datos poco precisos debido a la naturaleza incierta de la aplicación del TSP en un entorno real (donde el tiempo y costo puede

variar) y la inclusión de los números difusos permite crear escenarios optimistas y pesimistas con respecto a los totales, concluyendo que el algoritmo genético propuesto es competente en la experimentación contra otros algoritmos genéticos (elitistas y no elitistas).

3.3.3 Algoritmos de optimización basados en Colonia de Hormigas

Los algoritmos de optimización basados en colonia de hormigas o ACO están catalogados dentro de la inteligencia de enjambre o *swarm intelligence*, que basan sus métodos en la conducta natural de especies diferentes al ser humano (Dorigo *et al.*, 2006).

En específico, los algoritmos ACO imitan el comportamiento de forrajeo de las hormigas (Figura 3.9), que utilizan un modo de comunicación indirecto y no simbólico mediado por el entorno: depositando feromonas en el suelo para indicar que cierto camino resulta ser favorable para ser seguido por otros miembros de la colonia.

De esta manera las «hormigas artificiales» usan un mecanismo estocástico similar al usado por las hormigas reales (Goss *et al.*, 1989), utilizando la información de distancia y el grado en el que un nodo es «deseable» como siguiente movimiento (si ya fue visitado antes).

Por lo tanto, esta categoría de metaheurísticos se vale de un conjunto de hormigas artificiales que construyen rutas sobre el grafo, para la solución del TSP, con una función de probabilidad en la que influye la cantidad de feromonas y el costo asociado con cada arco para el movimiento de cada hormiga, intercambiando la información sobre la calidad de sus resultados encontrados por medio un esquema que rememora el usado por las hormigas.

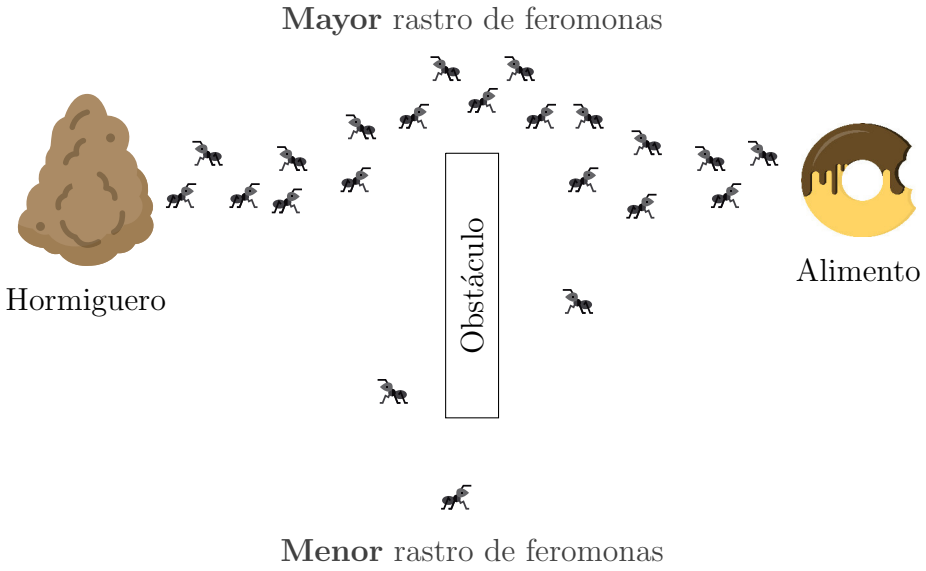


Figura 3.9: Organización de los miembros de una colonia de hormigas
Fuente: Deneubourg *et al.* (1989)

Una representación básica de como funcionan este tipo de algoritmos se puede observar en el algoritmo 2.

El primer trabajo en ACO se propuso por parte de Dorigo (1992), definiendo un «sistema de hormigas» o *ant system*, desarrollándolo sobre esa misma dirección en la investigación de Dorigo *et al.* (1996), para posteriormente aventajar el planteamiento, un año después bajo el mismo enfoque tomando como base el TSP, por su categorización dentro de la optimización combinatoria, para contrastar su método contra los metaheurísticos existentes (GA y SA, por ejemplo) al momento de su experimentación (Dorigo y Gambardella, 1997), y sucesivamente formalizando el algoritmo como un metaheurístico, incluyendo una variante orientada a la solución

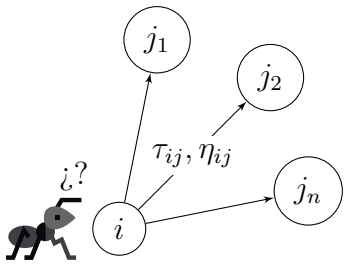


Figura 3.10: Elección del siguiente movimiento
Fuente: Dorigo *et al.* (2006)

Pseudocódigo 2 Metaheurístico ACO

1: **Mientras** condición de término no cumplida **hacer**
2: *Hormigas artificiales construyen soluciones*
3: *Aplicar búsqueda local* (opcional)
4: *Actualizar feromonas*
5: **Fin Mientras**

Fuente: (Dorigo *et al.*, 2006)

del TSP (Dorigo y Caro, 1999).

Las propuestas para solucionar el TSP mediante algoritmos ACO son variadas como la de Ismkhan (2017) que presenta un trabajo que busca contrarrestar las áreas de oportunidad que pudieran mermar el rendimiento de este tipo de algoritmos mediante la implementación de estrategias específicas; (1) usar la información de las feromonas en conjunto con un algoritmo de búsqueda local, (2) usando una nueva representación de valores de feromonas en el que a diferencia de usar la memoria para retener todos los valores de las feromonas, entre los distintos nodos, se generan matrices dispersas limitadas por un volumen máximo, (3) e incluyendo un operador genético dentro del ACO que mitigue los cuellos de botella generados en el proceso de seleccionar el siguiente movimiento. El trabajo aplica el algoritmo propuesto en problemas de TSP de distinto tamaño, que van desde los 51 nodos hasta los 1577 (y mayores), con un tiempo de respuesta de 1.11775 y 46.439 segundos (respectivamente para las dimensiones señaladas). La reducción del tiempo computacional se contrasta con el tiempo de $O(n^2)$ que tardan este tipo de algoritmos de acuerdo con el tamaño de la instancia.

Aziz (2015) propone la solución al TSP bajo una aproximación metaheurística, debido a la calidad de las soluciones arrojadas por este tipo de metodologías, con base en ACO que, a su vez pretende crear un algoritmo general que pueda ser aplicado en más de una disciplina («hiper-heurística» como el autor lo menciona). El algoritmo desarrollado se centra en el cálculo de un óptimo local y uno global, haciendo uso de heurísticas. El autor menciona que este método es comparable en calidad de resultados a métodos específicos como lo es la búsqueda tabú y el recocido simulado.

Siguiendo con los trabajos desarrollados haciendo uso del algoritmo ACO, es oportuno mencionar el trabajo de Mahi *et al.* (2015), en este método se hace uso de un algoritmo de enjambre de partículas para determinar los parámetros que afectan directamente al desempeño del modelo de colonia de hormigas, para posteriormente aplicando el algoritmo *opt-3*, descartar los óptimos locales. Concluye con un mejor desempeño ligado al menor número de hormigas artificiales introducidas como parámetro para el algoritmo ACO. Esta aproximación es comparada con resultados encontrados en la literatura arrojando un producto notablemente cercano al óptimo conocido.

Toro *et al.* (2013) planteó un modelo para el ruteo del trayecto en vacío (sin carga) de la flota de vehículos entre las ciudades capitales de Colombia, el planteamiento aclara que los vehículos que abastecen papa en las distintas ciudades requieren hacer el recorrido entre las mismas sin el producto para realizar la distribución del producto dentro de las ciudades. El recorrido se planteó con base en el problema del vendedor viajero y se plantearon distintos métodos de solución metaheurísticos que mostraban distintos desempeños siendo la mejor solución la planteada por medio de resolvió por ACO.

Para resolver el m-TSP, los algoritmos ACO han mostrado ser una alternativa a problemas reales intratables con algoritmos exactos determinísticos (Necula *et al.*, 2015).

Necula *et al.* (2018) exponen una comparación entre la aplicación de ACO sobre el m-TSP «descompuesto» en partición, mediante el método de k-medias y ruteo, contra su aplicación al modelo original, señalando que los mejores resultados se mostraron al resolver el modelo «completo», pero la solución en «dos partes» resultó ser más estable en términos de la consistencia de los resultados obtenidos. Una razón por la que ocurre lo anterior, es porque la partición limita severamente el espacio de soluciones.

3.3.4 Otros algoritmos de solución

Existen trabajos con el uso de algoritmos relativamente nuevos, como lo es el caso de Venkatesh y Singh (2014) quienes proponen dos aproximaciones distintas para resolver el m-TSP, usando dos objetivos diferentes. El primer objetivo es minimizar la distancia total recorrida por todos los agentes basándose en un algoritmo de colonia artificial de abejas (ABC por sus siglas en inglés) que se basa en el comportamiento de las abejas buscando polen y tipifica a las abejas artificiales en exploradora (busca una solución), trabajadora (exploran y comparten la aptitud de la solución) y espectadora (evalúan las posibles soluciones proporcionadas por las trabajadoras y pasan a ser de este tipo), rotando sus funciones en cuanto se encuentra una posible solución.

El segundo objetivo (del trabajo anterior) busca minimizar la distancia máxima recorrida por los agentes, a través de un algoritmo IWO que comienza con soluciones generadas de forma aleatoria (hierbas) y en cada iteración cada hierba produce cierto número de soluciones (semillas) dependiendo de su aptitud para convertirse en hierba, produciendo soluciones aleatorias que siguen una distribución normal pero cambiando su varianza causando una búsqueda aleatoria alrededor de cada hierba (posible solución). Aproximaciones que los autores contrastan con métodos usualmente utilizados, y que en la experimentación tuvieron una ligera mejora sobre el desempeño de ACO seguido por los algoritmos genéticos, esto en instancias sin búsquedas locales dentro de los algoritmos.

Para el caso de la partición de nodos a repartir para cada agente, el uso del PMP es una alternativa *ad hoc* al presente planteamiento, como se mencionó anteriormente, y siendo que la complejidad de solución no representa un sesgo significativo con respecto al óptimo para problemas de mayor volumen, se hará uso de el algoritmo de ramificación y corte para la etapa en cuestión.

No es imperioso encontrar la solución óptima a un TSP para algunas aplica-

ciones prácticas, como lo sugiere Reinelt (1994), algunas de las posibles razones que podrían sustentar esta aseveración son:

- Modelado incorrecto: El problema real que se pretende atacar, puede diferir del modelo planteado, y por lo tanto, la solución óptima del modelo no es necesariamente la del primero.
- Tiempo: La preparación del ruteo podría encontrarse con un tiempo limitado e insuficiente para calcular el mejor resultado posible al modelo.
- Dimensiones del problema: El número de nodos complica llegar al óptimo conforme los anteriores aumentan en número, aunque, hasta cierto punto, las innovaciones para los algoritmos exactos aumentan su alcance, en razón de dimensiones abarcables, y este argumento va perdiendo vigencia.

Siguiendo la tónica anteriormente mencionada, y considerando los algoritmos vistos hasta ahora, el metaheurístico ACO, presenta ser una alternativa adecuada para la solución de cada TSP generado por la partición.

Capítulo 4

Metodología

Concerniente a la solución del problema planteado en el caso de estudio, en los capítulos precedentes se pormenorizaron las formas de diseñar y resolver el ruteo de vehículos.

Anteriormente se ha delimitando el planteamiento a la modelación matemática a través del modelo del problema del agente viajero múltiple, y su posterior «división» para su solución en un algoritmo de «partición primero y posterior ruteo», que se encuentra dentro de los métodos de solución aproximados.

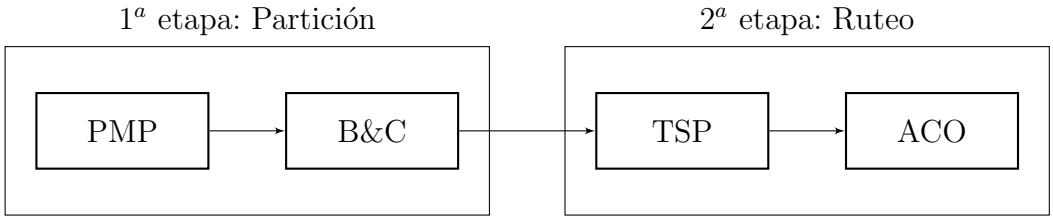


Figura 4.1: Diagrama general del algoritmo propuesto
Fuente: Elaboración propia con información de Xu *et al.* (2017), Yakici y Yiğit (2017), Padberg y Rinaldi (1991), Lawler *et al.* (1985) y Dorigo *et al.* (2006)

En este capítulo, primero se oteará el método para la asignación de nodos a cada agente (o vehículo), para después pasar al algoritmo basado en ACO que devolverá una ruta factible para cada agente.

4.1 Condiciones preliminares

Para la aplicación de la presente metodología se presupone que se cuenta con un número finito de vehículos que transitarán por la infraestructura terrestre aplicable al proyecto, sin considerar costos asociados con el cruce de fronteras ni operativos (por ejemplo: el tiempo máximo que un conductor puede operar su unidad).

Se considera únicamente el ruteo de m vehículos sin hacer distinción de las características de los mismos, así como de capacidad y ventanas de tiempo.

También se ha de considerar que todos los puntos (nodos) que visitarán los vehículos deben de estar conectados por infraestructura terrestre (carreteras, caminos, autopistas, etc.), o en su defecto por un transbordo que le permita al vehículo llegar de un punto a otro, por ejemplo, un ferri.

El objetivo de las fases mencionadas a continuación es generar la mejor ruta posible para todos los vehículos con un tiempo de cómputo que no exceda la proporción mencionada en el apartado 3.3.3, generando un margen de adaptabilidad del ruteo según lo requiera la implementación en caso de modificar los puntos de interés a visitar por los mismos durante el recorrido.

4.2 Cálculo de distancias

Sin contar las instancias existentes que sirven como una métrica de referencia para comparar el desempeño de algún método de solución (y entre métodos), en este caso para el TSP, por ejemplo, la colección de instancias «TSPLIB» (Reinelt, 1991), en distintos trabajos como el propuesto por Xu *et al.* (2017), se utilizan datos relativos a las distancias euclidianas para el cálculo de la distancia entre los nodos, tomando como referencia el posicionamiento geográfico, sin embargo, cuando se pretende abordar un problema aplicable a vehículos que transitarán la infraestructura

carretera, es necesario tomar en cuenta las distancias a lo largo de estas vías.

Una forma de obtener las distancias es a través de un servicio de posicionamiento geográfico que cuente con este tipo de datos. Google Inc. (2017) hace posible esta consulta en formatos compatibles con el desarrollo de una interfaz de lectura en un lenguaje de programación abierto.

En el presente trabajo se considera la recopilación de distancias carreteras, **como referencia y con fines cien por ciento de investigación**[1], a través del servicio anteriormente mencionado debido a su mejora continua en su base de datos, accesibilidad, actualización y asequibilidad. Se programó la solicitud de los datos en lenguaje Python 3.6.3 (ver código fuente B.1 del Apéndice B), siguiendo los pasos del pseudocódigo 3.

Pseudocódigo 3 Solicitud de datos a *Google Distance Matrix API*

Lectura de datos de ubicaciones (latitud y longitud)
Para ubicación $i = 1$ **hasta** número de nodos **hacer**
    Para destino $j = 1$ **hasta** número de nodos **hacer**
        Formular solicitud de distancia desde la ubicación i hasta la ubicación j
        Enviar solicitud y recibir resultado de la API
        Agregar dato a la matriz de distancias c_{ij}
    Fin Para
Fin Para
Generar matriz de distancias

Fuente: Elaboración propia

Para el uso del presente método de solución se deben de contar con las latitudes y longitudes definidas de antemano de los puntos de interés a visitar, así como del nodo de salida, como se ejemplifica en la Figura 4.2.

[1] Apegándose a los criterios de Google Inc. (2011), al momento de la publicación del presente documento.

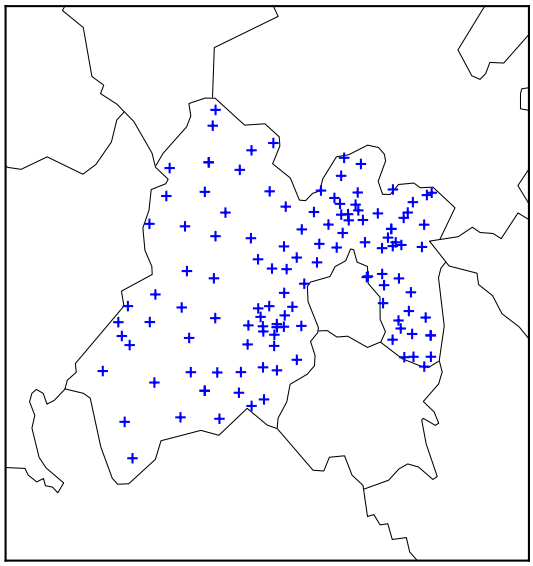


Figura 4.2: Ejemplo de 125 nodos en el Estado de México
Fuente: Elaboración propia con datos de Instituto Nacional de Estadística y Geografía (2017)

4.3 Asignación de nodos

Los puntos de interés a visitar bajo una perspectiva geográfica se encuentran concentrados en ciudades, es por esto que se comienza agrupando dichos puntos bajo el modelo del PMP (apartado 2.4.7) para su reparto entre los agentes en función de la ciudad en la que se ubican.

Al modelo mencionado se le agregará una restricción para delimitar el número máximo de nodos que se le pueden asignar a una mediana, esto con el fin de «nivelar la carga» de trabajo entre los vehículos, y un elemento que minimice la distancia desde cada mediana hacia el nodo de inicio, con el objetivo de generar subconjuntos que después serán conectados con el nodo de inicio para generar las rutas.

Observando la distancia desde cada posible mediana hasta el nodo de inicio como el parámetro d_j.

Sin incluir el nodo de inicio (que será añadido posteriormente para generar las

rutas) en la asignación, el modelo resulta como se muestra a continuación.

$$\text{mín} \sum_{i \in I} \sum_{j \in J} c_{ij} x_{ij} + \sum_{j \in J} d_j y_j \tag{4.1}$$

$$\text{s.a:} \sum_{j \in J} x_{ij} = 1 \qquad \forall i \in I \tag{4.2}$$

$$x_{ij} \leq y_j \qquad \forall i \in I, j \in J \tag{4.3}$$

$$\sum_{j \in J} y_j = p \tag{4.4}$$

$$\sum_{i \in I} x_{ij} \leq Q \qquad \forall j \in J \tag{4.5}$$

$$x_{ij}, y_j \in \{0, 1\} \qquad \forall i \in I, j \in I \tag{4.6}$$

donde (4.1) representa la función objetivo que minimiza la distancia total[2] desde cada nodo hacia su correspondiente mediana asignada y la distancia desde cada mediana hasta el nodo de inicio, las restricciones (4.2), (4.3) y (4.4), corresponden a las (2.30), (2.31) y (2.32), respectivamente, detalladas en el apartado 2.4.7, la desigualdad (4.5), asegura que el número de nodos asignados a cada mediana (en este caso cada agente) sea equitativo, en razón de $Q = \left\lceil \frac{n}{p} \right\rceil$, terminando con la restricción de estado (4.6).

Resolviendo entonces, el PMP modificado con el algoritmo de ramificación y corte (subsección 3.1.2), para lo cuál se hará uso del *solver* CPLEX a través de la programación en GAMS, como se muestra en el código fuente C.1 del Apéndice C.

Retomando el ejemplo del Estado de México, partiendo de Toluca, para 5 vehículos, una posible asignación se muestra en la Figura 4.3, que permite la visualización del vínculo entre las medianas y el nodo de partida, así como el de las medianas a cada uno de los elementos de su subconjunto.

[2]Generalmente representadas por valores positivos mayores a cero.

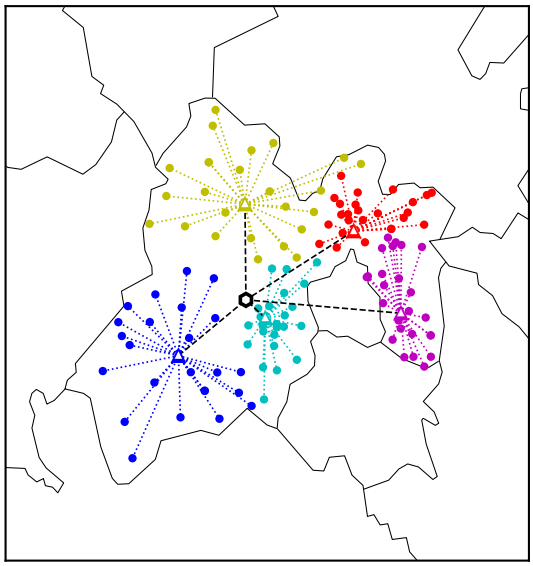


Figura 4.3: Ejemplo de la asignación de nodos para ejemplo Estado de México
Fuente: Elaboración propia con datos de Instituto Nacional de Estadística y Geografía (2017)

4.4 Solución de ruteo

Una vez realizada la asignación de los nodos a cada agente, se obtendrá el mejor circuito encontrado por el algoritmo ACO, que es un algoritmo iterativo en el que cierto número de hormigas artificiales construyen la solución viajando de un nodo a otro sobre el grafo con la restricción de visitar cada nodo solamente una vez, visitando la totalidad de los mismos.

Los pasos a seguir por el algoritmo ACO seguirán la pauta del algoritmo ESA-CO (*Effective Strategies* + ACO) propuesto por Ismkhan (2017), observados en el pseudocódigo 4.

La programación del algoritmo mostrado se desarrolla en el apéndice D. Para comenzar a describir el algoritmo que resolverá el ruteo de cada vehículo se definirá la notación a usar durante este apartado.

El número de hormigas estará representado por el conjunto $K = \{1, 2 \ldots, k\}$,

Pseudocódigo 4 ACO propuesto

```
1: Inicialización de parámetros
2: Para iteración = 1 hasta número de iteraciones hacer
3:     Para i = 1 hasta número de nodos hacer
4:         Para k = 1 hasta número de hormigas hacer
5:             Hormiga k selecciona el siguiente arco
6:             Hormiga k actualiza valor de feromona del movimiento seleccionado
7:             Si i = número de nodos entonces
8:                 Hormiga k mejora su recorrido con algoritmo de búsqueda local 2-opt
9:             Fin Si
10:        Fin Para
11:    Fin Para
12:    Actualizar: mejor circuito global, valores de feromonas a nivel global pertenecientes al mejor circuito y conjunto de candidatos
13: Fin Para
```

Fuente: Ismkhan (2017)

los nodos y arcos seguirán siendo la pareja ordenada ($\mathcal{N}$,$\mathcal{A}$) que se describió en el apartado 2.4.1, el nodo de inicio en cualquier movimiento dado será i, resultando j como el nodo de destino, el conjunto de movimientos factibles para la hormiga k desde el nodo i será S_i^k.

Cada arco (i, j) está asociado con una variable llamada «feromona» que puede ser leída y modificada por cada hormiga, τ_{ij} simbolizará el valor de la misma. La información disponible para el metaheurístico será η_{ij} definida por la ecuación (4.7), donde d_{ij} es la distancia entre los nodos i y j (Dorigo *et al.*, 2006).

$$\eta_{ij} = \frac{1}{d_{ij}} \tag{4.7}$$

4.4.1 Selección de candidatos

Como ya se ha insistido anteriormente, el algoritmo propuesto por Ismkhan (2017), reporta tiempos de cómputo eficientes, es por esto que se adoptarán las

estrategias propuestas por el autor citado. Antes que nada, la elección y actualización de los nodos «candidatos» será un proceso crucial para el cálculo de nuevas rutas.

Los candidatos se refieren a los nodos que, con respecto a uno de referencia, representan mayor elegibilidad para ser seleccionados en un siguiente movimiento y son calculados inicialmente jerarquizando su posición con respecto a un criterio de vecino más cercano, es decir, para determinado nodo, su primer candidato será su vecino, más cercano, su segundo candidato será su segundo nodo más cercano, continuando así respectivamente.

Para actualizar cada lista de candidatos se usará el proceso detallado en el pseudocódigo 5.

Pseudocódigo 5 Actualización de nodos candidatos

1: **Para** $n = 1$ **hasta** número de nodos **hacer**
2: Mejor candidato = $\arg\max\left\{\tau_{n0}, \tau_{n1}, \ldots, \tau_{n(\text{número de nodos}-1)}\right\}$
3: Remover el último nodo del conjunto de candidatos para el nodo n
4: Insertar Mejor candidato en la primera posición del conjunto de candidatos para el nodo n
5: Remover el último nodo del conjunto de candidatos para el nodo n
6: Insertar el nodo sucesor de n en el mejor circuito global en la primera posición del conjunto de candidatos para el nodo n
7: **Fin Para**

Fuente: Ismkhan (2017)

La actualización de nodos candidatos no permite duplicados y se actualiza constantemente a cada iteración.

4.4.2 Construcción de soluciones

Las k hormigas artificiales comienzan a construir la ruta para la solución del TSP desde un nodo aleatorio, eliminándolo de su respectivo e individual conjunto de movimientos factibles S_i^k. Para los siguientes arcos a considerar Dorigo *et al.* (1996) establecen en su «sistema de hormigas» la articulación que seguirá la construcción de soluciones.

Cada hormiga k puede seguir un mecanismo estocástico para la construcción de soluciones, conocido como función proporcional aleatoria (*random proportional rule*) en el que, cuando se encuentra con una solución parcial en el nodo i, la probabilidad de que se elija el siguiente nodo j está establecida por la expresión (4.8).

La mejora de cada ruta construída para cada hormiga artificial se efectúa con el algoritmo 2-opt, pero, para aumentar la eficiencia de la búsqueda, el intercambio de arcos se efectúa únicamente con los cuatro mejores candidatos para cada intercambio.

$$p_{ij}^{k} = \begin{cases} \frac{\tau_{ij}^{\alpha} \cdot \eta_{ij}^{\beta}}{\sum_{l \in S_i^k} \tau_{il}^{\alpha} \cdot \eta_{il}^{\beta}} & \text{si } j \in S_i^k \\ 0 & \text{de otro modo} \end{cases} \tag{4.8}$$

donde l es un nodo que la hormiga k aún no ha visitado, luego entonces, los elementos factibles estarán formados por los arcos (i,l) que pertenecerán al conjunto denotado por S_i^k, α y β son parámetros que definirán la importancia del valor de las feromonas contra la información disponible para el metaheurístico.

En lo que respecta al algoritmo ACO aplicable al presente método de solución, la selección del siguiente arco (i, j) o movimiento se efectúa a través de la función conocida como proporcional pseudoaleatoria (*pseudorandom proportional rule*) dada por la expresión (4.9).

$$j = \begin{cases} \arg \max_{l \in S_i^k} \left\{ \tau_{il} \left[\eta il \right]^{\beta} \right\} & \text{si } q \leq q_0 \\ J, & \text{de otro modo} \end{cases} \tag{4.9}$$

donde q_0 es un parámetro que puede tomar valores en el rango de $0 \leq q_0 \leq 1$ y q es un número aleatorio uniformemente distribuido en el intervalo $[0, 1]$. J efectúa la elección del siguiente nodo bajo los lineamientos de la función proporcional aleatoria (4.8). Esto hace posible que, ajustando el parámetro q_0, se efectué una búsqueda sesgada concentrando la búsqueda en la mejor solución global o que se diversifique la exploración (Dorigo y Stützle, 2004).

En el trabajo de Dorigo *et al.* (1996) se mostró la experimentación de distintas combinaciones de los valores α y β para evitar el estancamiento en un óptimo local. En la figura 4.4 se puede observar que altos valores de α generan estancamiento en óptimos locales (símbolo ▲), mientras que descartar la importancia del valor de las feromonas conduce a soluciones no muy buenas (símbolo ■), por último, las mejores soluciones se encontraron con las combinaciones denotadas por ❍.

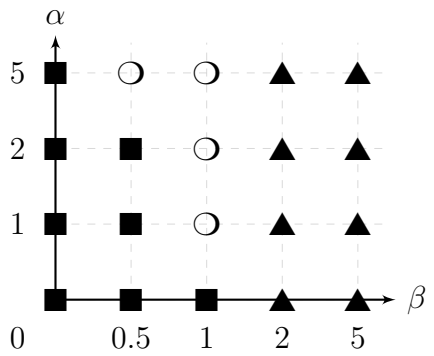


Figura 4.4: Comportamiento con distintos valores de α y β
Fuente: (Dorigo *et al.*, 1996)

Para el caso de la aplicación de la función proporcional aleatoria Dorigo y Stützle (2004) indican el uso del parámetro = 1 para ACO, además Ismkhan (2017) sugiere, fundamentándose en los resultados de su experimentación, que el valor ideal de $\beta = 2$ como el ideal para la ejecución del metaheurístico.

Una vez definidas las funciones para la elección de cada movimiento, se debe aclarar que la elección para cada uno de los mismos se realiza con respecto a los nodos candidatos calculados y actualizados como se indica en el apartado 4.4.1. Por consiguiente la elección de cada movimiento se efectúa bajo los criterios señalados en el pseudocódigo 6.

4.4.3 Actualización de feromónas

La representación de la matriz que contiene los valores asociados con la feromona, que es el grado de elegibilidad de cada arco en razón de su uso anterior por

Pseudocódigo 6 Instrucciones para el siguiente movimiento

1: Aplicar ecuación (4.9) para los nodos candidatos de i, si no es posible seguir al siguiente paso
2: Aplicar ecuación (4.9) para los nodos de i que contengan valores de feromona, si no es posible seguir al siguiente paso
3: Aplicar ecuación (4.9) para todos los nodos no visitados

Fuente: Ismkhan (2017)

las hormigas artificiales durante la construcción de soluciones, representa un área de oportunidad en términos de memoria del equipo de cómputo usado.

Ismkhan (2017) identifica que para instancias mayores a cuatro mil nodos, el número de valores modificados en la matriz de feromonas es del orden de $50 \cdot n$, dejando los demás valores en cero, ajustando con la definición las matrices dispersas (Duff *et al.*, 2017), que explota esta característica para ahorrar espacio en la memoria del ordenador, y es por esta razón que el autor decide representar dicha información en una matriz dispersa, actualizando unicamente los valores necesarios en cada iteración.

Ahora bien, una vez definida su representación, se verán las rutinas usadas por las hormigas artificiales durante la construcción de soluciones para leer y modificar esta información, resultando en una comunicación indirecta como la existente en la especie real.

Al final de las iteraciones, los valores de las feromonas depositadas en los arcos se actualizan para coincidir con los de las mejores soluciones en razón de que sea posible construir soluciones similares en las iteraciones futuras, incrementando los valores de las feromonas asociadas con las mejores soluciones, descartando las peores soluciones con la «evaporación de feromonas» (parámetro ρ).

Las soluciones obtenidas por las k hormigas durante una determinada iteración, asocian a cada arco (i,j) que compone su solución con un valor de feromona τ_{ij}, que

se actualiza en razón de la expresión (4.10).

$$\tau_{ij} \leftarrow (1-\rho)\cdot\tau_{ij} + \rho\Delta\tau_{ij}^{ms} \qquad \forall(i,j)\in T^{ms} \tag{4.10}$$

donde $\rho \in (0,1]$ es la tasa de evaporación, τ_{ij}^{ms} se refiere a la cantidad de feromonas depositadas en el arco (i,j) por la hormiga que presentó la mejor solución hasta el momento, causando que la adición del elemento multiplicado por ρ y de su complemento generen un resultado que puede interpretarse como un promedio ponderado del valor original con respecto al nuevo, y considerando los arcos no visitados con valor de cero como se muestra en la ecuación (4.11), donde T^{ms} representa la longitud del circuito construido por la hormiga k (Dorigo y Stützle, 2004).

$$\tau_{ij}^{ms} = \begin{cases} \frac{1}{T^{ms}} & \text{si el arco } (i,j) \text{ pertenece a la mejor solución calculada} \\ 0 & \text{de otro modo} \end{cases} \tag{4.11}$$

En desarrollos sobre ACO, una de las aportaciones con mayor impacto en el desempeño de este algoritmo es la inclusión de la «actualización local de feromonas» (también llamada *"offline" pheromone update*), que busca, de manera similar que la actualización global, depositar un rastro de feromonas durante las iteraciones después de añadir un arco en cada una de las soluciones en construcción, como se muestra en la expresión (4.12), donde τ_0 es el valor inicial de la feromona y $\varepsilon \in (0,1]$ es el coeficiente de deterioro de la misma (Gambardella y Dorigo, 1996).

$$\tau_{ij} \leftarrow (1-\varepsilon)\cdot\tau_{ij} + \varepsilon\cdot\tau_0 \tag{4.12}$$

La diversificación de la búsqueda con este parámetro se puede impulsar volviendo «menos deseables» los arcos ya visitados por las soluciones en construcción, como lo sugieren Dorigo y Stützle (2004) con un valor de 0.1, mientras que Ismkhan (2017) identifica el valor de 0.6 como el mejor para éste parámetro.

Dorigo y Stützle (2004) indican que un valor deseable para τ_0 es $\frac{1}{n\cdot D^{NN}}$ donde n

es el número de nodos de la instancia a resolver, y D^{NN} es la distancia de la solución bajo el heurístico de construcción vecino más cercano. Ismkhan (2017) menciona que ρ puesto a un valor de 0.9 resulta en los mejores resultados.

Continuando con el ejemplo del Estado de México, las rutas calculadas con el algoritmo descrito en el presente apartado se visualizan en la Figura 4.5.

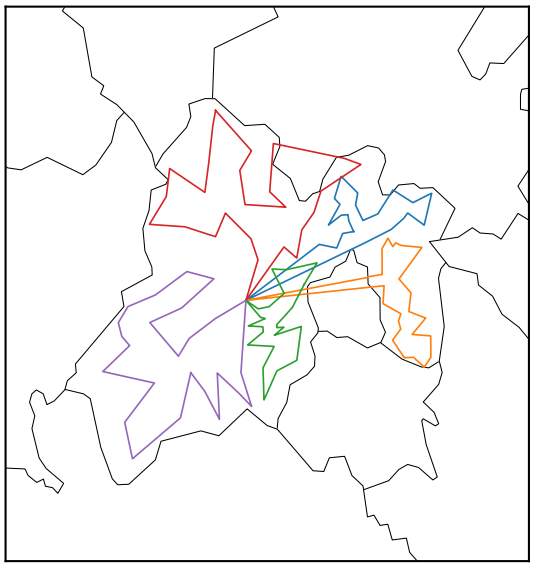


Figura 4.5: Ejemplo del ruteo de nodos para ejemplo Estado de México
Fuente: Elaboración propia con datos de Instituto Nacional de Estadística y Geografía (2017)

Capítulo 5

Experimentación computacional

Para medir la efectividad del algoritmo propuesto se retomarán algunas instancias usadas en la literatura, provenientes del TSPLIB (Reinelt, 1991) y del trabajo de Cirasella *et al.* (2001), como se muestra en la Tabla A.1 del Apéndice A, debido a que se tratan de instancias «conocidas» y algunas tomadas de problemas reales (Aguayo *et al.*, 2017).

De la misma manera se considerarán instancias generadas con datos del INEGI, obteniendo la ubicación de los municipios de algunos estados de la República Mexicana, nombrándolas conforme al número de nodos que contienen: Aguascalientes (ags11), Coahuila (coah38), Nuevo León (nl51) y Estado de México (edomex125), basando las ubicaciones en el centro geográfico de cada municipio.

5.1 PMP

Para la etapa de la partición de nodos, el algoritmo se aplicó a las instancias del TSPLIB indicadas en la Tabla A.1 con el propósito de medir el tiempo en el que el modelo puede ser resuelto con CPLEX y bajo la sintaxis de GAMS. Se ejecutó para 5 agentes y los resultados se muestran en la Tabla 5.1 del Apéndice A.

Por consiguiente es posible visualizar en la Figura 5.1 que el método de so-

Tabla 5.1: Resultados de la partición para las instancias consideradas

Instancia	Mínimo	Tiempo	Brecha	Instancia	Mínimo	Tiempo	Brecha
br17	46	00:00:01	0 %	dc126	89563	00:00:11	0 %
atex3	7434	00:00:06	0 %	ftv130	8107	00:00:36	0 %
ftv33	2197	00:00:01	0 %	dc134	4696	00:00:15	0 %
ftv35	2473	00:00:01	0 %	ftv140	9101	00:01:37	0 %
ftv38	2500	00:00:01	0 %	ftv150	9866	00:01:31	0 %
p43	10675	00:00:01	0 %	ftv160	10673	00:05:54	0 %
ftv44	3011	00:00:01	0 %	ftv170	11387	00:10:09	0 %
atex4	8988	00:00:01	0 %	dc176	7062	00:00:29	0 %
ftv47	3391	00:00:01	0 %	dc188	8028	00:00:37	0 %
ry48p	22454	00:00:01	0 %	code198	5000025097	00:01:36	0 %
ft53	17953	00:00:02	0 %	code253	5000433152	00:02:13	0 %
ftv55	3608	00:00:01	0 %	rbg323	4491	00:21:21	0.11 %
ftv64	4201	00:00:02	0 %	rbg358	4991	00:23:34	0.11 %
ft70	59358	00:00:13	0 %	rbg403	5886	00:27:29	0.16 %
ftv70	4607	00:00:02	0 %	rbg443	6518	00:32:39	0.18 %
atex5	18250	00:00:14	0 %	dc563	21409	00:42:09	0.00 %
ftv90	4919	00:00:15	0 %	atex8	237235	01:15:22	0.20 %
ftv100	5642	00:00:26	0 %	big702	491527	02:06:50	0.41 %
ftv110	6431	00:00:20	0 %	dc849	33851	03:03:12	0.00 %
dc112	8025	00:00:07	0 %	dc895	86471	04:34:05	0.00 %
ftv120	7136	00:00:27	0 %	dc932	322662	05:19:58	0.00 %

Fuente: Elaboración propia

lución exacto tiene un comportamiento exponencial con respecto a las instancias del TSPLIB, aunque, el número de nodos que representa la escalada de tiempo de solución no es representativo para el caso de estudio.

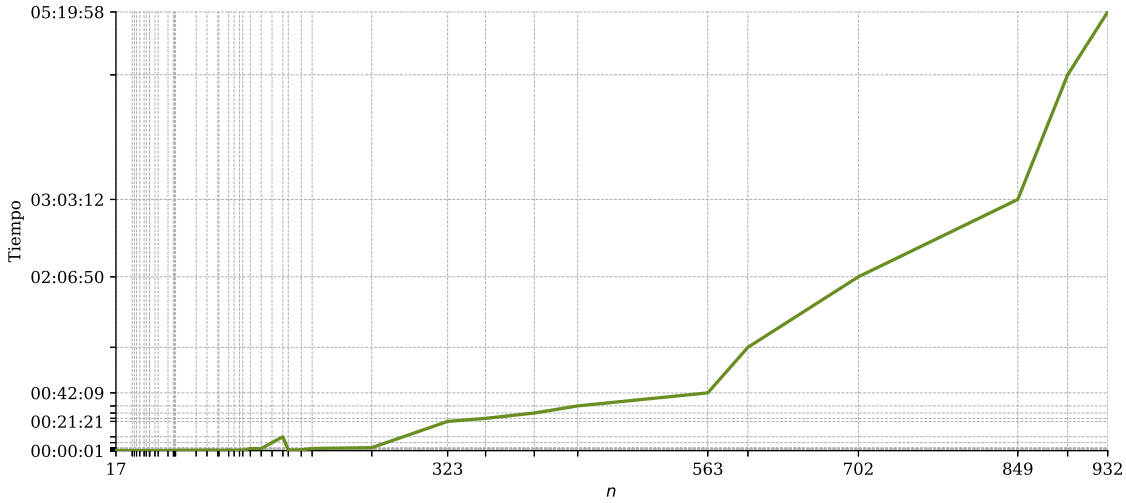


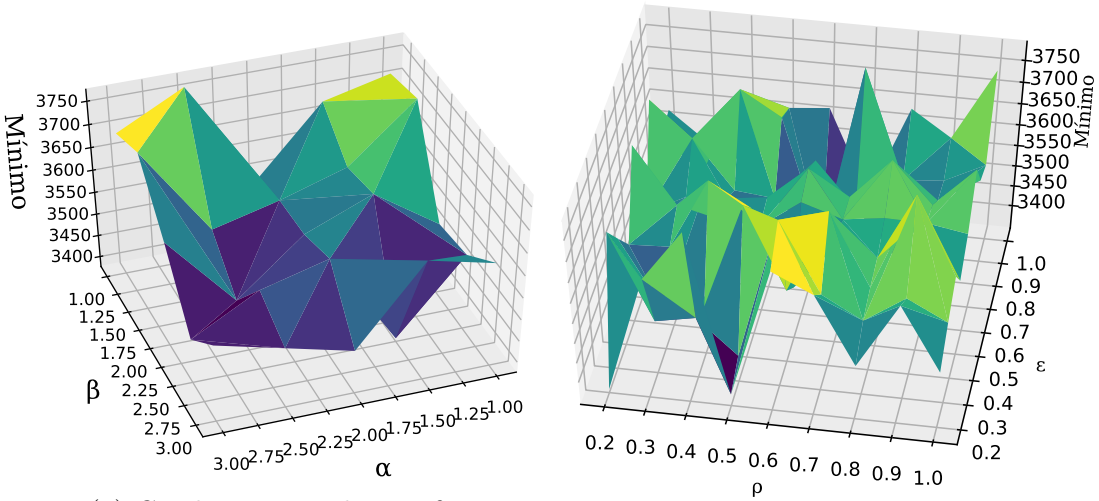
Figura 5.1: Escalada de tiempo para el modelo PMP propuesto
Fuente: Elaboración propia

5.2 ACO

Una vez que se realiza la partición, la ruta es resuelta por el algoritmo ACO propuesto. Como primer paso se analizarán las resultados del algoritmo de ruteo usando los parámetros recomendados por los trabajos antecesores (apartado 4.4).

Se realizaron pruebas con los parámetros de α y β, siendo que el primero está recomendado para ser usado con un valor de 1 y el segundo en un rango de 2 a 2.5, con incrementos de 0.5. La Figura 5.2b muestra los resultados con las distintas combinaciones de estos parámetros, siendo los valores de $\alpha = 1.5$ y $\beta = 2.5$ los que se asocian con el valor mínimo para la instancia de ftv150.

De la misma manera, para la misma instancia, se probaron distintos valores de ρ y ε, en un rango de 0.2 a 1, en intervalos de 0.1; reportando los mejores resultados con los valores $\rho = 0.6$ y $\varepsilon = 0.9$, y $\rho = 0.5$ y $\varepsilon = 0.2$ para la instancia ftv150, como



(a) Combinaciones de α y β

(b) Combinaciones de ρ y ε

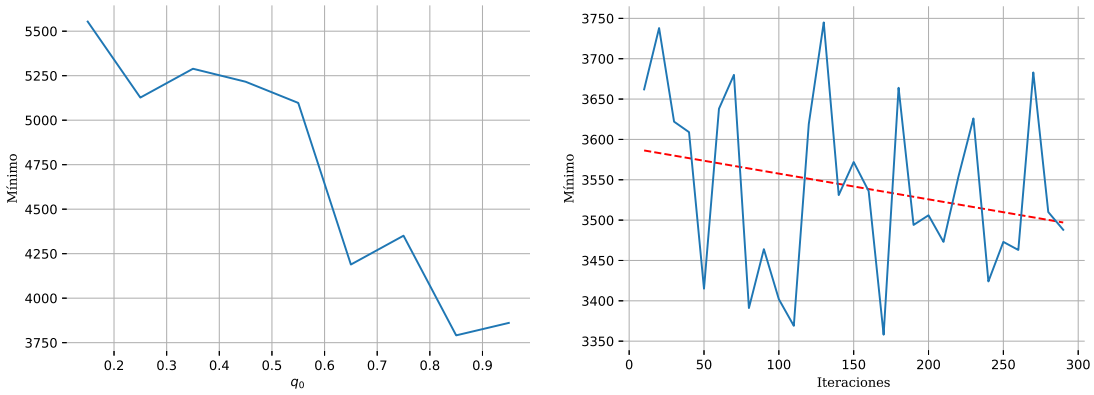
Figura 5.2: Combinaciones de parámetros para la instancia ftv150
Fuente: Elaboración propia con datos de Reinelt (1994); Ismkhan (2017); Dorigo y Stützle (2004)

se muestra la Figura 5.2b.

Bajo la misma tónica, se escaló el parámetro q_0 en un intervalo de 0.15 a 0.95 con crecimientos de 0.1, rindiendo los resultados mostrados en la Figura 5.3a para instancia ftv150, en este sentido, el número de iteraciones mejora el resultado, como se ve la aplicación de distinto número de las mismas, para la instancia mencionada anteriormente, en la Figura 5.3b.

Continuando con las pruebas en los parámetros, la misma lógica se aplicó para la instancia dc188, generando los resultados mostrados en la Figura 5.4. Los resultados para los parámetros continuaron mostrando buenos resultados para los valores de α y β con valor de 2.

Con el objetivo de confirmar los mejores parámetros para α y β se realizó experimentación de tal manera que los demás parámetros se tomarán como grupo control de las variables y los parámetros mencionados al principio serán variados para analizar el tiempo y la mínima distancia alcanzada. El grupo control estará formado por los parámetros: número de iteraciones, número de hormigas, q_0, ε y



(a) (b)

Figura 5.3: Parámetros e iteraciones para la instancia ftv150
Fuente: Elaboración propia con datos de Reinelt (1994); Ismkhan (2017); Dorigo y Stützle (2004)

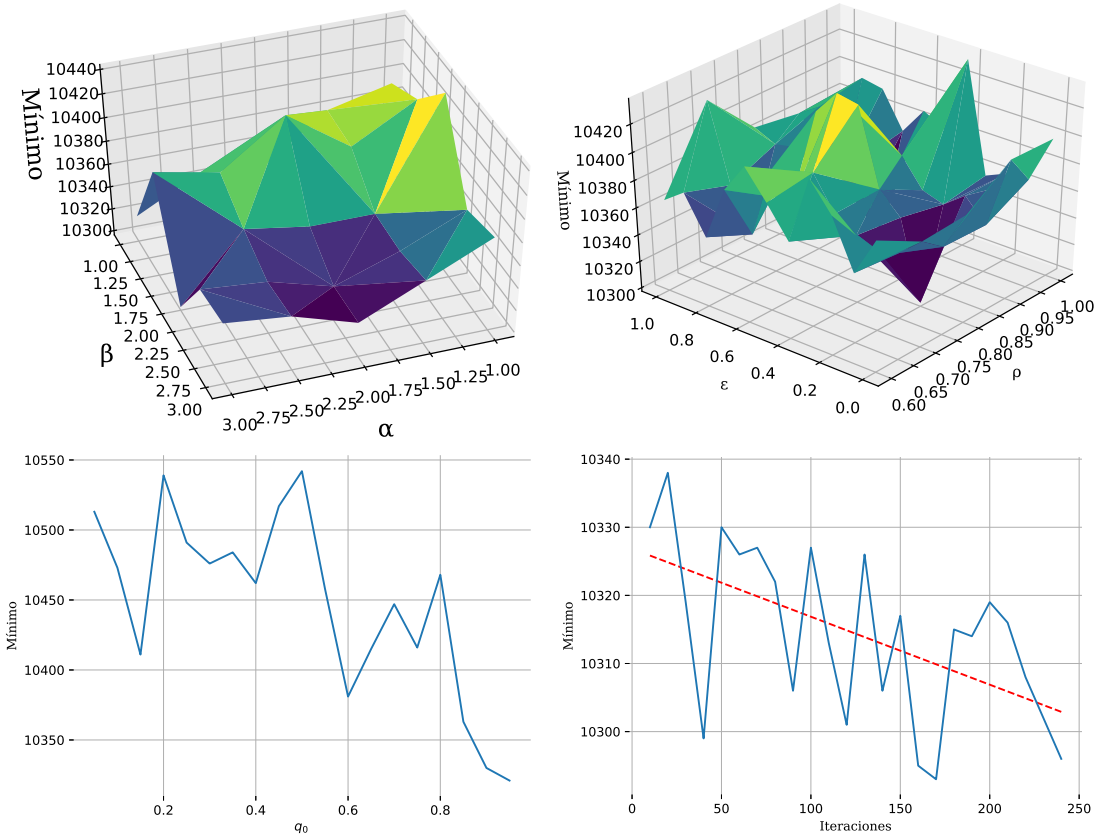


Figura 5.4: Resultados de parámetros para la instancia dc188
Fuente: Elaboración propia con datos de Reinelt (1994); Ismkhan (2017); Dorigo y Stützle (2004)

ρ. Se mantendrán los valores de iteraciones y hormigas ambos en 10, $q_0 = 0.98$, ε tendrá un valor de 0.6 y ρ valdrá 0.9. La primera instancia analizada será la ftv38, los valores de α y β oscilarán en el rango de (0.2, 5) con intervalos de: 0.2, 0.5, 0.7 y 1.

Se filtraron los resultados conforme a los mejores resultados tomando como criterios la distancia y el tiempo. De los 799 resultados obtenidos es posible inferir un valor mínimo alcanzado de hasta 1630 en la distancia mínima alcanzada, seleccionando los mismos y posteriormente filtrando con respecto tiempo, de 3.3 segundos. De tal manera que los mejores resultados considerados se reducen a 13, lo cual simplifica el análisis.

Los valores favorables para la Instancia ftv38 de α y β serían 1.73 para α y 4 para β. Se realizó la misma experimentación, con las instancias ft53, ft70, y dc112. Realizando el mismo procedimiento descrito anteriormente, se obtuvieron los resultados de los promedios para los mejores valores de α y β en sus respectivas instancias, como se puede observar en la Tabla 5.2.

Tabla 5.2: Resultados de la experimentación α y β

Instancia	α	β
Ftv38	1.73	4
Ft53	2.37	3.14
Ft70	2.37	3.70
Dc112	1.30	3.26

Fuente: Elaboración propia

Para la comparación con los valores de α y β predeterminados por los autores Ismkhan (2017) y Dorigo y Stützle (2004), se obtuvo el promedio de las instancias de la experimentación y se compararon con las de la literatura, como se puede observar en la Tabla 5.3.

Los resultados previamente obtenidos indican que los parámetros sugeridos por Ismkhan (2017) y Dorigo y Stützle (2004) rinden resultados aceptables, es por esto que se realizó una experimentación para todas las instancias incluidas en la Tabla A.1 del Apéndice A, resolviéndolas para un solo agente (como un TSP), con

los parámetros indicados a continuación: 10 iteraciones, 10 hormigas artificiales, $\rho = 0.9$, $\varepsilon = 0.6$, $\alpha = 1.5$, $\beta = 2.5$ y $q_0 = 0.98$, obteniendo los resultados señalados en la Tabla 5.4.

El promedio de la brecha al mejor resultado conocido para todas las instancias fue de 14.75 % y el tiempo que le tomó al algoritmo resolver una distancia de 932 nodos fue de casi dos horas, pero, con tiempos menores a doce minutos para las instancias de 700 nodos o menos, como se muestra en la Figura 5.5.

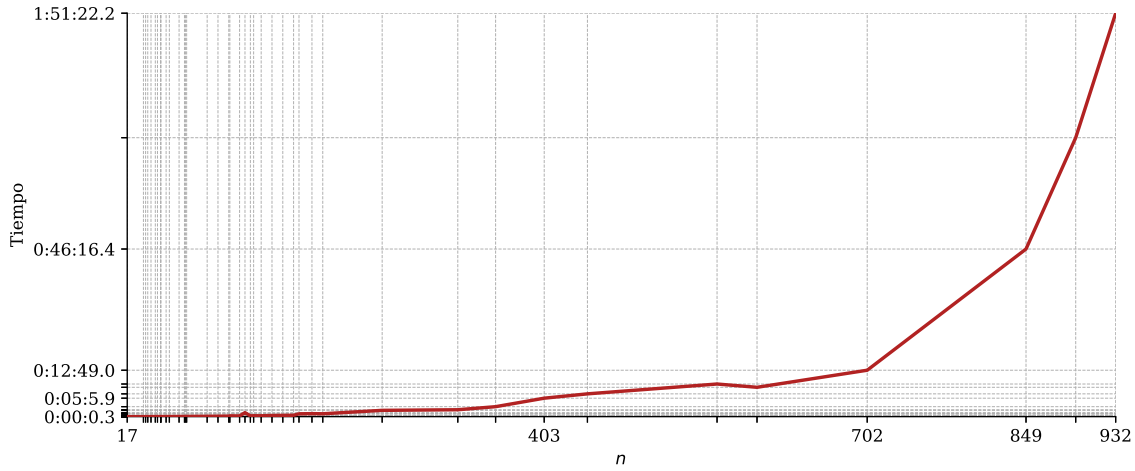


Figura 5.5: Escalada de tiempo para el ruteo
Fuente: Elaboración propia

Para el caso de las instancias generadas para los estados de la República Mexicana no se cuenta con una referencia del óptimo, pero es posible visualizar el comportamiento del algoritmo, como se muestra en la Figura 5.6a que ilustra la ruta generada para la instancia ags11, mientras que la Figura 5.6b expone la ruta generada para la instancia coah38, con una distancia total de 309.916 y 2, 797.330 kilómetros, respectivamente.

Tabla 5.3: Comparación entre literatura y experimentación

	Experimentación	Literatura	Diferencia
α	1.94	1.5	0.44
β	3.52	2.5	1.02

Fuente: Elaboración propia

Tabla 5.4: Resultados del ruteo para las instancias consideradas

Instancia	Mínimo alcanzado	Segundos	Tiempo (h:m:s)	Brecha al óptimo	Instancia	Mínimo alcanzado	Segundos	Tiempo (h:m:s)	Brecha al óptimo
br17	39	0.31	00:00:00	0 %	dc126	125193	66.76	00:01:07	1.59 %
atex3	2960	0.76	00:00:01	0.27 %	ftv130	2892	13.11	00:00:13	25.36 %
ftv33	1464	0.91	00:00:01	13.84 %	dc134	5678	15.57	00:00:16	1.18 %
ftv35	1573	1.15	00:00:01	6.79 %	ftv140	2928	14.69	00:00:15	20.99 %
ftv38	1600	1.27	00:00:01	4.58 %	ftv150	3302	18.77	00:00:19	26.46 %
p43	5631	2.47	00:00:02	0.2 %	ftv160	3532	21.25	00:00:21	31.64 %
ftv44	1776	1.65	00:00:02	10.11 %	ftv170	3564	23.06	00:00:23	29.36 %
atex4	3475	1.68	00:00:02	7.99 %	dc176	8797	46.95	00:00:47	2.45 %
ftv47	2079	1.96	00:00:02	17.06 %	dc188	10362	50.16	00:00:50	1.34 %
ry48p	15306	1.95	00:00:02	6.13 %	code198	4546	47.73	00:00:48	0.11 %
ft53	8424	2.36	00:00:02	22 %	code253	106957	104.41	00:01:44	0 %
ftv55	1874	2.03	00:00:02	16.54 %	rbg323	1699	114.37	00:01:54	28.13 %
ftv64	2233	3.40	00:00:03	21.42 %	rbg358	1688	165.35	00:02:45	45.14 %
ft70	41859	3.66	00:00:04	7.99 %	rbg403	3405	305.88	00:05:06	38.13 %
ftv70	2301	3.78	00:00:04	18 %	rbg443	3747	377.64	00:06:18	37.76 %
atex5	5850	4.45	00:00:04	11.03 %	dc563	26636	540.69	00:09:01	2.64 %
ftv90	1934	6.42	00:00:06	22.48 %	atex8	48554	486.26	00:08:06	21.44 %
ftv100	2297	7.55	00:00:08	28.47 %	big702	103939	768.99	00:12:49	31.43 %
ftv110	2561	9.33	00:00:09	30.8 %	dc849	37762	2776.40	00:46:16	0.76 %
dc112	11213	13.64	00:00:14	0.94 %	dc895	110479	4620.18	01:17:00	2.58 %
ftv120	2664	11.52	00:00:12	22.99 %	dc932	486604	6682.16	01:51:22	1.4 %

Fuente: Elaboración propia con datos de Reinelt (1994); Ismkhan (2017); Dorigo y Stützle (2004)

5.3 Algoritmo completo

Una vez validadas las etapas de partición y ruteo, es tarea de la presente investigación reportar los resultados del algoritmo completo para las instancias del TSPLIB que se tomaron como referencia para validar la herramienta. Se aplicó el algoritmo de partición y de ruteo con los mismos parámetros para el ACO y con 5 agentes para todas las instancias, los resultados se muestran en la Tabla 5.5.

Tabla 5.5: Resultados del algoritmo completo para las instancias consideradas

Nombre	n	Mínimo	Tiempo (s)	PMP (h:m:s)	Tiempo (s)	ACO (h:m:s)	Tiempo (s)	Total (h:m:s)
atex3	32	5896	5.59	0:00:06	0.42	0:00:00	6	0:00:06
ftv33	34	1790	1.2	0:00:01	0.46	0:00:00	1.66	0:00:02
ftv35	36	2001	1.33	0:00:01	0.37	0:00:00	1.69	0:00:02
ftv38	39	2040	1.2	0:00:01	0.59	0:00:01	1.8	0:00:02
p43	43	6096	0.78	0:00:01	0.5	0:00:01	1.28	0:00:01
ftv44	45	2280	0.87	0:00:01	0.71	0:00:01	1.59	0:00:02
atex4	48	6026	1.06	0:00:01	0.48	0:00:00	1.54	0:00:02
ftv47	48	2434	1.12	0:00:01	0.71	0:00:01	1.84	0:00:02
ry48p	48	20245	1.09	0:00:01	0.63	0:00:01	1.72	0:00:02
ft53	53	11291	1.54	0:00:02	0.67	0:00:01	2.21	0:00:02
ftv55	56	2304	1.4	0:00:01	0.62	0:00:01	2.02	0:00:02
ftv64	65	2544	1.84	0:00:02	0.89	0:00:01	2.73	0:00:03
ft70	70	48943	12.68	0:00:13	1.01	0:00:01	13.69	0:00:14
ftv70	71	2669	1.86	0:00:02	0.96	0:00:01	2.81	0:00:03
atex5	72	9968	13.78	0:00:14	0.88	0:00:01	14.66	0:00:15
ftv90	91	2437	15.37	0:00:15	1.39	0:00:01	16.75	0:00:17
ftv100	101	2751	25.57	0:00:26	1.7	0:00:02	27.27	0:00:27
ftv110	111	3123	19.77	0:00:20	2.09	0:00:02	21.86	0:00:22
dc112	112	12050	6.69	0:00:07	2.54	0:00:03	9.23	0:00:09
ftv120	121	3001	26.65	0:00:27	2.34	0:00:02	28.99	0:00:29
dc126	126	132601	10.56	0:00:11	5.04	0:00:05	15.6	0:00:16
ftv130	131	3443	35.6	0:00:36	2.53	0:00:03	38.13	0:00:38
dc134	134	6068	14.7	0:00:15	3.12	0:00:03	17.81	0:00:18
ftv140	141	3374	97	0:01:37	3.31	0:00:03	100.31	0:01:40
ftv150	151	3758	91.48	0:01:31	3.38	0:00:03	94.86	0:01:35
ftv160	161	3698	354.21	0:05:54	3.84	0:00:04	358.05	0:05:58
ftv170	171	4192	609.12	0:10:09	4.46	0:00:04	613.58	0:10:13
dc176	176	9401	29.02	0:00:29	6.79	0:00:07	35.81	0:00:36
dc188	188	11329	37.08	0:00:37	6.11	0:00:06	43.19	0:00:43

Continúa en la siguiente página...

code198	198	4000015445	96.38	0:01:36	5.05	0:00:05	101.43	0:01:41
code253	253	4000233423	132.65	0:02:13	8.91	0:00:09	141.56	0:02:22
rbg323	323	4641	1280.56	0:21:21	18.86	0:00:19	1299.42	0:21:39
rbg358	358	5577	1414.17	0:23:34	21.21	0:00:21	1435.39	0:23:55
rbg403	403	6469	1648.6	0:27:29	24.28	0:00:24	1672.88	0:27:53
rbg443	443	7383	1959.46	0:32:39	33.28	0:00:33	1992.74	0:33:13
dc563	563	27422	2528.95	0:42:09	72.37	0:01:12	2601.32	0:43:21
atex8	600	58874	4521.82	1:15:22	53.66	0:00:54	4575.48	1:16:15
big702	702	132719	7609.9	2:06:50	76.24	0:01:16	7686.14	2:08:06
dc849	849	38348	10992.15	3:03:12	124.52	0:02:05	11116.67	3:05:17
dc895	895	113325	16445.07	4:34:05	329.01	0:05:29	16774.07	4:39:34
dc932	932	494105	19198.42	5:19:58	323.59	0:05:24	19522.01	5:25:22

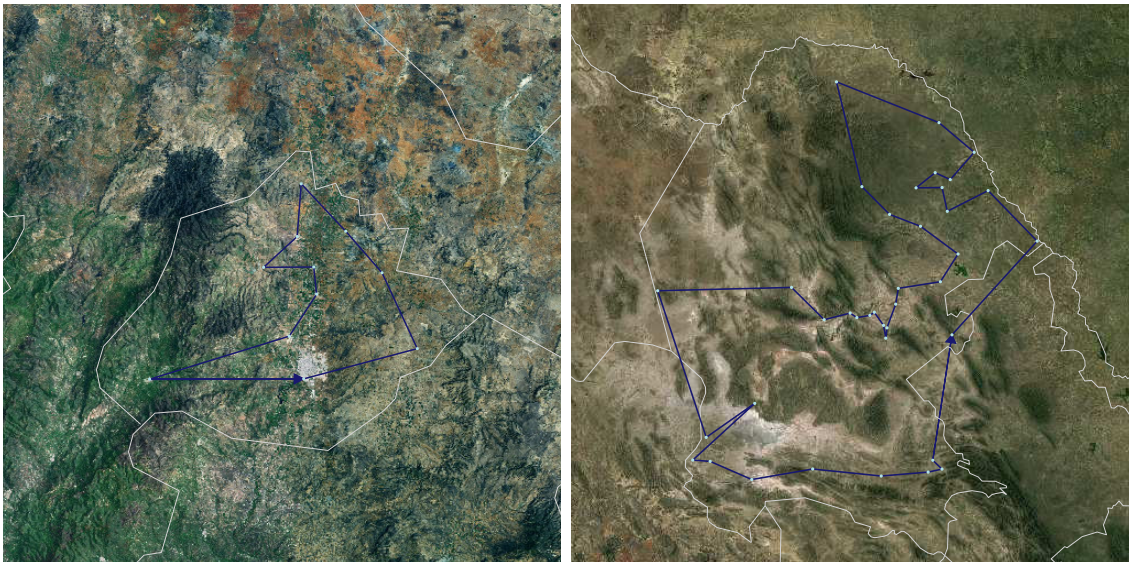
Fuente: Elaboración propia con datos de Reinelt (1994); Ismkhan (2017); Dorigo y Stützle (2004)

El comportamiento con respecto al tiempo se muestra en la Figura 5.7. Aunque el tiempo comienza a escalar de una manera considerable después de los quinientos nodos, el algoritmo muestra tiempos aceptables (menores a 40 minutos) para las instancias menores al volumen mencionado.

Como se observa en la Figura 5.8, con los parámetros usados, la mayor parte del tiempo fue para la realización de la partición de los nodos, y una mínima parte para el cálculo de la ruta, esto es debido a que la primera fase busca llegar al óptimo, aunque, podría hacerse uso de otro método de solución para minimizar el tiempo de ejecución a cambio de una menor calidad en los resultados.

El algoritmo es capaz de, estableciendo un nodo de inicio, calcular una ruta para cada agente, haciendo posible la mejora de una parte del resultado si no se considera que el mencionado sea satisfactorio en el primer cálculo. Un ejemplo del comportamiento del algoritmo en la instancia edomex125 se muestra en la Figura 5.9.

Es importante informar que las rutas generadas se basan en las distancias carreteras reales y que los aparentes cruces, responden a la inclusión del elemento topográfico, como lo es el caso de las rutas mostradas en la Figura 5.10.



(a) (b)

Figura 5.6: Rutas generadas para las instancias ags11 y coah38
Fuente: Elaboración propia con datos de (INEGI, 2017)

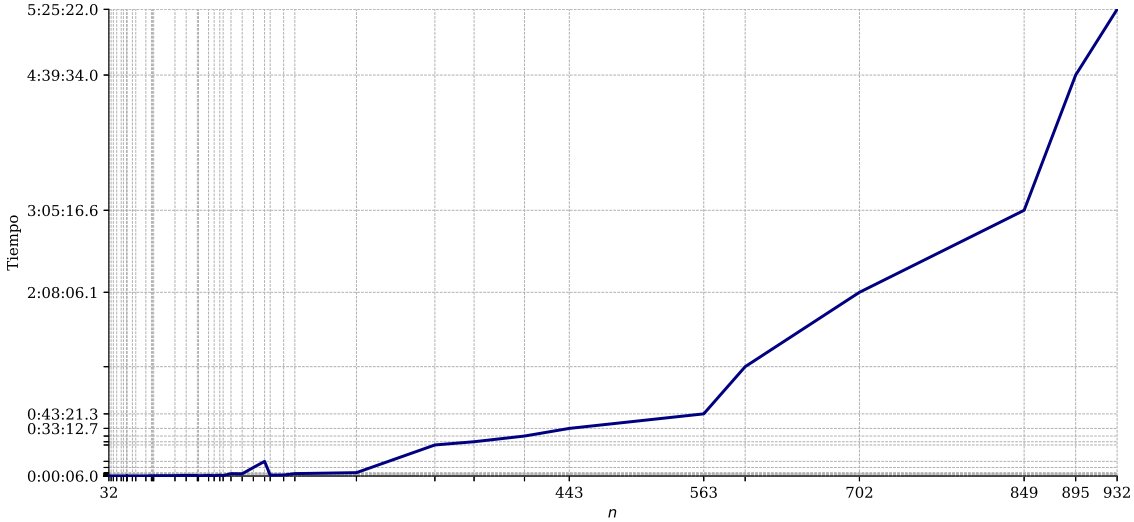


Figura 5.7: Escalada de tiempo para el algoritmo completo
Fuente: Elaboración propia

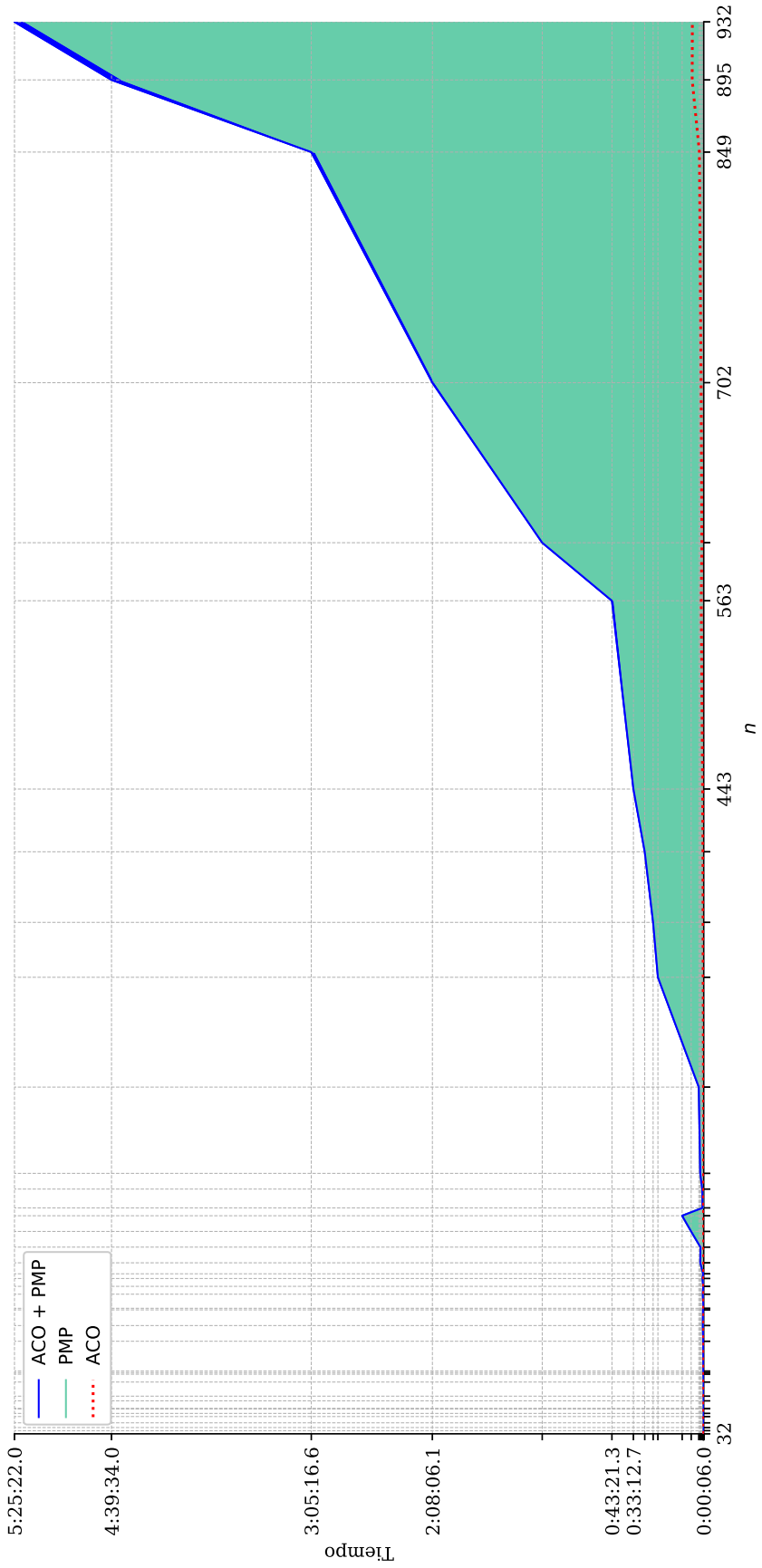


Figura 5.8: Composición del tiempo para el algoritmo completo
Fuente: Elaboración propia

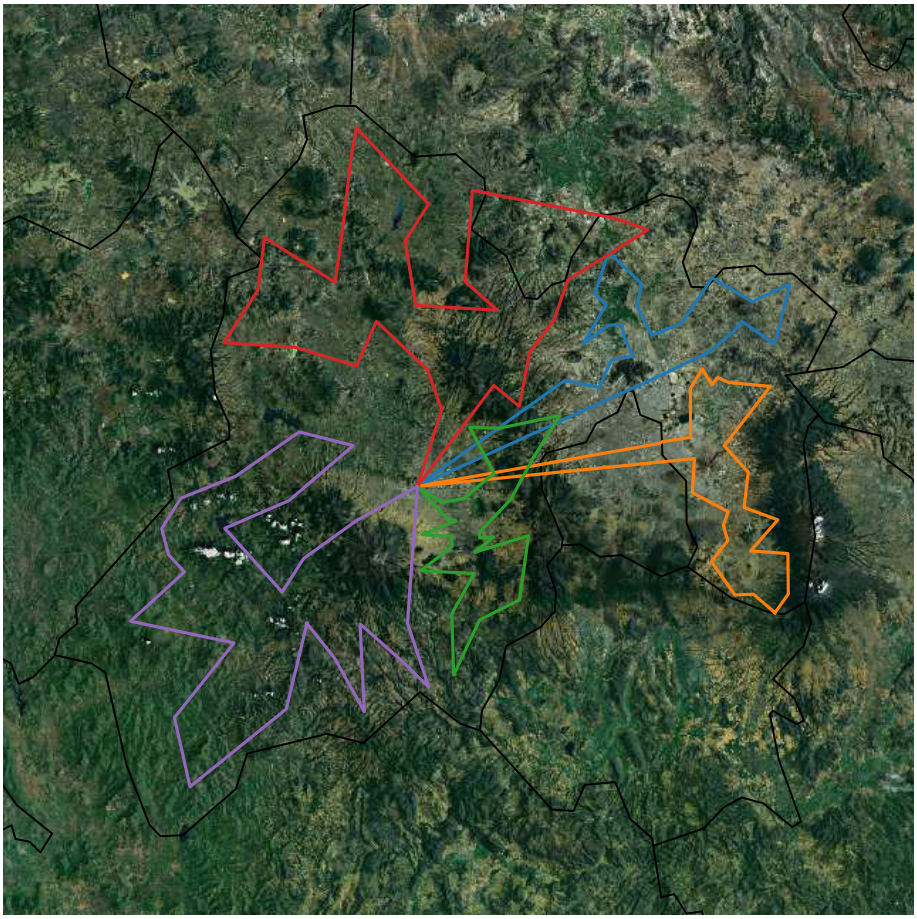


Figura 5.9: Ruta generada para la instancia edomex125
Fuente: Elaboración propia

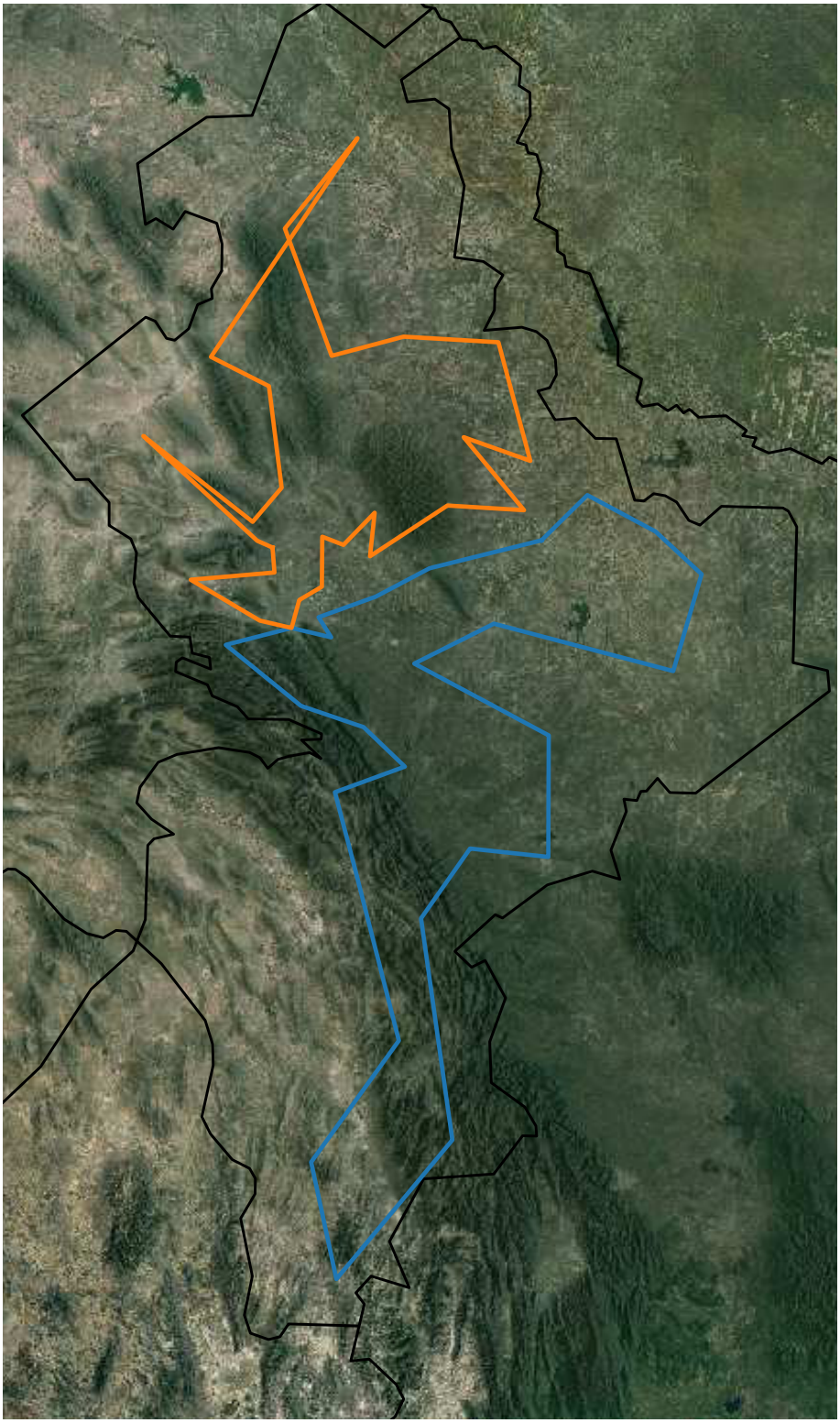


Figura 5.10: Ruta generada para la instancia nl51
Fuente: Elaboración propia

Por último, se compararon algunos de los resultados obtenidos por el algoritmo contra los resultados del modelo del m-TSP resuelto a través de la sintaxis de GAMS enviándolo al servidor NEOS, obteniendo un tiempo de respuesta constante y aceptable pero, la variación porcentual del mínimo por cada método de solución muestra un relación negativa después de las instancias mayores a 65 nodos, como se muestra en la Figura 5.11.

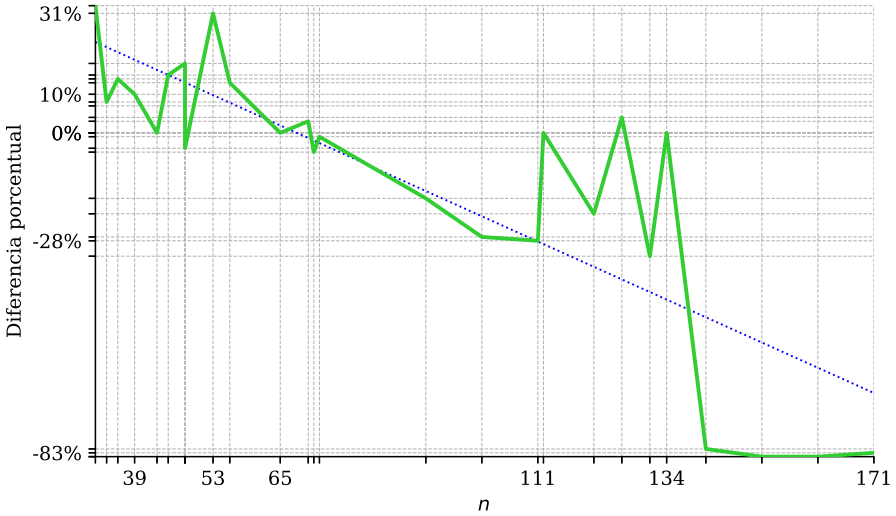


Figura 5.11: Diferencia entre resultados algoritmo exacto y aproximado
Fuente: Elaboración propia

Esto quiere decir que después del volumen de nodos mencionado anteriormente (para cinco agentes) el algoritmo propuesto tiende a ofrecer mejores resultados sin la necesidad de adquirir una licencia de un solver comercial.

Capítulo 6

Caso de estudio

El proyecto por desarrollar estará encauzado a la red de distribución de la promoción de un nuevo producto. La empresa con la que se colaboró para el desarrollo del proyecto presta el servicio de mercadotecnia a una firma refresquera significativa en la república mexicana, misma que introducirá un nuevo SKU (*Stock Keeping Unit*) al mercado, debido a esto, se está planteando la distribución para dar a conocer producto mediante la entrega de muestras gratuitas, se proyectó entregar las muestras en áreas estratégicas de la república elegidas por los encargados de la implementación del proyecto. El propósito general de la estrategia del reparto de muestras es incentivar el consumo al menor costo posible. Las muestras no tendrán precio para los consumidores, pero se contemplará como inversión bajo el presupuesto de mercadotecnia.

Se contemplan 22 ciudades y cinco vehículos; el listado, así como los puntos de interés que contiene cada una, se muestra en la Tabla E.1 del Apéndice E. Su ubicación geográfica se expone en la Figura 6.1. Cada ciudad tiene puntos de interés preestablecidos por la empresa, mismos que no serán analizados a detalle ahora mismo como variables del problema, sino que se considerarán agrupados por ciudades, considerando el centro de cada ciudad como un punto. Los vehículos serán provistos en cada ciudad de las muestras, y no las transportarán entre cada punto de interés, sino que las recibirán y fungirán como una ubicación fija para el reparto a las per-

sonas que transiten esa ubicación, es por esto que la capacidad no está contemplada como restricción.

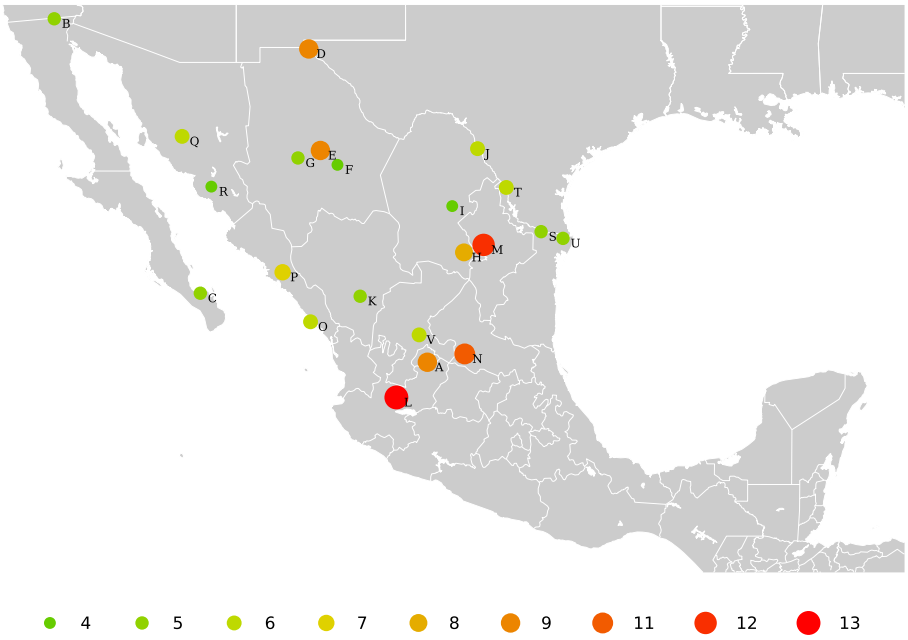


Figura 6.1: Ubicación de las ciudades contempladas

El lanzamiento de producto se contempló a nivel ciudad, es por este motivo que una de las características del proyecto es cubrir la mayor cantidad de zonas al mismo tiempo con la flota disponible.

Todos los vehículos estarán simultáneamente en distintas ciudades para abarcar la cantidad de muestras asignadas a cada ciudad. Los vehículos serán proporcionados por la compañía productora, pero tendrán el diseño de puntos de reparto de muestras con refrigeradores comerciales y debido a que reflejan la marca de la empresa refresquera no serán detallados en el presente documento.

Se desconoce el punto de partida de los vehículos, siendo este punto sujeto a cambio por los encargados en la empresa refresquera. Los cinco vehículos deben de cubrir las cantidades de reparto por ciudad.

6.1 Cálculo de la ruta

Como ya se ha visto anteriormente los 150 puntos a visitar se encuentran dentro de 22 ciudades mismos que fueron asignados como se observa en la Figura 6.2.

Figura 6.2: Partición de las 22 ciudades para 5 vehículos

La ciudad de partida para todos los vehículos será Zapopan, misma que se incluirá en la ruta de Culiacán, Durango y Mazatlán; el punto de partida para todos los vehículos será la Basílica de Zapopan. Efectuando ya la partición las distancia total de las rutas generadas por el algoritmo propuesto se visualizan en la Figura 6.3.

El tiempo total, considerando el de la partición, para esta solución fue de 450.37 segundos (o 7 minutos con 30 segundos), utilizando los parámetros: $\alpha = 1$, $\beta = 2$, 500 iteraciones, 10 hormigas, $\rho = 0.9$, $\varepsilon = 0.1$ y $q_0 = 0.98$. La longitud total recorrida por los vehículos es de $14,291.039$ kilómetros.

Siguiendo la tónica anterior, ¿cómo es posible hacer la comparación con lo que

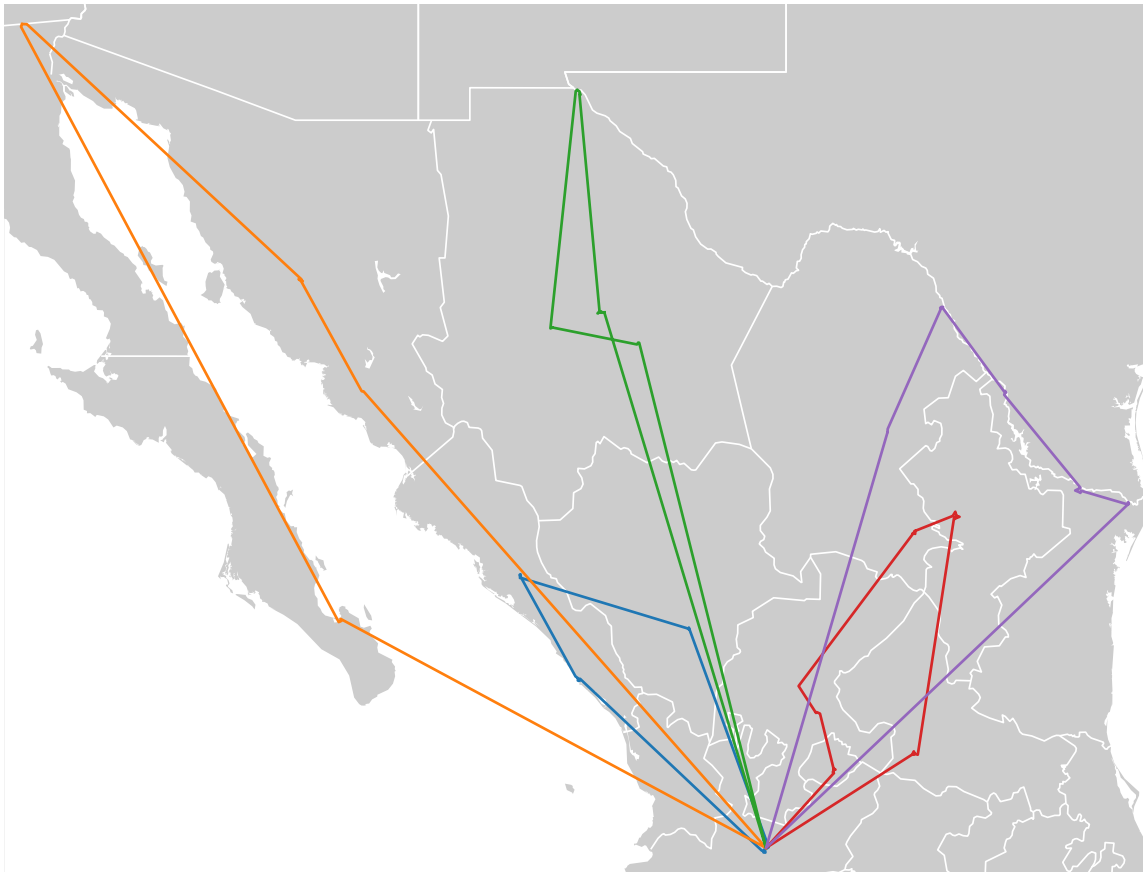


Figura 6.3: Ruteo para 5 vehículos con previa partición por ciudad

se hubiera realizado sin un método estructurado?, la respuesta a esta pregunta en imprecisa, sin embargo es posible tomar la asignación por zonas que proporcionó la empresa de mercadotecnia y aplicar un heurístico de vecino más cercano que es lo que usualmente se emplea para generar una ruta «sobre la marcha», tal ruteo se efectuó, y es mostrado en la Figura 6.4, resultando en una distancia de $14,604.401$ kilómetros, razonablemente mayor a la propuesta con el método desarrollado en capítulos anteriores.

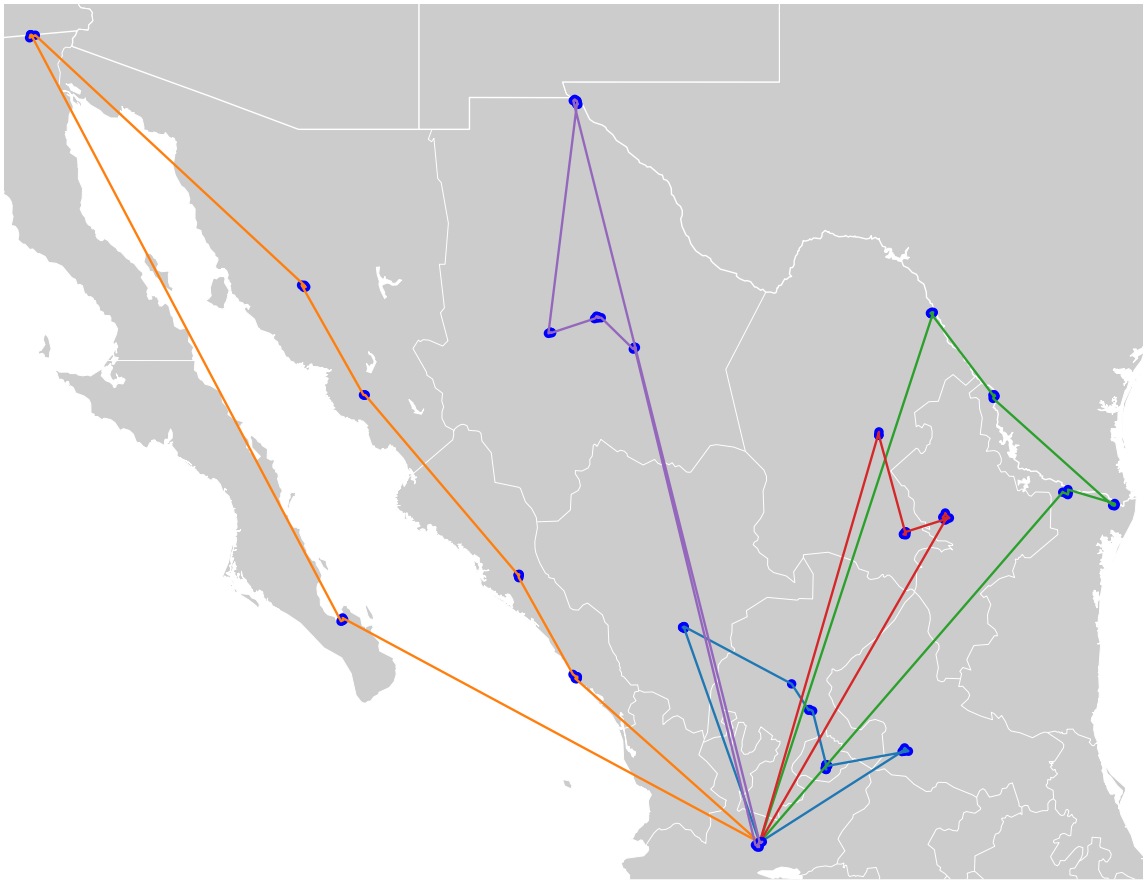


Figura 6.4: Ruteo para 5 vehículos con previa asignación y vecino más cercano

Ahora bien, considerando una partición estrictamente equitativa de los nodos genera una respuesta con similar rapidez pero con distancias mayormente desproporcionadas. La Figura 6.5 expone este comportamiento que tuvo una distancia total de $15,236.705$ kilómetros que fue resuelta en un tiempo total de 782.36 segundos (o 13 minutos con 2 segundos).

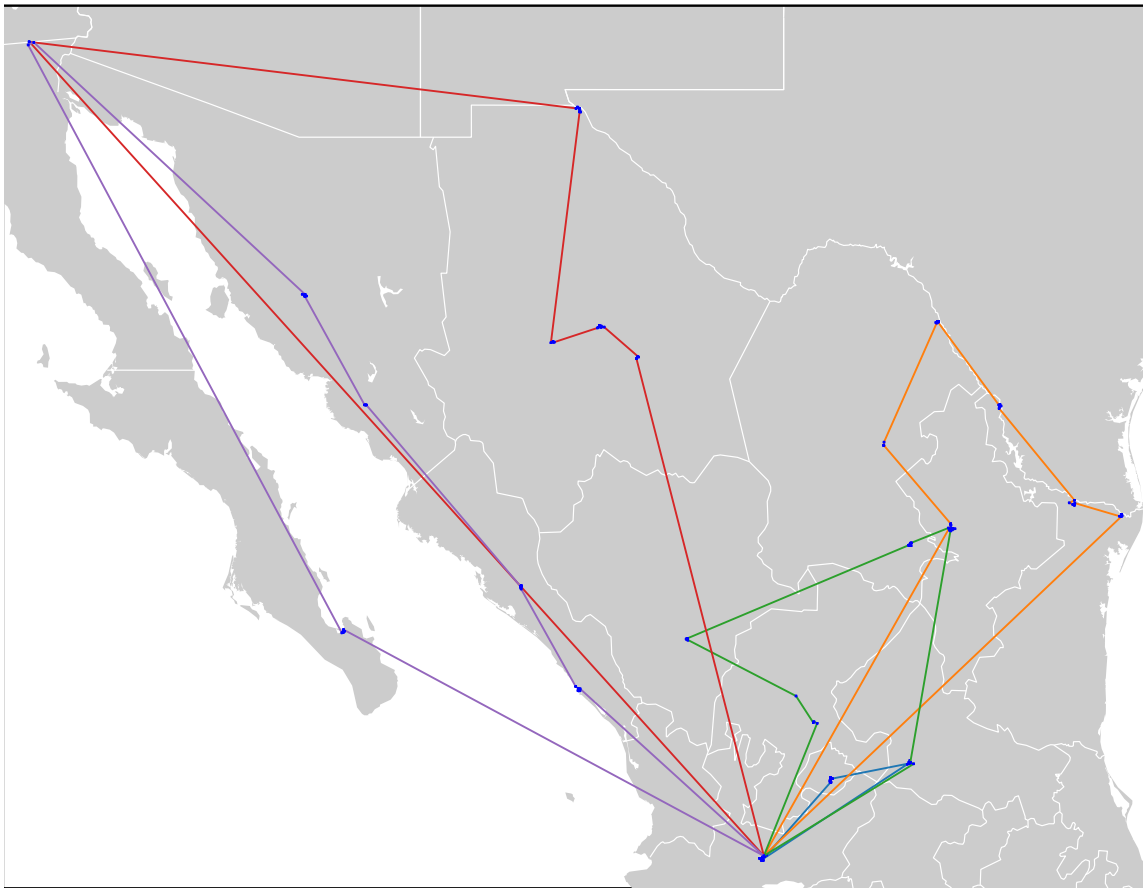


Figura 6.5: Ruteo para 5 vehículos con reparto equitativo de nodos

Capítulo 7

Conclusiones

Como ya se avizoraba al principio de la investigación, los problemas derivados de la distribución, y más específicamente de transporte, pueden representar tanto una ventaja competitiva como un detrimento que cause ineficiencias en el uso de los recursos disponibles por indeterminada empresa, generando a su vez pérdidas de recursos (por ejemplo financieros o de tiempo) que podrían haber sido empleados de una manera eficiente.

Para el caso de estudio que se abordó previamente, se consideró como único parámetro la distancia, que en cierto modo, puede impactar positivamente otras variables como el tiempo, el consumo de combustible o el manejo administrativo de los itinerarios de viaje.

Con respecto a el cálculo de tales itinerarios, la perspectiva matemática es siempre un estándar de comparación, y de uso, para obtener datos fehacientes y exactos; conteniendo modelos clásicos que ofrecen planteamientos probados para una variedad de problemas en una de sus ramas: la investigación de operaciones.

Los modelos clásicos de ruteo, por consiguiente, son ampliamente usados para los problemas de distribución, ya sea en su formulación clásica o con adecuaciones que ayuden a considerar las dimensiones de las aplicaciones especificas a las que se esté atacando. Para la presente investigación, y debido a que se consideró unicamente

la distancia como parámetro, la mejor alternativa fue usar el modelo del m-TSP.

Persiguiendo la mejor solución posible al problema del caso de estudio, se buscó un método para la solución del modelo anteriormente mencionado que ha probado presentar un reto significativo bajo diferentes circunstancias debido a su complejidad (por ser un problema **NP**); siendo un aspecto destacable el tiempo de solución. La mejor alternativa para atacar el anterior dependerá del contexto en el que se aplique y de los recursos con los que se cuente al momento de realizar el cálculo (acceso a software comercial, capacidad de cómputo, presupuesto, etc.).

La descomposición del problema en dos partes presenta un beneficio de flexibilidad en su manejo, que radica en la agrupación de los nodos durante la etapa de asignación, no obstante, la asignación previa podría dejar fuera ciertas combinaciones de nodos que conduzcan al óptimo. Aclarado ya el punto anterior, es importante señalar que el método propuesto logra llegar a buenos resultados comparándolos contra la solución del modelo del m-TSP mediante ramificación y corte, estableciendo un límite de tiempo para tal tarea.

La comparación ya mencionada colocó al método propuesto en una buena posición contra el método exacto, en una situación en la que no es necesario realizar la inversión de adquirir un software para la solución de los modelos, y lográndolo bajo el mismo tiempo que se le otorgó a ambos métodos.

Se acepta la hipótesis de minimizar la distancia recorrida mediante el desarrollo de una herramienta matemática para el diseño de una ruta entre puntos de interés para resolver el problema de una empresa de mercadotecnia; como se mostró en la ruta generada para el caso de estudio en la sección 6.1, que bajo la comparación efectuada, logró un mejor resultado al que se pudo dar posiblemente bajo un enfoque heurístico.

Además de la generación de escenarios descrita, se efectuó la validación de la herramienta en la sección 5.3, con tiempos y resultados alentadores para el tipo de problema que se trató.

7.1 Contribuciones

El método propuesto se vale de herramientas sin un costo significativo para la mayoría de las empresas que pudieran replicar el mismo, no obstante, las soluciones arrojadas por el anterior difícilmente alcanzan el óptimo debido a la naturaleza del algoritmo.

La ayuda de las herramientas de uso libre y en servidores (o como se le nombra hoy en día, «la nube»), fueron indispensables para generar la herramienta propuesta, no tan solo facilitando las rutinas empleadas por el algoritmo sino por añadidura, causando que la replicabilidad del mismo represente un costo financiero insignificativo.

Visto que los problemas de ruteo no son tarea fácil, el presente documento aporta una manera más de cómo solucionar dicha problemática con soluciones alcanzables para casi cualquiera que desee probarla, y posiblemente, comparar sus resultados de su método de solución o algoritmo con los del método propuesto, para así encontrar mejores soluciones al sinfín de aplicaciones que estas herramientas tienen en muchas cadenas de suministro: uniendo a través del transporte y el ruteo los eslabones que las componen y, por ende, mejorando las mismas cada vez que se progresa en la eficiencia y eficacia de la distribución y el transporte.

Es prudente comentar el aporte de la presente investigación del uso del PMP para la etapa de partición de los nodos, que no es común para la aproximación asignación primero, ruta segundo estudiada en la literatura.

7.2 Trabajo futuro

Las posibilidades de solución, son tan variadas como los distintos tipos de problemas de ruteo que se presentan a diario en las empresas participes de una (o

varias) cadena(s) de suministro, en este sentido, un posible aspecto de investigación futura podría ser la aplicación del metaheurístico al modelo completo del m-TSP, o a la etapa de partición, con el objetivo de mejorar el tiempo de respuesta total, ya que como se mostró en la sección 5.3, esta etapa toma una parte considerable de este recurso en la ejecución del algoritmo, así como la adecuación del método propuesto para problemas de ruteo de mayor complejidad, que consideren mayores tipos de restricciones en su planteamiento.

Toda la experimentación (más el algoritmo metaheurístico usado) estuvo articulada en lenguaje Python, mismo que es interpretado (no compilado). Es por esto que se considerará su transcripción a un lenguaje compilado que ofrezca una mayor velocidad de ejecución (por ejemplo C++).

La complejidad de solución de los problemas de ruteo conmina un panorama en el que futuros y presentes investigadores seguirán, con voluntad vehemente, buscando la mejor metodología a seguir, que marque el colofón para resolver los primeros, pero, mientras tanto, distintas disciplinas seguirán aportando lo que el presente documento buscó desde un principio: desvelar un ápice más de un problema intratable.

Apéndice A

Lista de instancias

Las instancias consideradas para la etapa de experimentación provenientes de la revisión de literatura se muestran en la tabla A.1.

Tabla A.1: Instancias consideradas para la experimentación

Nombre	n	Fuente	Óptimo	Nombre	n	Fuente	Óptimo
br17	17	Reinelt (1991)	39	dc126	126	Cirasella *et al.* (2001)	123235
atex3	32	Reinelt (1991)	2952	ftv130	131	Reinelt (1991)	2307
ftv33	34	Reinelt (1991)	1286	dc134	134	Cirasella *et al.* (2001)	5612
ftv35	36	Reinelt (1991)	1473	ftv140	141	Reinelt (1991)	2420
ftv38	39	Reinelt (1991)	1530	ftv150	151	Reinelt (1991)	2611
p43	43	Reinelt (1991)	5620	ftv160	161	Reinelt (1991)	2683
ftv44	45	Reinelt (1991)	1613	ftv170	171	Reinelt (1991)	2755
atex4	48	Cirasella *et al.* (2001)	3218	dc176	176	Cirasella *et al.* (2001)	8587
ftv47	48	Reinelt (1991)	1776	dc188	188	Cirasella *et al.* (2001)	10225
ry48p	48	Reinelt (1991)	14422	code198	198	Cirasella *et al.* (2001)	4541
ft53	53	Reinelt (1991)	6905	code253	253	Cirasella *et al.* (2001)	106957
ftv55	56	Reinelt (1991)	1608	rbg323	323	Reinelt (1991)	1326
ftv64	65	Reinelt (1991)	1839	rbg358	358	Reinelt (1991)	1163
ft70	70	Reinelt (1991)	38763	rbg403	403	Reinelt (1991)	2465
ftv70	71	Reinelt (1991)	1950	rbg443	443	Reinelt (1991)	2720
atex5	72	Cirasella *et al.* (2001)	5269	dc563	563	Cirasella *et al.* (2001)	25951
ftv90	91	Reinelt (1991)	1579	atex8	600	Cirasella *et al.* (2001)	39982
ftv100	101	Reinelt (1991)	1788	big702	702	Cirasella *et al.* (2001)	79081
ftv110	111	Reinelt (1991)	1958	dc849	849	Cirasella *et al.* (2001)	37476
dc112	112	Cirasella *et al.* (2001)	11109	dc895	895	Cirasella *et al.* (2001)	107699
ftv120	121	Reinelt (1991)	2166	dc932	932	Cirasella *et al.* (2001)	479900

Fuente: Elaboración propia con datos de Aguayo *et al.* (2017)

Apéndice B

Distancias Google API

El código fuente para obtener las distancias terrestres a través de *Google API services* se muestra en el código B.1.

Se requiere solicitar una (o varias) clave(s) API en el portal de Google, así como conocer los límites de consulta gratuitos aplicables a la fecha de ejecución del programa. También es necesario un archivo en formato .csv o .xls con los datos del nombre de la ubicación que se consultará (solamente como referencia) y su latitud y longitud.

El programa mostrado generará un archivo de texto con el mismo formato de las instancias asimétricas del TSPLIB (Reinelt, 1994).

```
1  import csv
2  import math
3  import requests
4  import pandas as pd
5  from bs4 import BeautifulSoup
6
7  def url_distancematrix(inicio, destino, apikey):
8      dest_s = len(destino)
9      url = "https://maps.googleapis.com/maps/api/distancematrix/xml?
       units=metric&origins={inicio}&destinations={destino}&key={apikey
       }"
10     solic = url.format(inicio=inicio,destino=destino,apikey=apikey)
11     return solic
12
13 archivo = "archivo_con_latitudes_y_longitudes.csv"
```

```
nombre = "nombre_de_la_instancia_a_generar"

try:
    xl = pd.ExcelFile(archivo)
    df = xl.parse(xl.sheet_names[0])
except:
    df = pd.read_csv(archivo, sep=',')

nombres, lats, lons = [i for i in df['
   nombre_columna_con_datos_lugares']], [float(i) for i in df['
   nombre_columna_con_latitudes_lugares']], [float(i) for i in df['
   nombre_columna_con_longitudes_lugares']]
claves = ["clave_1", "clave_2", "clave_3", "clave_n"]#las claves
   deben tener 39 caracteres

d_ij = []
for md in range(len(lats)):
    d_ij.append([])

log = set()
clave_turno = 0

for fila in range(n):
    for colu in range(n):
        if fila == colu:
            d_ij[fila].append("0")
        else:
            try:
                limite_consultas = True
                while limite_consultas:
                    apikey = claves[clave_turno]
                    inicio = str(lats[fila])+","+str(lons[fila])
                    destino = str(lats[colu])+","+str(lons[colu])
                    url = url_distancematrix(inicio, destino,
   apikey)
                    html_gdm = requests.get(url)
                    sopa = BeautifulSoup(html_gdm.content, "html.
   parser")
                    status_lec = sopa.findAll("status")
                    status = status_lec[0].getText()
                    if status == "OVER_QUERY_LIMIT":
                        clave_turno += 1
                        break
                    elif status == "OK":
                        limite_consultas = False
                        break
                    elif status == "OVER_QUERY_LIMIT" and
   clave_turno > len(claves)+1:
                        print("Las claves ya no tienen cosultas
   disponibles")
                        print(" Esperar ? S/N")
                        r_usuario = input()
                        if r_usuario == "s" or r_usuario == "S":
                            print("Cuano quieras continuar presiona
```

```
    'Enter'")
60                              input()
61                              break
62                          else:
63                              print("Cierra la ventana, de lo
    contrario 'Enter' para reanudar.")
64                              input()
65                              break
66                  lectura = sopa.findAll("value")
67                  valor = lectura[1].getText()
68                  if len(valor) == 0:
69                      d_ij[fila].append(str(status))
70                      pos = str(fila) + " " + str(colu)
71                      with open("fails\\{}.txt".format(pos), "w") as
    txt:
72                          txt.write(str(sopa))
73                      log += " " + str(fila) + str(colu) + " "
74                  else:
75                      d_ij[fila].append(valor)
76              except:
77                  d_ij[fila].append("  !")
78                  pos = str(fila) + " " + str(colu)
79                  try:
80                      with open("fails\\{}.txt".format(pos), "w") as
    txt:
81                          txt.write("'Ultimo resgistro:\n"+str(sopa))
82                  except:
83                      with open("fails\\{}.txt".format(pos), "w") as
    txt:
84                          txt.write("No se recibi  html revisar
    conexi n")
85                  log.add((fila,colu))
86
87 maximos = list(max(len(str(x)) for x in line) for line in zip(*d_ij
    ))
88 stringMayor = max(maximos)
89 anchoDeColumnaTxt = stringMayor + 2
90
91 d_ij_txt = ""
92 for Colum in range(n):
93      for Reng in range(n):
94           d_ij_txt += (''.join((str(d_ij[Colum][Reng])).ljust(
    anchoDeColumnaTxt)))
95      d_ij_txt += ('\n')
96
97 formato_instancia = """
98 NAME: {nombre}
99 TYPE: ATSP
100 COMMENT: Generado con distancias de google maps
101 DIMENSION:  {dimension}
102 EDGE_WEIGHT_TYPE: EXPLICIT
103 EDGE_WEIGHT_FORMAT: FULL_MATRIX
104 EDGE_WEIGHT_SECTION
105 {d_ij}
```

```
106 EOF
107 """
108
109 instancia = formato_instancia.format(nombre=nombre,dimension=n,d_ij
       =d_ij_txt)
110
111 salida = nombre + ".txt"
112 with open(salida, "w") as text_file:
113     text_file.write(instancia)
114
115 salida_log = "log_" + nombre + ".txt"
116 with open(salida_log, "w") as text_file:
117     text_file.write(str(log))
```

Código B.1: Código en lenguaje Python

Apéndice C

Código fuente PMP

Se muestra a continuación el código fuente para la solución del modelo PMP modificado, que se mostró en el apartado 4.3, para su solución bajo la sintaxis de GAMS (*General Algebraic Modeling System*), haciendo uso del *solver* CPLEX.

```
Sets
I /1*{nodos}/;
alias (i,j);
Table
c(i,j)
{matriz}
;
c(i,j) = max(c(i,j),c(j,i));
Parameters
p /{agentes}/
d(j) /{dist_j}/
Q;
Q=ceil(card(I)/p);
Binary variables
x(i,j)
y(j);
Free Variables
z;
Equations
Obj
Asig(i)
Medianas
Activacion(j)
Balance(j);
Obj.. z =e= Sum[(i,j),c(i,j)*x(i,j)]+Sum[(j),d(j)*y(j)];
Asig(i).. Sum[j, x(i,j)] =e= 1;
Medianas.. Sum[j,y(j)] =e= p;
Activacion(j).. sum(i, x(i,j)) =l= Q*y(j);
```

```
29 Balance(j).. Sum[i,x(i,j)] =l= Q;
30 Model MODELO /All/;
31 Option OptCA=0, OptCR=0;
32 Solve MODELO using MIP minimizing z;
```

Código C.1: Código en GAMS para la solución del PMP

Apéndice D

Código fuente ACO

En el código D.1 se expondrá la programación del algoritmo metaheurístico ACO, mostrado en el apartado 4.4, escrito en lenguaje Python 3.6.3.

```
import sys
import math
import random
import numpy as np
from scipy import sparse

def esaco(iteraciones,hormigas,q_cero,epsilon,alfa,beta,ro,
    matriz_instancia):
     tamanio_instancia = len(matriz_instancia)
     forma = (tamanio_instancia,tamanio_instancia)
     tau = sparse.lil_matrix(forma)
     eta = np.ones(forma)
     for fila in range(tamanio_instancia):
         for val_heur in range(tamanio_instancia):
             try:
                 eta[fila][val_heur] = 1 / matriz_instancia[fila][
    val_heur]
             except:
                 eta[fila][val_heur] = 1
     eta = tuple(tuple(i) for i in eta)

     def dosopt(secuencia,mejor_distancia,candidatos):
         combinaciones = list()
         for i in range(tamanio_instancia):
             for j in range(len(candidatos[i])):
                 combinaciones.append((i,candidatos[i][j]))
         combinaciones = tuple(tuple(i) for i in combinaciones)
         iteraciones = 0
         usados = set()
         reinicio = True
```

```
29          while reinicio:
30              for combinacion in combinaciones:
31                  intercambio = list(secuencia)
32                  pos_n = intercambio.index(combinacion[0])
33                  pos_i = intercambio.index(combinacion[1])
34                  if pos_n == pos_i:
35                      continue
36                  else:
37                      if pos_n > pos_i:
38                          intercambio[pos_i:pos_n+1] = intercambio[
    pos_i:pos_n+1][::-1]
39                      else:
40                          intercambio[pos_n:pos_i+1] = intercambio[
    pos_n:pos_i+1][::-1]
41                      distancia_uno = distancia_secuencia(intercambio
    )
42                      distancia_dos = distancia_secuencia(intercambio
    [::-1])
43                      if distancia_dos < distancia_uno:
44                          intercambio = intercambio[::-1]
45                          distancia = distancia_dos
46                      else:
47                          distancia = distancia_uno
48                      if distancia < mejor_distancia:
49                          mejor_distancia = distancia
50                          secuencia = intercambio
51                          usados = set()
52                          break
53                      elif combinacion in usados:
54                          reinicio = False
55                          break
56                      elif combinacion not in usados and distancia >=
     mejor_distancia:
57                          usados.add(combinacion)
58                          iteraciones += 1
59                          continue
60          resultados = (secuencia,iteraciones,mejor_distancia)
61          return resultados
62
63      def distancia_secuencia(secuencia):
64          ultimo_arco = matriz_instancia[secuencia[-1]][secuencia[0]]
65          distancia = 0
66          for nodo in range(tamanio_instancia-1):
67              distancia += matriz_instancia[secuencia[nodo]][
    secuencia[nodo+1]]
68          distancia += ultimo_arco
69          return distancia
70
71      def proporcional_pseudoaleatoria(nodos_considerados,nodo_actual
    ):
72          q = random.uniform(0,1)
73          if q <= q_cero:
74              args_S = list()
75              for u in nodos_considerados:
```

```
76                  tau_r_u = tau[nodo_actual,u]
77                  if tau_r_u == 0:
78                      tau_r_u = tau_por_defecto
79                  eta_r_u = eta[nodo_actual][u]
80                  arg = tau_r_u * math.pow(eta_r_u,beta)
81                  args_S.append((u,arg))
82              pp_S = max(args_S,key=lambda item:item[1])
83              movimiento = pp_S[0]
84          else:
85              numeradores = list()
86              for s in nodos_considerados:
87                  tau_r_s = tau[nodo_actual,s]
88                  if tau_r_s == 0:
89                      tau_r_s = tau_por_defecto
90                  eta_r_s = eta[nodo_actual][s]
91                  numerador = math.pow(tau_r_s,alfa) * math.pow(
    eta_r_s,beta)
92                  numeradores.append((s,numerador))
93              sigma_u = sum(tuple(i[1] for i in numeradores))
94              p_rs = list()
95              for numerador in numeradores:
96                  p_k_rs = numerador[1] / sigma_u
97                  p_rs.append((numerador[0], p_k_rs))
98              movimiento = np.random.choice(tuple(i[0] for i in p_rs)
    , 1, p=tuple(i[1] for i in p_rs))[0]
99          return movimiento
100
101     def siguiente_movimiento(nodos_por_visitar,nodo_actual,
    candidatos):
102         try:
103             nodos_considerados = tuple(i for i in nodos_por_visitar
     if i in candidatos)
104             movimiento = proporcional_pseudoaleatoria(
    nodos_considerados,nodo_actual)
105         except:
106             try:
107                 valores = tau[nodo_actual].nonzero()[1:]
108                 if len(valores[0]) == 0:
109                     raise Exception('No hay valores de feromonas')
110                 else:
111                     candidatos = tuple(i for i in valores[0])
112                     nodos_considerados = tuple(i for i in
    nodos_por_visitar if i in candidatos)
113                     movimiento = proporcional_pseudoaleatoria(
    nodos_considerados,nodo_actual)
114             except:
115                 try:
116                     movimiento = proporcional_pseudoaleatoria(
    nodos_por_visitar,nodo_actual)
117                 except:
118                     raise Exception('No hay nodos para el siguiente
     movimiento')
119         return movimiento
120
```

```
121     def inicializar_candidatos():
122         candidatos = list()
123         ind_fila = 0
124         for lista_candidatos in matriz_instancia:
125             cercanos = sorted(lista_candidatos)
126             indices = []
127             for indic in cercanos:
128                 indice_actual = lista_candidatos.index(indic)
129                 esta = True
130                 if indice_actual not in indices:
131                     indices.append(indice_actual)
132                 else:
133                     while esta:
134                         if indice_actual in indices:
135                             indice_actual = lista_candidatos[
    indice_actual+1:].index(indic)+indice_actual+1
136                         else:
137                             indices.append(indice_actual)
138                             esta = False
139                             break
140             indices.remove(ind_fila)
141             ind_fila += 1
142             candidatos.append(indices)
143         return candidatos
144 
145     NN_inic = np.random.randint(tamanio_instancia)
146     circuito_NN = []
147     circuito_NN.append(NN_inic)
148     NN = inicializar_candidatos()
149     for movimiento_NN in range(tamanio_instancia-1):
150         esta = True
151         indice_evaluado = 0
152         while esta:
153             movimiento = NN[NN_inic][indice_evaluado]
154             if movimiento in circuito_NN:
155                 indice_evaluado += 1
156             else:
157                 circuito_NN.append(movimiento)
158                 NN_inic = movimiento
159                 esta = False
160                 break
161     distancia_NN = distancia_secuencia(circuito_NN)
162     tau_por_defecto = 1 / distancia_NN
163     candidatos = inicializar_candidatos()
164     mejor_global = ([],np.inf)
165     for iteracion in range(iteraciones):
166         visitados = [[np.random.randint(0,tamanio_instancia)] for i
     in range(hormigas)]
167         por_visitar = [list(range(tamanio_instancia)) for i in
    range(hormigas)]
168         distancias = list()
169         for i in range(len(visitados)):
170             por_visitar[i].remove(visitados[i][0])
171         for nodo in range(tamanio_instancia-2):
```

```
172             for hormiga_k in range(hormigas):
173                 nodo_actual = visitados[hormiga_k][-1]
174                 movimiento_k = siguiente_movimiento(por_visitar[
    hormiga_k], nodo_actual, candidatos[nodo_actual])
175                 tau_rs = tau[nodo,movimiento_k]
176                 if tau_rs == 0:
177                     tau_rs = tau_por_defecto
178                 tau[nodo,movimiento_k] = ((1 - epsilon) * tau_rs) +
     (epsilon * tau_por_defecto)
179                 visitados[hormiga_k].append(movimiento_k)
180                 por_visitar[hormiga_k].remove(movimiento_k)
181                 if nodo == tamanio_instancia-3:
182                     visitados[hormiga_k] = visitados[hormiga_k] +
    por_visitar[hormiga_k]
183                     distancia = distancia_secuencia(visitados[
    hormiga_k])
184                     b_local = dosopt(visitados[hormiga_k],distancia
    ,[i[:4] for i in candidatos])
185                     visitados[hormiga_k] = b_local[0]
186                     distancias.append((hormiga_k, b_local[2]))
187         mejor_iteracion = min(distancias,key=lambda item:item[1])
188         mejor_ruta = visitados[mejor_iteracion[0]]
189         if mejor_iteracion[1] < mejor_global[1]:
190             mejor_global = (mejor_ruta, mejor_iteracion[1])
191             delta_tau_rs = 1 / mejor_global[1]
192             ifg = 0
193             for r in mejor_global[0][:-1]:
194                 s = mejor_global[0][ifg+1]
195                 tau_rs = tau[r,s]
196                 if tau_rs == 0:
197                     tau_rs = tau_por_defecto
198                 tau[r,s] = ((1 - ro) * tau_rs) + (ro * delta_tau_rs
    )
199                 ifg += 1
200             if tau[mejor_global[0][-1],mejor_global[0][0]] == 0:
201                 tau[mejor_global[0][-1],mejor_global[0][0]] =
    tau_por_defecto
202             tau[mejor_global[0][-1],mejor_global[0][0]] = ((1 - ro)
     * tau[mejor_global[0][-1],mejor_global[0][0]]) + (ro *
    delta_tau_rs)
203             for num in range(tamanio_instancia):
204                 c_en_t = candidatos[num]
205                 valores = tau[num].nonzero()[1:]
206                 ind_nodos = tuple(i for i in valores[0])
207                 if len(valores[0]) == 0:
208                     print("No se actualizaron valores de candidatos
     con respecto a feromonas")
209                 else:
210                     ifc = tuple(tau[num,i] for i in ind_nodos)
211                     nuevos_c = tuple((i,j) for i,j in zip(ind_nodos
    ,ifc))
212                     mcf = max(nuevos_c,key=lambda item:item[1])
213                     candidatos[num].insert(0,mcf[0])
214                     candidatos[num] = candidatos[num][:-1]
```

```
215                     try:
216                         ocurr = next(i for i, val in zip(range(len(
    candidatos[num])-1, -1, -1), reversed(candidatos[num])) if val
    == mcf[0])
217                         del candidatos[num][ocurr]
218                     except:
219                         print("Candidato ferom, actualizado sin
    duplicado")
220                 indice_sucesor = mejor_global[0].index(num)
221                 if indice_sucesor == tamanio_instancia - 1:
222                     sucesor = mejor_global[0][0]
223                 else:
224                     sucesor = mejor_global[0][mejor_global[0].index
    (num) + 1]
225                 candidatos[num].insert(0,sucesor)
226                 try:
227                     ocurr = next(i for i, val in zip(range(len(
    candidatos[num])-1, -1, -1), reversed(candidatos[num])) if val
    == sucesor)
228                     del candidatos[num][ocurr]
229                 except:
230                     print("Candidato sucesor, actualizado sin
    duplicado")
231                 num += 1
232     return mejor_global
```

Código D.1: Código fuente ACO

Apéndice E

Datos caso de estudio

En este apartado se pormenorizarán los datos citados en el capítulo 6, iniciando con el listado completo de las ciudades (Tabla E.1), indicando estado y ubicación en latitud y longitud, considerando como punto central la cabecera municipal o el palacio de gobierno, debido a que son ubicaciones generalmente cercanas al centro de cada poblado. La visualización de dicha información se encuentra en la Figura 6.1.

Siguiendo con las distancias terrestres en kilómetros lineales desde cada una de las coordenadas mostradas en la tabla anterior.

Tabla E.1: Listado de ciudades

Etiqueta	Estado	Ciudad	Latitud	Longitud
A	Aguascalientes	Aguascalientes	21.88019081	-102.295687
B	Baja California	Mexicali	32.63994411	-115.474964
C	Baja California Sur	La Paz	24.12383821	-110.3153359
D	Chihuahua	Ciudad Juárez	31.73837791	-106.4832893
E	Chihuahua	Chihuahua	28.63919051	-106.0732948
F	Chihuahua	Delicias	28.19297171	-105.4710889
G	Chihuahua	Cuauhtémoc	28.40659101	-106.8661154
H	Coahuila	Saltillo	25.43374371	-101.0020007
I	Coahuila	Monclova	26.90120436	-101.4172333
J	Coahuila	Piedras Negras	28.69569631	-100.5234364
K	Durango	Durango	24.02641151	-104.6693725
L	Jalisco	Zapopan	20.72156331	-103.3898415
M	Nuevo León	Monterrey	25.67234331	-100.309302
N	San Luis Potosí	San Luis Potosí	22.15200361	-100.9758432
O	Sinaloa	Mazatlán	23.20021201	-106.422215
P	Sinaloa	Culiacán	24.79810391	-107.4084068
Q	Sonora	Hermosillo	29.07550871	-110.9584606
R	Sonora	Obregón	27.51274051	-109.9254931
S	Tamaulipas	Reynosa	26.09234301	-98.2782143
T	Tamaulipas	Nuevo Laredo	27.48630661	-99.5076011
U	Tamaulipas	Matamoros	25.87989751	-97.5047809
V	Zacatecas	Zacatecas	22.77432101	-102.587312

Fuente: Elaboración propia con datos de Google Inc. (2017)

Bibliografía

Aguayo, M. M., S. C. Sarin y H. D. Sherali (2017), «Solving the single and multiple asymmetric Traveling Salesmen Problems by generating subtour elimination constraints from integer solutions», *IISE Transactions*, **50**(1), págs. 45–53.

Ahuja, R. K., T. L. Magnanti y J. B. Orlin (1993), *Network flows: Theory, Algorithms, and Applications*, primera edición, Prentice hall, E.U.A. New Jersey.

Altinel, İ. y T. Öncan (2005), «A new enhancement of the Clarke and Wright savings heuristic for the capacitated vehicle routing problem», *Journal of the Operational Research Society*, **56**, págs. 954–961.

Anbuudayasankar, S. P., K. Ganesh y S. Mohapatra (2014), *Models for practical routing problems in logistics*, primera edición, Springer International Publishing, Suiza.

Çavdar, B. y J. Sokol (2015), «TSP Race: Minimizing completion time in time-sensitive applications», *European Journal of Operational Research*, **244**(1), págs. 47–54.

Ayers, J. B. (2000), *Handbook of supply chain management*, cuarta edición, CRC Press, E.U.A.

Ayu, K. G., N. Septivani, S. Xu y A. S. Kwan (2015), «Optimum Clustering and Routing Model using CVRP Cluster-First, Route-Second in a 3PL Provider», *Industrial Engineering and Operations Management (IEOM). IEEE.*, págs. 1–6.

Aziz, Z. A. (2015), «Ant Colony Hyper-heuristics for Travelling Salesman Problem», *Procedia Computer Science*, **76**, págs. 534–538.

Basheer, P. (2017), *Trade Marketing Focus: Empower Key Influencing Factors*, primera edición, Partridge Publishing, India.

Bektas, T. (2006), «The multiple traveling salesman problem: an overview of formulations and solution proceduree», *Omega*, **34**, págs. 209–219.

B.F.Voigt (1832), *Der Handlungsreisende wie er sein soll und was er zu tun hat, um Aufträge zu erhalten und eines glücklichen Erfolgs in seinen Geschäften gewiß zu sein*, primera edición, Republicado (1981) Verlag Bernd Schramm, Kiel, Alemania.

Bianchi, L., M. Dorigo, L. M. Gambardella y W. J. Gutjahr (2008), «A survey on metaheuristics for stochastic combinatorial optimization», *Natural Computing*, **8**(2), págs. 239–287.

Blum, C. y A. Roli (2003), «Metaheuristics in combinatorial optimization: Overview and conceptual comparison», *ACM Computing Surveys*, **35**(3), págs. 268–308.

Bonavia, M. (1956), *Economía de los transportes*, primera edición, Fondo de Cultura Económica, México.

Changdar, C., G. Mahapatra y R. K. Pal (2014), «An efficient genetic algorithm for multi-objective solid travelling salesman problem under fuzziness», *Swarm and Evolutionary Computation*, **15**, págs. 27–37.

Chopra, S. y P. Meindl (2013), *Administración de la cadena de suministro. Estrategia, planeación y operación*, quinta edición, Pearson Educación, México.

Christopher, M. (2011), *Logistics & Supply Chain Management*, cuarta edición, Pearson, E.U.A.

Cirasella, J., D. S. Johnson, L. A. McGeoch y W. Zhang (2001), «The Asymmetric Traveling Salesman Problem: Algorithms, Instance Generators, and Tests.», *ALENEX 2001: Algorithm Engineering and Experimentation*, **2153**, págs. 32–59.

Clausen, J. (1999), «Branch and Bound Algorithms - Principles and Examples», *Department of Computer Science, University of Copenhagen.*

Cook, W. J. (2012), *In pursuit of the traveling salesman: mathematics at the limits of computation*, primera edición, Princeton University Press, E.U.A.

Dantzig, G. y D. Fulkerson (1954), «Minimizing the number of tankers to meet a fixed schedule», *Naval Research Logistics Quarterly*, **1**(4), págs. 217–222.

Dantzig, G. B. y M. N. Thapa (1997), «Linear Programming 1: Introduction», *Springer Series in Operations Research*, **1**.

Daskin, M. S. y K. L. Maass (2015), *The p-Median Problem*, Springer International Publishing.

Deneubourg, J. L., S. Aron, S. Goss y J. M. Pasteels (1989), «The Self-Organizing Exploratory Pattern of the Argentine Ant», *Journal of Insect Behavior*, **3**, págs. 579–581.

Dent, J. (2008), *Distribution channels. Understanding and managing channels to market*, primera edición, Kogan Page, E.U.A.

Dhanachandra, N., K. Manglem y Y. J. Chanu (2015), «Image Segmentation using K-means Clustering Algorithm and Subtractive Clustering Algorithm», *Procedia Computer Science*, **54**, págs. 764–771.

Díaz, A., F. Glover, H. M. Ghaziri, J. González, M. Laguna, P. Moscato y F. T. Tseng (1996), *Optimización heurística y redes neuronales*, primera edición, Paraninfo, España, Madrid.

Diestel, R. (2005), *Graph Theory*, tercera edición, Springer, Alemania.

Dorigo, M. (1992), *Optimization, Learning and Natural Algorithms*, Tesis Doctoral, Politecnico di Milano, Italy.

Dorigo, M., M. Birattari y T. Stutzle (2006), «Ant Colony Optimization. Artificial Ants as a Computational Intelligence Technique», *IEEE Computational Intelligence Magazine*, **1**(4), págs. 28–39.

Dorigo, M. y G. D. Caro (1999), «Ant colony optimization: A new meta-heuristic», *IEEE Conference: Evolutionary Computation*, **2**, pág. 1477.

Dorigo, M. y L. M. Gambardella (1997), «Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem», *IEEE Transactions On Evolutionary Computation*, **1**(1), págs. 28–39.

Dorigo, M., V. Maniezzo y A. Colorni (1996), «The Ant System: Optimization by a colony of cooperating agents», *IEEE Transactions on Systems, Man, and Cybernetics*, **26**, págs. 1–13.

Dorigo, M. y T. Stützle (2004), *Ant Colony Optimization*, primera edición, Massachusetts Institute of Technology, Londres, Inglaterra.

Duff, I. S., A. M. Erisman y J. K. Reid (2017), *Direct methods for sparse matrices*, segunda edición, Numerical methods and scientific computation, Oxford University Press, Reino Unido.

Eldem, H. y E. Ülker (2017), «The application of ant colony optimization in the solution of 3D traveling salesman problem on a sphere», *Engineering Science and Technology, an International Journal*, **20**(4), págs. 1232–.

Gambardella, L. M. y M. Dorigo (1996), «Solving Symmetric and Asymmetric TSPs by Ant Colonies», *IEEE Conference on Evolutionary Computation*, pág. 622–627.

García Travieso, M. V. (2014), «Problema del viajante de comercio (TSP). Métodos exactos de resolución», *Tesis*, Universidad de La Laguna, España.

Garey, M. R. y D. S. Johnson (1979), *Computers and intractability. A guide to the theory of NP-completeness*, primera edición, W.H. Freeman and company, E.U.A.

Gimenez, Y. (2010), «Clasificación no supervisada: El método de k-medias», *Tesis*, Universidad de Buenos Aires.

Glover, F., G. Gutin, A. Yeo y A. Zverovich (2001), «Construction heuristics for the asymmetric TSP», *European Journal of Operational Research*, **129**(3), págs. 555–568.

Golden, B., L. Bodin, T. Doyle y J. W. Stewart (1980), «Approximate Traveling Salesman Algorithms», *Operations Research*, **28**(3), págs. 694–711.

Google Inc. (2011), «Google APIs Terms of Service», Disponible en: https://developers.google.com/terms/terms-2011?hl=ES. Consultado el 15 de noviembre del 2017.

Google Inc. (2017), «Google Maps Distance Matrix API», Disponible en: https://developers.google.com/maps/documentation/distance-matrix/?hl=ES, consultado el 15 de noviembre del 2017.

Goss, S., S. Aron, J. Deneubourg y J. Pasteels (1989), «Self-organized shortcuts in the Argentine ant.», *Naturwissenschaften*, **76**, págs. 579–581.

Gouveia, L., M. Leitner y M. Ruthmair (2017), «Extended formulations and branch-and-cut algorithms for the Black-and-White Traveling Salesman Problem», *European Journal of Operational Research*, **262**(3), págs. 908–928.

Gutin, G. y A. P. Punnen (2007), *The traveling salesman problem and its variations*, 9ª edición, Springer US, E.U.A.

Gutin, G., A. Yeo y A. Zverovich (2002), «Traveling salesman should not be greedy: domination analysis of greedy-type heuristics for the TSP», *Discrete Applied Mathematics*, **117**(1-3), págs. 81–86.

Hahsler, M. y K. Hornik (2007), «TSP - Infrastructure for the Traveling Salesperson Problem», *Journal of Statistical Software*, **23**(2), págs. 1–21.

Helsgaun, K. (2009), «General k-opt submoves for the Lin–Kernighan TSP heuristic», *Mathematical Programming Computation*, **1**(2–3), págs. 119–163.

Hernández Pérez, H. y J. J. Salazar González (2004), «A branch-and-cut algorithm for a traveling salesman problem with pickup and delivery», *Discrete Applied Mathematics*, **145**(1), págs. 126–139.

Herrán, A., J. M. Colmenar, R. Martí y A. Duarte (2018), «A parallel variable neighborhood search approach for the obnoxious p-median problem», *International Transactions in Operational Research*, págs. 1475–3995.

Huang, C. y C. Wong (2018), «Optimization of crane setup location and servicing schedule for urgent material requests with non-homogeneous and non-fixed material supply», *Automation in Construction*, **89**, págs. 183–198.

Huang, W. y J. X. Yu (1964), «Scheduling of Vehicles from a Central Depot to a Number of Delivery Points», *Operations Research*, **12**(4), págs. 568–581.

Huang, W. y J. X. Yu (2017), «Investigating TSP Heuristics for Location-Based Services», *Data Science and Engineering*, **2**(1), págs. 71–93.

Instituto Nacional de Estadística y Geografía (2017), «Catálogo Único de Claves de Áreas Geoestadísticas Estatales, Municipales y Localidades», Disponible en: http://www.inegi.org.mx/geo/contenidos/geoestadistica/CatalogoClaves.aspx, consultado el día 3 de diciembre del 2017.

Ismkhan, H. (2017), «Effective heuristics for ant colony optimization to handle large-scale problems», *Swarm and Evolutionary Computation*, **32**, págs. 140 – 149.

JOMINI, A. H. (1838), *Précis de l'art de la guerre ou Nouveau tableau analytique des principales combinaisons de la stratégie, de la grande tactique et de la politique militaire*, primera edición, Republicado (1977) Ivrea, Francia.

LAWLER, E., D. SHMOYS, A. R. KAN y J. LENSTRA (1985), *The traveling salesman problem*, primera edición, John Wiley & Sons, Gran Bretaña.

LIN, S. (1965), «Computer Solutions of the Traveling Salesman Problem», *The Bell System Technical Journal*, **44**(10), págs. 2245–2269.

LIN, S. y B. W. KERNIGHAN (1973), «An Effective Heuristic Algorithm for the Traveling-Salesman Problem», *Operations Research*, **21**(2), págs. 498–516.

LUENBERGER, D. G. y Y. YE (2016), «Linear and Nonlinear Programming», *International Series in Operations Research & Management Science*, **228**(4), págs. XIII–546.

MAHI, M., O. K. BAYKAN y H. KODAZ (2015), «A new hybrid method based on Particle Swarm Optimization, Ant Colony Optimization and 3-Opt algorithms for Traveling Salesman Problem», *Applied Soft Computing*, **30**, págs. 484–490.

MARR, N. (1984), «Marketing Freight Transport: The Need for Customer-orientation», *Service Industries Journal*, **4**(3), págs. 125–132.

MILLER, C. E., A. W. TUCKER y R. A. ZEMLIN (1960), «Integer Programming Formulation of Traveling Salesman Problems», *Journal of the ACM*, **7**(4), págs. 326–329.

MLADENOVIĆ, N., J. BRIMBERG, P. HANSEN y J. A. MORENO-PÉREZ (2007), «The p-median problem: A survey of metaheuristic approaches», *European Journal of Operational Research*, **179**, págs. 927–939.

MLADENOVIC, N. y P. HANSEN (1997), «Variable neighborhood search», *Computers Operations Research*, **24**, págs. 1097–1100.

Necula, R., M. Breaban y M. Raschip (2015), «Performance Evaluation of Ant Colony Systems for the Single-Depot Multiple Traveling Salesman Problem», *EVOLVE - A Bridge between Probability, Set Oriented Numerics, and Evolutionary Computation VI*, **674**, págs. 210–223.

Necula, R., M. Raschip y M. Breaban (2018), «Balancing the Subtours for Multiple TSP Approached with ACS: Clustering-Based Approaches Vs. MinMax Formulation», *EVOLVE - A Bridge between Probability, Set Oriented Numerics, and Evolutionary Computation VI. Advances in Intelligent Systems and Computing*, **674**, págs. 210–223.

Nilofer, M. y M. Rizwanullah (2017), «The New Approach to Traveling Salesman Problem using Branch and Bound Method with case study of Domino's Pizza Centers», *Advances in Fuzzy Mathematics*, **12**(4), págs. 1017–1033.

Olivos, P. C., F. O. Carrasco, J. L. M. Flores y Y. M. Moreno (2015), «Modelo de gestión logística para pequeñas y medianas empresas en México.», *Contaduría y Administración*, **60**, págs. 181–203.

Osman, I. y G. Laporte (1996), «Metaheuristics: A bibliography», *Annals of Operations Research*, **63**(5), págs. 511–623.

Padberg, M. y G. Rinaldi (1987), «Optimization of a 532-city symmetric traveling salesman problem by branch and cut», *Operations Research Letters*, **6**(1), págs. 1–7.

Padberg, M. y G. Rinaldi (1991), «A Branch-and-Cut Algorithm for the Resolution of Large-Scale Symmetric Traveling Salesman Problems», *Society for Industrial and Applied Mathematics*, **33**(1), págs. 60–100.

Pekny, J. y D. Miller (1992), «A parallel branch and bound algorithm for solving large asymmetric traveling salesman problems», *Mathematical Programming*, **55**, págs. 17–33.

Raffo, E. y E. Ruiz (2005), «Modelo de optimización de la ruta de entrega», *Applied Soft Computing*, **8**(1), págs. 75–83.

Ragsdale, C. (2012), *Spreadsheet modeling & decision analysis. A practical introduction to management science*, 6ª edición, South-Western, Cengage Learning, E.U.A.

Real Academia Española (2014), *Diccionario de la lengua española*, 23ª edición, España.

Reinelt, G. (1991), «TSPLIB—A Traveling Salesman Problem Library.», *ORSA Journal on Computing*, **3**, págs. 376–384.

Reinelt, G. (1994), *The Traveling Salesman: Computational Solutions for TSP Applications*, primera edición, Springer, Alemania.

Rodrigue, J. P., C. Comtois y B. Slack (2013), *The Geography of Transport Systems*, tercera edición, Routledge, Londres, Inglaterra.

Russell, J. T., W. R. Lane y K. W. King (2005), *Kleppner publicidad*, 16ª edición, Pearson Educación, México.

Schlapfer, M. F. (1997), *A comparison of genetic and other algorithms for the traveling salesman problem*, Tesis de Maestría, San Jose State University, proQuest Dissertations and Theses.

Schwarz, L. B. (2008), «The Economic Order-Quantity (EOQ) Model», *Building Intuition. International Series in Operations Research & Management Science*, **115**, págs. 135–154.

Secretaría de Economía (2008), «Decreto por el que se aprueba el Programa Sectorial de Economía 2007-2012.», Diario Oficial de la Federacíon, publicado el día miércoles 14 de mayo.

Segura, E., R. B. C. Benitez y A. Lozano (2015), «Implications of the assumptions on which the p-median problem are based when distribution network design», *Transportation Research Procedia*, **27**, págs. 1137–1143.

Sosa, R. (2012), *Documentos, logística de transporte, seguro y embalaje internacional de mercancías*, tercera edición, Carpenter consulting group, México, DF.

Soylu, B. (2017), «A general variable neighborhood search heuristic for multiple traveling salesmen problem», *Computers & Industrial Engineering*, **90**(1), págs. 1137—-1143.

Sun, W. y Y.-X. Yuan (2006), «Optimization Theory and Methods. Nonlinear Programming», *Springer US*, **1**, págs. 135–154.

Taha, H. A. (2012), *Investigación de operaciones*, 9ª edición, Pearson Educación, México.

Toro, E., J. Santa y M. Granada (2013), «Solución del problema de ruteamiento de vehículos en la distribución de papa en Colombia. Ingeniería Industrial», *Scientia et Technica*, **18**(1), págs. 139–148.

Toth, P. y D. Vigo (2014), *Vehicle Routing Problems, Methods, and Applications*, segunda edición, Society for Industrial and Applied Mathematics, E.U.A.

Vanhoucke, M. (2013), «The Critical Path Method», *Project Management with Dynamic Scheduling, Springer*, págs. 37–57.

Venkatesh, P. y A. Singh (2014), «Two metaheuristic approaches for the multiple traveling salesperson problem», *Applied Soft Computing*, **26**(1), págs. 74–89.

Woodruff, D. L. (1998), *Advances in Computational and Stochastic Optimization, Logic Programming, and Heuristic Search. Interfaces in Computer Science and Operations Research*, primera edición, Springer Science & Business Media, E.U.A.

Wu, J. (2012), *Advances in K-means Clustering: a Data Mining Thinking*, Tesis Doctoral, Springer theses, Berlin.

Xu, X., H. Yuan, M. Liptrott y M. Trovati (2017), «Two phase heuristic algorithm for the multiple-travelling salesman problem», *Soft Computing*, págs. 1–15.

Yakici, E. y A. Yiğit (2017), «An Evolutionary Algorithm for p-Median Problem with Attribute Equity Constraint», *Journal of the Faculty of Engineering and Architecture*, **32**(4), págs. 1–10.

Yu, Q., D. Wang, D. Lin, Y. Li y C. Wu (2012), «A Novel Two-Level Hybrid Algorithm for Multiple Traveling Salesman Problems», *Advances in Swarm Intelligence*, **7331**, págs. 497–503.

Resumen autobiográfico

Azcarie Manuel Cabrera Cuevas

Candidato para obtener el grado de
Maestría en Logística y Cadena de Suministro

Universidad Autónoma de Nuevo León
Facultad de Ingeniería Mecánica y Eléctrica

Tesis:
Método de solución de ruta entre ciudades para reparto de muestras

Nací en Toluca, Estado de México, México. mis padres Manuel Cabrera Chamorro y María Lucía Cuevas Zarco. Realicé mis estudios de Licenciatura en la Facultad de Economía de la Universidad Autónoma del Estado de México, obteniendo en el año 2014 el título de Licenciado en Relaciones Económicas Internacionales con acentuación en Comercio Internacional.