UNIVERSIDAD NACIONAL DE TRUJILLO

FACULTAD DE CIENCIAS FISICAS Y MATEMÁTICA

ESCUELA PROFESIONAL DE INFORMÁTICA

Modelo de optimización y metaheurística para localizar centros de distribución de productos y su transporte usando ruteo de vehículos abierto en una red logística

TESIS

Para optar el Título Profesional de Ingeniero Informático

Autor: Cruzado Berrú, Luis Enrique

Asesor: Dr. José Antonio Rodríguez Melquiades

Trujillo – Perú

2021

Dedico esta tesis a :

A mi madre por el apoyo en todo este tiempo a lo largo de mi vida.

Agradecimientos

A mi asesor el Dr. Jose A. Rodriguez Melquiades y al Mg. Edwar G. Lujan Segura, por sus invalorables contribuciones, sin los cuales esta investigacion no podría haberse concluido.

UNIVERSIDAD NACIONAL DE TRUJILLO
UNT

ESCUELA PROFESIONAL - PREGRADO

ACTA DE SUSTENTACIÓN DE TESIS

PARA OPTAR EL TÍTULO DE INGENIERO INFORMÁTICO

En la ciudad de Trujillo, en la Sala Virtual https://meet.google.com/kni-xbjh-srd, siendo las 16 horas, del día martes 6 de julio de 2021, se reunió el Jurado conformado por:

Presidente (a): IRIS AUREA CRUZ FLORIAN

Secretario (a): JORGE DAVID BRAVO ESCALANTE

Miembro: JOSÉ ANTONIO RODRIGUEZ MELQUIADES

Para el acto de: (Marcar el que corresponde)

1. (X)Sustentación de Tesis intitulada:

"MODELO DE OPTIMIZACION Y METAHEURISTICA PARA LOCALIZAR CENTROS DE DISTRIBUCION DE PRODUCTOS Y SU TRANSPORTE USANDO RUTEO DE VEHICULOS ABIERTO"

Con el fin de optar al Título Profesional de INGENIERO INFORMATICO (a) por el/la graduado/a(os,as):

Br. **CRUZADO BERRU LUIS ENRIQUE**

Después de concluido el acto de sustentación y luego de que el (los) mencionado(s) ha (n) dado respuesta a las preguntas respectivas, el Jurado Evaluador, declara:

1. ( ) Aprobado, con mención honrosa. La cual amerita su publicación
2. (X) Aprobado, por unanimidad
3. ( ) Aprobado, por mayoría
4. ( ) Desaprobado

Según el Art. 13º del Reglamento de Grados y Títulos de la Escuela de Informática, Facultad de Ciencias Físicas y Matemáticas de la Universidad Nacional de Trujillo.

Por lo tanto los/las Graduados (as) se encuentra expeditos (X), impedidos ( ) para realizar los trámites correspondientes para la obtención del Título Profesional de INGENIERO (a) INFORMÁTICO

Siendo las 5:36 PM se dio por terminado el acto de sustentación.

Iris Aurea Cruz Florián
Presidente (a)

Jorge David Bravo Escalante
Secretario(a)

José Antonio Rodríguez Melquiades
Miembro

F-M01.03.04-DRT/PG-03 – (Rev. 1)

Resumen

La investigación bibliográfica realizada revela que existe preocupación en lo relacionado a la ubicación de los centros de distribución y el transporte sostenible de los productos hacia los clientes mediante vehículos apropiados para zonas urbanas, pues es necesario mantener un orden en las mismas y en consecuencia se preserva la salud y el medio ambiente urbano. Esto es posible si hacemos uso de los conocimientos computacionales aplicados en el contexto urbano.

En este sentido la investigación realizada en esta tesis, considerando la complejidad del problema NP-completo, logro alcanzar el objetivo principal que fue desarrollar e implementar un modelo de optimización y una metaheurística basados en los paradigmas de algoritmos de programación lineal entera binaria y de aproximación, para viabilizar una red logística. Los resultados de las experiencias computacionales realizadas son alentadores los cuales pueden ser aplicados en diversas regiones sin necesidad de grandes cambios en el modelo propuesto y por tanto también en la metaheurística.

Palabras claves: logística, localización, ruteo de vehículos, modelo de optimización, metaheurística.

Abstract

The bibliographic research carried out reveals that there is concern regarding the location of distribution centers and the sustainable transport of products to customers by appropriate vehicles for urban areas, since it is necessary to maintain order in them and consequently it is preserved health and the urban environment. This is possible if we make use of the computational knowledge applied in the urban context.

In this sense, the research carried out in this thesis, considering the complexity of the NP-complete problem, achieved the main objective that was to develop and implement an optimization model and a metaheuristic based on the paradigms of binary integer linear programming and approximation algorithms, to enable a logistics network. The results of the computational experiments carried out are encouraging, which can be applied in various regions without the need for major changes in the proposed model and therefore also in the metaheuristics.

Keywords: logistics, location, vehicle routing, optimization model, metaheuristics.

Índice de figuras

Índice de tablas

Índice general

Capítulo 1

Introducción

Las ciudades de diversos países están en constante desarrollo en busca de un bienestar para la población como un todo. Individualmente cada persona en el mundo actual al buscar alcanzar mejores días para él y sus familiares, determina un crecimiento poblacional en las zonas urbanas (Bugliarello, 2006). En este contexto, la logística urbana bajo el principio de atención oportuna en el momento o tiempo oportuno, aliado al transporte se constituyen como una oportunidad para atender a las personas, a través del transporte ya sea de pasajeros o de productos y de esta manera atender la fuerte demanda de la población. Esa es la realidad actual del mundo globalizado que necesita atención por parte de las autoridades.

La logística urbana tendrá éxito si se elaboran proyectos de inversión de modo tal que pueda contribuir en la atención de este tipo de servicio urbano. En este punto se hace necesario la elaboración de modelos de optimización y estrategias algorítmicas para obtener óptimas y buenas soluciones debido a la complejidad computacional del problema en estudio, como lo es ruteo de vehículos en general. En particular ruteo abierto. Nuestro interés es optimizar costos el cual esta en función de la distancia recorrida por el vehículo en el proceso atender a los denominados clientes, quienes previamente localizan a algún centro de distribución (facilidad).

1.1. Justificación de la investigación

El crecimiento poblacional en las zonas urbanas ha generado un incremento en la demanda por bienes y servicios. Esto ha determinado la mejora y/o creación de empresas en diversos rubros comerciales, las cuales con diferentes estrategias llegan a sus potenciales

clientes y por tanto lograr la satisfacción de los mismos (Melquiades, 2015). Para alcanzar el objetivo, los centros de distribución son abastecidas por productores los cuales considerando el mundo competitivo cuentan con un sistema logístico eficiente. En etapa posterior los centros de distribución son localizados por los clientes para recibir la atención respectiva y de este modo tal centro responda al ambiente dinámico del sector comercial.

El transporte es un medio fundamental para el traslado de los bienes y servicios, tanto para abastecer a los centros de distribución como a los clientes. Además, siendo que el transporte es un medio altamente contaminante debido a la combustión química que realizan los vehículos para su funcionamiento, ha determinado tener en cuenta el concepto de logística urbana y últimamente logística verde.

Por razones expuestas este trabajo de investigación ofrece una alternativa de solución para los problemas de abastecimiento de productos en las ciudades mediante propuestas cuantitativas, es decir, haciendo un estudio detallado de su funcionamiento es posible modelar y solucionar usando el paradigma de programación lineal entera binaria, para mediante una localización y ruteo se mejore la logística urbana. Además, al ser el problema computacionalmente difícil, también es necesario usar nuevas estrategias algorítmicas denominadas metaheurísticas.

1.2. Formulación del problema

De la descripción expresada con respecto a los cambios que experimenta el mundo actual, es necesario proporcionar una alternativa de solución con único interés de lograr el bienestar de la población bajo los principios de la logística. Se planteó el siguiente problema:

¿Cómo viabilizar una red logística para localizar centros distribución de productos y su transporte?

1.3. Hipótesis

Considerando las razones expuestas en el contexto de la logística para un óptimo abastecimiento de productos en general, nos planeamos la siguiente respuesta: El desarrollo de un modelo de optimización y una metaheurística de colonia de hormigas, basado en localizar centros de distribución y ruteo de vehículos abierto para el transporte, viabiliza la red para un recorrido en el contexto de la logística.

1.4. Objetivos

Para establecer lo que se pretende con la investigación y de este modo contribuir con la solución del problema formulado, se tienen los siguientes objetivos:

1.4.1. General

Desarrollar e implementar un modelo de optimización y una metaheurística, basado en colonia de hormigas, para viabilizar el recorrido en una red logística.

1.4.2. Específicos

a) Aplicar una metodología de programación lineal entera binaria, pertenece a la clase de complejidad NP para solucionar un problema, de modo tal que optimice recursos como tiempo, distancia y disminución de contaminación ambiental en la zona de estudio.

b) Investigar estrategias algorítmicas para solucionar problemas NP-Difíciles.

c) Investigar los problemas de localización y ruteo de vehículos incluido sus variaciones para tomarlo como referencia, y de este modo elaborar una nueva propuesta que sea usada para la solución del problema.

1.5. Estructura de la tesis

El presente trabajo está dividido en cinco capítulos. El primer capítulo presenta los aspectos generales de la investigación realizada tal como justificación, formulación del problema, hipótesis, los objetivos y la estructura de la tesis. En el capítulo dos se presenta el referencial teórico, soporte del tema, contemplando los conceptos de logística y desarrollo sustentable; optimización combinatoria y complejidad computacional; ruteo de vehículos y estrategias algorítmicas denominadas metaheurísticas. Finalmente el método empleado en la investigación.

El tercer capítulo constituye el tema central de la tesis. Se presenta el modelo de optimización propuesto y la metaheurística, además de las respectivas experiencias computacionales realizadas. En el cuarto capítulo se presentan los resultados y discusión obtenida en la investigación. En el capítulo cinco se presentan las consideraciones finales obtenidas en esta tesis. Inicialmente se presentan las conclusiones, seguida de las recomendaciones para futuras investigaciones relacionadas al tema en cuestión. Finalmente las referencias bibliográficas usadas para la investigación en esta tesis y la declaración jurada y autorización de la tesis.

Capítulo 2

Materiales y métodos

En este capítulo se explican los conocimientos investigados los cuales son el soporte a la investigación realizada, pues ayudaron a consolidar las bases del conocimiento científico para elaborar esta tesis, como lo son los temas de logística, optimización combinatoria, complejidad computacional, programación lineal entera, meta heurísticas, entre otros conocimientos sin los cuales sería difícil de modelar y solucionar matemática y computacionalmente cualquier tipo de problema de optimización.

2.1. Marco teórico

2.1.1. Logística y desarrollo sustentable

El concepto de logística surgió hace buen tiempo, cuando las fuerzas armadas tenían que planificar la mejor estrategia para transportar el material bélico a los campos de batalla. Posteriormente los conceptos se han ampliado y/o se han acercado a la realidad en donde se ponga en práctica, de este modo logística se define como colocar cierto producto en la cantidad, lugar, plazo de entrega y calidad correcta; a menor costo (Monteiro da Costa Cruz and De Alvarenga Rosa, 2009). Por lo tanto, la gestión logística esta dado por la coordinación entre las actividades logísticas realizadas. Se entiende que el transporte, *stock* y depósito de los productos constituyen elementos de la logística.

Una definición acertada y muy cerca para los intereses de la presente investigación es dada en Ghiani et al. (2004), quienes establecen que la logística se ocupa de la planificación y el control de los flujos de materiales e informaciones relacionadas en las organizaciones privadas y del Estado, siendo su misión entregar los pedidos en buen estado, en el lugar correcto y en la hora apropiada siguiendo un proceso de optimización de los costos operacionales que

demanda el proceso; pero satisfaciendo un determinado conjunto de restricciones o condiciones.

A pesar que la logística trae muchos beneficios, su modelo necesita actuar en sociedad con otras actividades para de este modo generar estrategias de crecimiento e inversiones en conjunto. De este modo surge la cadena de suministro[1], el cual establece sociedades entre la empresa, abastecedores, clientes y transportistas.

La cadena de suministro ha significado un desarrollo para los países pues genera nuevas fuentes de empleo mejorando así la capacidad adquisitiva de las personas quienes podrán satisfacer sus necesidades básicas, generando mayor producción y también mayor generación de restos sólidos urbanos. Esto es bueno si es debidamente administrado por parte de las autoridades municipales. De este modo va surgiendo el desarrollo sustentable, (Figura 2.1), el cual estará garantizado si se consideran los aspectos económico, social y ambiental, donde su intersección garantiza la calidad de vida en el espacio urbano y el equilibrio en las clases sociales en busca del bienestar (Tanguay et al., 2010) apud (Melquiades, 2015). Esto compromete a las cadenas de suministro a planificar sus actividades ya que está de por medio el adecuado uso de los recursos naturales y la protección del medio ambiente.

Una definición reciente relacionada con la gestión de la cadena de suministro es dada en Stadtler et al. (2015), quienes la definen como la tarea de integrar una cadena de suministro y coordinar los flujos de material, información y financieros para satisfacer las demandas de los clientes, con el objetivo de mejorar la competitividad de una cadena de suministro como un todo. Actualmente en el mundo globalizado, el término cadena de suministro sigue teniendo vigencia, pues también se aplica en las empresas transnacionales quienes al tener sucursales ubicados en diferentes países tienen que coordinar tales flujos de manera eficiente.

[1] Supply chain.

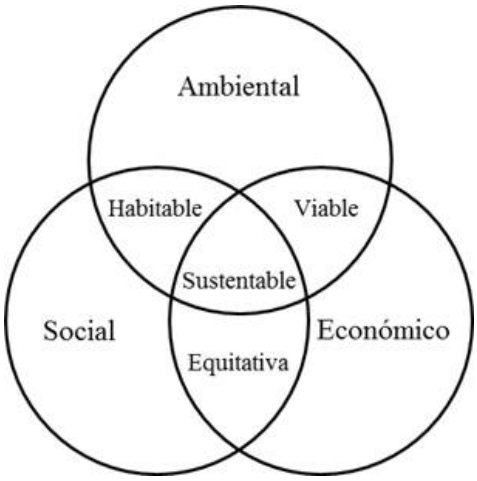

Figura 2.1: Aspectos claves para el desarrollo sustentable.
Fuente: Tanguay et al. (2010)

La Figura 2.2 muestra la dependencia que existe en los niveles estratégico y operacional. Esto es muy importante por cuanto es en estos niveles de la cadena de suministro donde se debe diseñar una red logística, y por tanto desarrollar un modelo de optimización para poner en práctica la localización de las facilidades y en consecuencia el flujo de las demandas a atender mediante el transporte para de este modo alcanzar los objetivos trazados por las empresas.

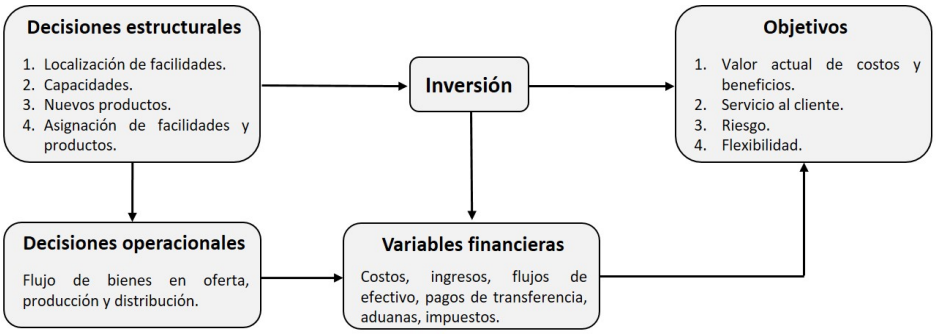

Figura 2.2: Interdependencia entre niveles de planificación estratégica y operativa
Fuente: Stadtler et al. (2015)

Problemas ambientales y sociales siempre han existido, solamente que tal vez no se le daba la importancia debida o porque simplemente la sociedad tenía otra forma de pensar. Sin embargo con el pasar de los años tales problemas se han agudizado, debido al incremento po-

blacional en las ciudades generando de este modo un necesario incremento en la producción. Ver Figura 2.3. Lamentablemente tal aumento productivo trae sus consecuencias inicialmente no percibidas en el sector agrícola e industrial. Una consecuencia es el cambio climático que esta amenazando la biodiversidad y por tanto, poniendo en peligro la continuidad de las diversas producciones.

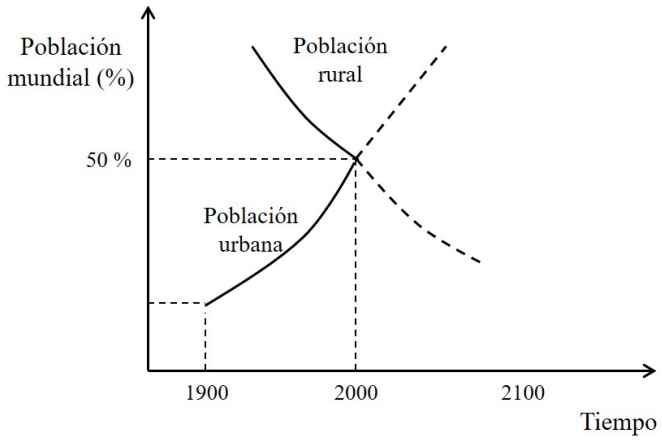

Figura 2.3: Crecimiento de la población urbana.
Fuente: Bugliarello (2006)

La satisfacción de las necesidades básicas no pueden ser evitadas, por ello y considerando el peligro que ya se tiene en el presente, los países están estableciendo nuevas reglas que los sectores agrícola e industrial deben cumplir, por ello la tendencia global es la regulación ambiental y social de modo integral. Actualmente las grandes industrias ya esta cumpliendo con tales normas, pues entienden que la responsabilidad para mitigar el impacto ambiental es compartida entre empresa-consumidor. Esto se llama cadena de suministro sustentable Bouchery et al. (2017).

En el contexto de proteger el medio ambiente Bouchery et al. (2017) también discuten el aspecto operativo de la cadena de suministro, por ello se tiene logística y localización de facilidades verde. Tradicionalmente la logística siempre se ha preocupado por minimizar costos y maximizar el beneficio. Esto se entiende, sin embargo actualmente el sector indus-

trial ha agregado metas sustentables para alcanzar sus objetivos comerciales ya que el nuevo objetivo también es mitigar el impacto social y ambiental de sus productos e operaciones.

Logística verde Bouchery et al. (2017), se refiere a medir, analizar y mitigar el impacto ambiental de las actividades logísticas, esto incluye reducir el consumo de la energía no renovable, emisiones aéreas, emisiones del efecto del gas invernadero [2] y restos. El transporte es una actividad altamente contaminante tal como muestra la Figura 2.4, donde se proporciona un estimado promedio de las emisiones de CO2 originado por el transporte de carga y las actividades logísticas. Se afirma que el transporte es responsable del $90\,\%$ de tales emisiones.

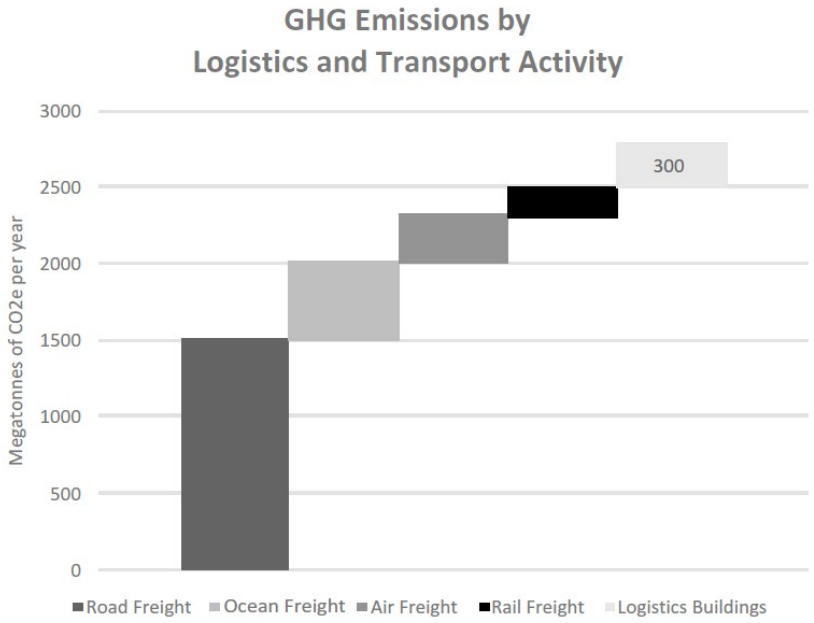

Figura 2.4: Emisiones del efecto del gas invernadero y actividad de transporte
Fuente: Bouchery et al. (2017)

La fórmula para calcular la emisión de CO2 esta dada en Bouchery et al. (2017) esta dada como:

$$E = \sum_{a} [F_a \times EF_a] \tag{2.1}$$

donde F representa el consumo total de combustible usado, multiplicado por un factor de emisión EF de tal combustible. Este factor es igual al carbon contenido en el combustible

[2] GreenHouse Gas emissions - GHG emissions.

multiplicado por $44/12$ y a representa el tipo de combustible.

La localización de facilidades verde es una actividad que se ejecuta una vez diseñado la red logística. Es muy importante porque ayuda a localizar fábricas, centros de ensamblaje, depósitos, así como también contribuye en la toma de decisiones para que el flujo de productos pueda viajar desde el centro de oferta hacia los clientes. Esta actividad es posible mediante el desarrollo de modelos de optimización de programación lineal entera que ayudaran a reducir distancias y por tanto, ahorro de combustible generando menos contaminación.

2.1.2. Optimización combinatoria y complejidad computacional

Los algoritmos juegan un rol muy importante en ciencia de la computación y son esenciales para modelar diversos problemas reales. Diseño y análisis de algoritmos para el caso de problemas específicos se encuentran en Cormen et al. (2009). La meta de este campo de investigación según Neumann and Witt (2010) es obtener algoritmos probablemente óptimos con respecto al tiempo de ejecución y/o aproximación para el problema en estudio. De este modo estudiar un problema específico permite conocer mucho acerca del problema, el cual puede ser usado para su desarrollo y análisis de algoritmos.

En muchas situaciones no es posible desarrollar algoritmos para problemas específicos que ofrezcan una buena performance. Esto sucede con problemas nuevos y complejos o que no han sido estudiados anteriormente. Ejemplos lo encontramos en la denominada computación bioinspirada y la programación entera mixta quienes no tienen pruebas rigurosas que den límites a la calidad del tiempo de ejecución y/o aproximación. Sin embargo tienen puntos a su favor, pues proporcionan una alta performance cuando se los experimenta para casos particulares, e incluso a veces es difícil entender como es que funcionan correctamente en un entorno particular (Neumann and Witt, 2010).

A continuación se discuten conceptos revisados durante la investigación bibliográfica, en

relación a problemas de optimización combinatoria y complejidad computacional.

2.1.2.1. Optimización combinatoria

Dos definiciones interesantes debido a Talbi (2009) relacionada con problemas de optimización y óptimo global son:

Definición: Un problema de optimización se define como un par ordenado (S, f), donde S representa el conjunto de soluciones viables o espacio de busca y $f : S \rightarrow \mathbb{R}$ una función objetivo que es la que optimiza un problema dado.

Definición: Una solución $s^* \in S$ es definido como un óptimo global, sí el alcanza el mejor valor en la función objetivo de entre todas las posibles soluciones que estan en el espacio de busca, es decir, $f(s^*) \leq f(s), \forall s \in S$.

Considerando estas definiciones, los investigadores han elaborado diversas metodologías para solucionar problemas mediante el diseño de modelos de optimización. La Figura 2.5 de acuerdo con Talbi (2009) muestra la clasificación de los modelos de optimización.

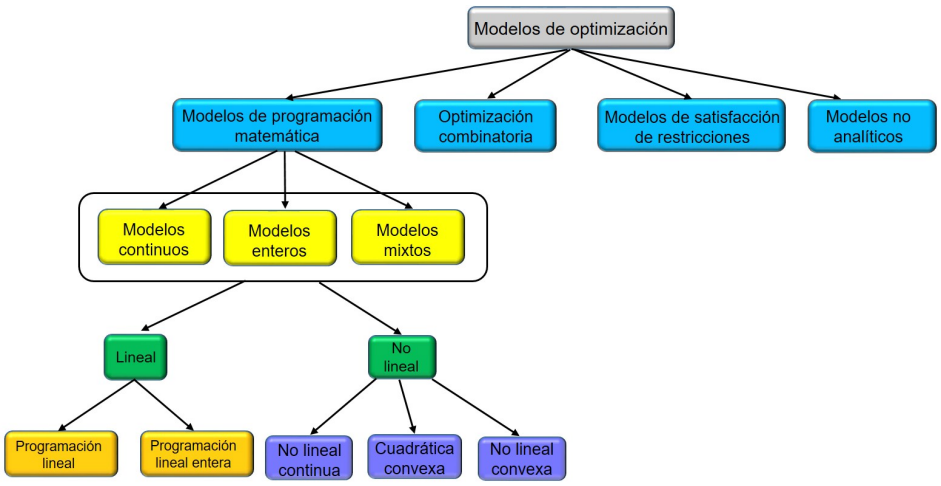

Figura 2.5: Clasificación de los modelos de optimización.
Fuente: Talbi (2009)

Todo modelo de optimización presenta los siguientes elementos:

(a) **Variable de decisión:** Constituye un conjunto de términos cuyos valores son descono-

cidos y que se pretenden obtener mediante el modelo desarrollado, para de este modo tener la solución del problema.

(b) **Función objetivo:** Expresión matemática formada por variables y parámetros denominados constantes. La meta es maximizar o minimizar la función objetivo.

(c) **Restricciones:** Conjunto de inecuaciones que establecen las condiciones bajo las cuales se pretende obtener la solución del problema.

(d) **Límite de las variables:** Se refiere el alcance que tienen las variables pudiendo ser reales, enteros o binario.

Por lo tanto, todo modelo de optimización que maximiza o minimiza una función objetivo, tiene una estructura, donde $\forall i$ las variables son representadas por x_i, los parámetros denominados constantes se representan por c_i; además a_{ij} también son parámetros que representan actividades realizadas en el problema. Finalmente el alcance de las variables que en este caso son $x_i \geq 0$; también pueden ser de tipo enteros o binarios. El valor que asume las variables del problema determinan modelos continuos, enteros o mixtos, del cual se tienen modelos de programación lineal y programación lineal entera tal como se muestra en la Figura 2.5 anteriormente presentada. Para fines de esta tesis nuestro interés es la programación lineal entera.

$$Min\ Z = c_1x_1 + c_2x_2 + c_3x_3 + \ldots + c_nx_n$$

s.t.

$$a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \ldots + a_{1n}x_n \leq b_1$$

$$a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + \ldots + a_{2n}x_n \leq b_2$$

$$\vdots$$

$$a_{m1}x_1 + a_{m2}x_2 + a_{m3}x_3 + \ldots + a_{mn}x_n \leq b_m$$

$$x_1, x_2, \ldots, x_n \geq 0, \textit{enteros}$$

De acuerdo con Talbi (2009) una clase mas general de problemas de programación entera, son los problemas de optimización combinatoria los cuales se caracterizan por el uso de variables de decisión de tipo discreto y tienen un espacio de busca finito. Esta forma de hacer optimización es muy importante pues permite modelar y solucionar problemas presentados en la vida real.

Dos ejemplos que nos hacen ver su importancia para casos reales son determinar el menor recorrido que enlace dos lugares diferentes en una zona geográfica, es decir, ruta corta; otro ejemplo es la construcción de un recorrido para visitar cada lugar. Este segundo ejemplo se denomina ruteo de vehículos. La importancia que presenta la optimización combinatoria es tal que actualmente se tienen dos categorías, es decir, uno para problemas con variables continuas y otro para casos con variables discretas.

El primer caso es estudiado en cursos de cálculo para el cual usa el concepto de la derivada. Nuestro interés es la categoría de variables discretas, ya que los problemas combinatorios tienen como objetivo maximizar o minimizar una función denominada objetivo el cual debe cumplirse, según cierto conjunto de condiciones también llamadas restricciones.

Definición: Un problema de optimización combinatoria se define como un terna (S, f, R), donde S es un espacio de busca; f una función objetivo la cual se desea maximizar o minimizar; y R un conjunto de condiciones que se deben cumplir para obtener las soluciones viables.

Esta definición dada por Neumann and Witt (2010) muestra que la meta es hallar una solución óptima que permita encontrar un valor máximo para el caso maximizar la función objetivo, o un valor mínimo para minimizar. En ambos casos deben cumplirse las condiciones dadas en el conjunto R. Para fines de la presente tesis, el modelo de optimización de ruteo propuesto esta basado en la representación gráfica una red expresada como un grafo

$G = (V, A)$, donde V denota el conjunto de nodos y A el conjunto de arcos que unen los nodos. Este trabajo de investigación modela un problema real como un problema de optimización combinatoria. Información adicional sobre grafos, algoritmos y complejidad se encuentra en Jungnickel (2013).

2.1.2.2. Complejidad computacional

En ciencia de la computación los problemas, según el grado de dificultad, se clasifican en fáciles y difíciles. En el primer caso se afirma que pertenecen a la clase P, es decir, problemas que presentan un comportamiento polinomial; en el segundo caso NP, es decir, un comportamiento no polinomial. Un área de ciencia de la computación que determina a que clase pertenecen los problemas computacionales se denomina Teoría de la complejidad computacional. El objetivo de esta área es mediante el uso de técnicas de análisis determinar a qué clase pertenecen los problemas que son modelados con algoritmos iterativos o recursivos. Ver Tabla 2.1.

Tabla 2.1: Clases de problemas computacionales

n	$log\ n$	n	$n\ log\ n$	n^2	n^3	2^n
1	0.0	1.0	0.0	1.0	1.0	2.0
2	1.0	2.0	2.0	4.0	8.0	4.0
5	2.3	5.0	11.6	25.0	125.0	32.0
10	3.3	10.0	33.2	100.0	1000.0	1024.0
15	3.9	15.0	58.6	225.0	3375.0	32768.0
20	4.3	20.0	86.4	400.0	8000.0	1048576.0
30	4.9	30.0	147.2	900.0	27000.0	1073741824.0
40	5.3	40.0	212.9	1600.0	64000.0	1099511627776.0
50	5.6	50.0	282.2	2500.0	125000.0	1125899906842620.0
60	5.9	60.0	354.4	3600.0	216000.0	1152921504606850000.0
70	6.1	70.0	429.0	4900.0	343000.0	1180591620717410000000.0
80	6.3	80.0	505.8	6400.0	512000.0	1208925819614630000000000.0
90	6.5	90.0	584.3	9100.0	729000.0	...
100	6.6	100.	664.4	10000.0	1000000.0	...

Fuente: McConnell (2008).

La Tabla 2.1, en la primera fila, muestra las diferentes funciones de complejidad que se obtienen después del análisis de algún algoritmo. Se observa que algunas funciones son de tipo polinomial y casi polinomial. Ellas son consideradas funciones de complejidad para problemas de la clase P. La función 2^n, al igual que el $n!$ (no aparece en la Tabla) son consideradas funciones para la clase NP. La primera columna representa la cantidad de datos que serán usados por los algoritmos, de modo tal que las funciones de complejidad puedan reportar aproximadamente la cantidad de operaciones que tales algoritmos realizarán al ejecutarse. Estas informaciones muestran la importancia que tiene el análisis de un algoritmo tal como también lo resalta McConnell (2008). De este modo es posible elegir el mejor algoritmo para que sea implementado en etapa posterior.

Otros estudios acerca de la tasa de crecimiento de un algoritmo que puede ser O, Ω y θ, incluido el uso de teoremas que formalizan estos estudios, son discutidos en Cormen et al. (2009). La Figura 2.6 explica, por ejemplo, que asumiendo la existencia de un problema para el cual existen dos algoritmos cuyos comportamientos son dados por las funciones de complejidad $f(n)$ y $g(n)$, entonces el algoritmo representado por $f(n)$, al estar dominado asintóticamente por $g(n)$, realiza menos operaciones y por tanto será mas eficiente desde una cantidad de datos representado por n_0, debiendo ser escogido para que sea implementado.

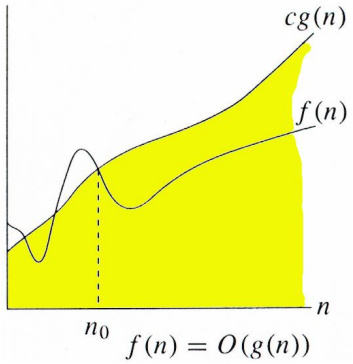

Figura 2.6: Notación O grande.
Fuente: Cormen et al. (2009)

Los problemas de la clase P serán solucionados por algoritmos tal que las soluciones obtenidas son exactas. Esto no sucede con aquellos problemas de la clase NP, pues al emplear mas tiempo de procesamiento para obtener soluciones exactas, se hace necesario el uso de nuevas estrategias algorítmicas, también denominadas metaheurísticas, tales como colonia de hormigas, algoritmos genéticos, busca tabú, *simulated annealing*, GRASP, entre otros. Estas nuevas formas de solucionar problemas computacionales difíciles, ayudan a obtener soluciones aproximadas. La Figura 2.7 muestra la clasificación de los métodos de solución existentes para problemas de optimización.

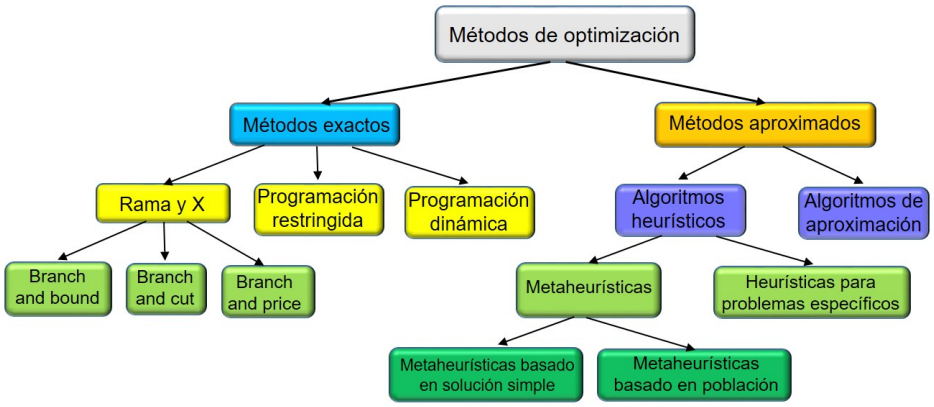

Figura 2.7: Métodos de solución para problemas de optimización.
Fuente: Talbi (2009)

2.1.3. Localización de facilidades

El transporte de personas o productos es una actividad que siempre va a existir, sin embargo también es cierto que es el responsable de las grandes emisiones de CO2, por ejemplo. Afortunadamente es posible reducir tal emisión, para ello es necesario efectuar decisiones en logística tales como cambiar el modo de transporte, ruta o la carga de los vehículos en la red.

El desempeño y la eficiencia del transporte en términos de costos y emisiones esta determinado por la red diseñada, así por ejemplo cuando se desea hacer las distribuciones de

productos, es fundamental tener la ubicación del respectivo centro de distribución o las fábricas donde se generan los productos. El problema de localización de facilidades (centros distribución o fábricas), de acuerdo con Farahani and Hekmatfar (2009) y Schneider and Drexl (2017), consiste en localizar un conjunto de facilidades, de modo tal que todas las demandas de los clientes sean asignadas a al menos una facilidad mientras se minimiza el costo de transporte. Una variación de este problema, según Bouchery et al. (2017), es la localización de facilidades verde, siendo su objetivo minimizar las emisiones de CO2, es decir, definir una cantidad y ubicación de las facilidades que atenderán a los clientes mientras se minimiza las emisiones de CO2.

El modelo de optimización para localización de facilidades, considerado NP-difícil, es dado en Daskin et al. (2005) tal como se describe a continuación.

(a) I: Conjunto de clientes i.

(b) J: Conjunto de j facilidades a ser localizados.

(c) d_i: Demanda del clientes i.

(d) f_j: Costo fijo de la facilidad que dara un servicio.

(e) c_{ij}: Costo o distancia facilidades y clientes.

(f) x_{ij}: Variable que representa la demanda del cliente i atendida por la facilidad j.

(g) y_j: Variable binaria donde el valor 1 indica que la facilidad fue localizada, caso contrario no fue localizada.

$$min\ Z = \sum_{j \in J} f_j Y_j + \sum_{j \in J, i \in I} d_i c_{ij} x_{ij} \tag{2.2}$$

s.t.

$$\sum_{j \in J} x_{ij} = 1, \forall i \in I \tag{2.3}$$

$$x_{ij} - y_j \leq 0, \forall i \in I, j \in J \tag{2.4}$$

$$y_j \in \{0, 1\}, \forall j \in J \tag{2.5}$$

$$x_{ij} \geq 0, \forall i \in I, j \in J \tag{2.6}$$

La función objetivo (2.2) optimiza los costos de localización de las facilidades y el costo de transporte según la demanda determinada por lo clientes. La condición (2.3) establece que cada cliente será atendido o asignado a alguna facilidad. En la restricción (2.4) un cliente no puede ser asignado a alguna facilidad, a menos que dicha facilidad este abierta y por tanto pueda dar un servicio al cliente asignado. Finalmente en (2.5) y (2.6) las declaraciones de las respectivas características de las variables.

Una experiencia computacional se realizó con la finalidad de conocer el funcionamiento del modelo propuesto por Daskin et al. (2005). Para tal efecto se considera un conjunto de cinco facilidades cuyas capacidades son de 100 unidades de peso y un conjunto de cuarenta clientes con sus respectivas demandas. Al finalizar el proceso computacional la facilidad f2 no fue seleccionada debido a la ubicación geográfica de los clientes quienes eligen las otras facilidades. Ver Figura 2.8 para análisis respectivo.

2.1.4. Ruteo de vehículos

El ruteo de vehículos es un problema combinatorio para el cual obtener su solución en grandes escenarios, implica mucho tiempo de proceso computacional. El problema consiste en que dado un grafo $G = (N, A)$, donde N es un conjunto de nodos interpretados como un conjunto de ciudades, casas, centros comerciales, computadoras, etc. y A un conjunto de arcos, que pueden ser pistas, cables, etc.; entonces se desea dar un servicio a cada nodo, con la condición de que un vehículo debe pasar por cada uno de ellos exactamente una sola vez, siendo que el proceso empieza y termina en un nodo denominado depósito. Ver Figura 2.9.

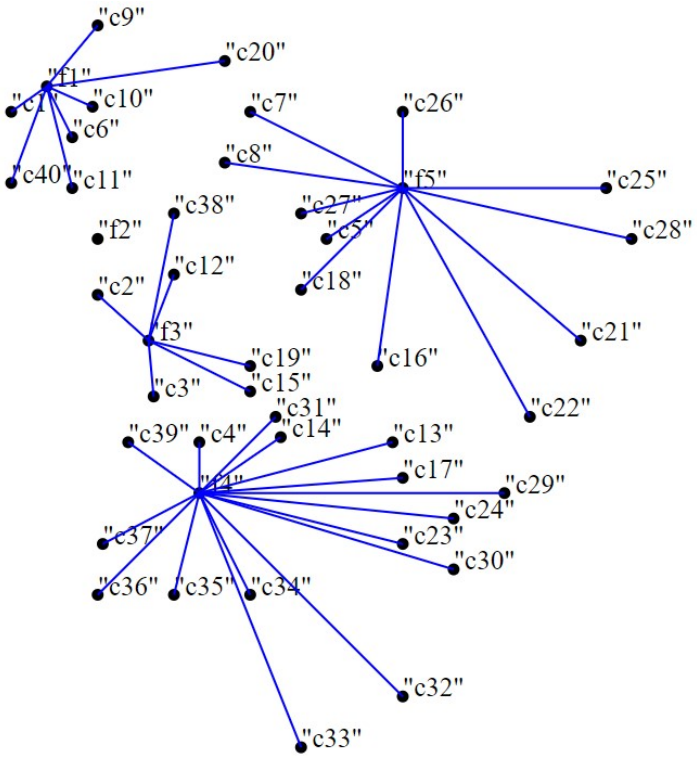

Figura 2.8: Localización de facilidades.
Fuente: Resultado computacional obtenido con glpk

Históricamente el problema de abastecer de combustible a los grifos, empezando desde un depósito central por medio de una flota homogénea fue solucionado por G. Dantzig y J. Ramser en 1959 tal como se muestra en Dantzig and Ramser (1959). Cinco años después en Clarke and Wright (1964) se generaliza este problema como un caso de optimización lineal, de modo tal que se pueda atender la demanda presentada por un conjunto de clientes ubicados en una zona geográfica alrededor de un depósito. Con esa publicación el tema se denomina ruteo de vehículos y encajaen logística y el transporte.

Con el transcurrir de los años el ruteo de vehículos y sus variantes fueron ganando mucho interés en la comunidad académica debido a que es posible incorporar situaciones de la vida real. Las diversidades de variantes tales como ruteo capacitado, ruteo con ventanas de tiempo, ruteo con colecta y entrega, ruteo con múltiples depósitos, ruteo abierto, entre otros hacen del tema mas complejo, pero igual de interesantes para solucionar problemas reales

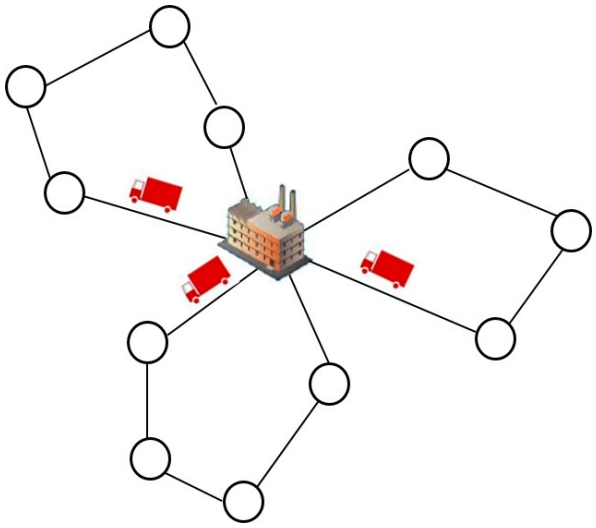

Figura 2.9: Ruteo de vehículos.
Fuente: Elaboración propia

son discutidos en Kumar and Panneerselvam (2012) y Braekers et al. (2016).

De acuerdo con Bouchery et al. (2017), ruteo de vehículos es una actividad ambientalmente intensa desde la perspectiva de la energía y GHG, pues durante el proceso de entrega de productos a los clientes se esta consumiendo combustible y por lo tanto genera emisiones que afectan al medio ambiente. Adicionalmente, como fue resaltado en Utama et al. (2020) se usa para la logística en las ciudades tal como abastecimiento de alimentos perecibles; además, también es aplicado a desastres naturales discutido en Gutiérrez et al. (2020). Estas propuestas muestran la importancia del ruteo de vehículos, ya que desde la óptica de la cadena de suministro no solo se debe optimizar costos económicos. También se debe mitigar costos ambientales protegiendo así el medio ambiente.

2.1.4.1. Modelos de optimización para ruteo de vehículos

Los problemas de ruteo mas populares y muy cerca a la realidad son ruteos con ventanas de tiempo presentado en El-Sherbeny (2010); ruteo capacitado en Liong Choong et al. (2008) y finalmente, ruteo con colecta y entrega. Las soluciones de estos tipos de ruteos se han realizado con métodos exactos tales como algoritmos exactos basados en las técnicas de

programación lineal y busca local guiada.

El clásico modelo de ruteo de vehículos se debe a la propuesta dada por Laporte (1992), quien basado en un grafo $G = (N, A)$, donde $N = \{1, \ldots, n\}$ es un conjunto de nodos (ciudades) con un depósito localizado en el nodo 1 y A un conjunto de arcos, donde para cada $(i, j), i \neq j$ se tiene asociado una distancia o tiempo de viaje o costo de viaje, expresado por c_{ij} y un conjunto de m vehículos disponibles; modela el problema que sigue a continuación.

$$min\ Z = \sum_{(i,j)\in A, i\neq j}^{n} c_{ij}x_{ij} \tag{2.7}$$

s.t.

$$\sum_{j\in N}^{n} x_{ij} = 1, \forall i \in N \tag{2.8}$$

$$\sum_{i\in N}^{n} x_{ij} = 1, \forall j \in N \tag{2.9}$$

$$\sum_{i\in N}^{n} x_{ij} \geq |S| - N(S), S : S \subset N - \{1\}, |S| \geq 2 \tag{2.10}$$

$$x_{ij} \in \{0, 1\}, \forall (i, j) \in A, i \neq j \tag{2.11}$$

La función objetivo (2.7) minimiza la distancia recorrida durante el proceso de ruteo, según las condiciones (2.8) y (2.9) que modelan el hecho de que a cada cliente le corresponde una atención; en (2.10) se garantiza la no formación de subcircuitos o subrecorridos. Finalmente (2.11) la variable de ruteo cuyos valores binarios ayudarán a formar el circuito solución del problema.

2.1.4.2. Modelo de ruteo de vehículos abierto

La logística empleada para el abastecimiento de los centros de atención al cliente, por ejemplo cadenas de supermercados, puede ser efectuada mediante las cadenas de suministro quienes con su propia flota de vehículos abastecen con diversos productos a tales centros. Sin embargo cuando ellas no cuentan con su propia flota de vehículos, entonces tendrá que

contratar los servicios de otra empresa dedicada al transporte. Esta real situación determina una nueva variante del ruteo denominada ruteo de vehículos abierto, es decir, que el proceso empieza en un depósito y termina cuando el último centro de atención fue atendido tal como se muestra en la Figura 2.10.

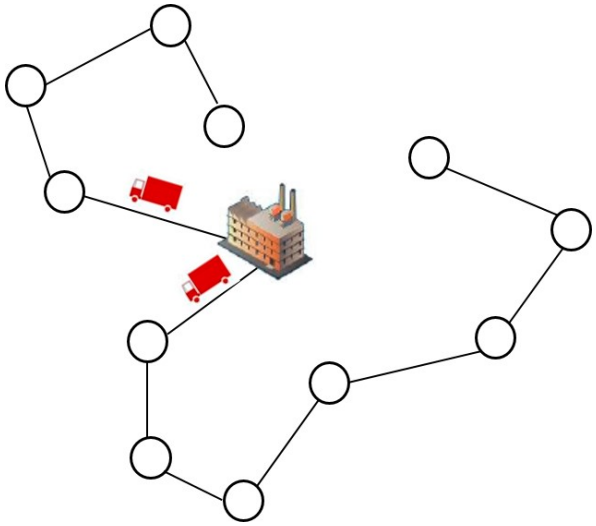

Figura 2.10: Ruteo de vehículos abierto.
Fuente: Elaboración propia

Un modelo basado en la metodología de programación lineal entera presentado por Pichka et al. (2014) establece que debido a la cantidad de vehículos limitado en cada depósito, se hace necesario un ruteo abierto. Como este tipo de problemas es NP, entonces los autores proponen la metaheurística *simulated annealing* para resolver casos de instancias medianas y de gran tamaño, ya que el modelo solo podrá mostrar soluciones para escenarios de tamaño pequeño.

Otra solución del ruteo de vehículos abierto mediante una metaheurística denominada colonia de la hormigas[3], fue presentado en GurpreetSingh and Dhir (2014) cuya propuesta se aplicó al caso de recoger escolares y volverlos a sus casas. Una nueva propuesta para solucionar el ruteo de vehículos abierto con restricciones de capacidad y distancia también

[3]Ant colony

fue presentado en Ruiz et al. (2019), cuya solución se obtiene mediante un algoritmo genético que minimiza la distancia total recorrida.

Brandão (2018) propone un modelo y un algoritmo de busca local iterada para el problema de ruteo de vehículos abierto con ventanas de tiempo, donde los resultados obtenidos usando un conjunto de *benchmarks* que contiene 418 instancias son considerados buenos. El modelo que se describe a continuación esta basado en las siguientes informaciones.

(a) $N = \{0, 1, \ldots, n\}$: Conjunto de nodos, donde 0 es depósito y n total de nodos.

(b) A: Conjunto de arcos de la red.

(c) d_{ij}: Distancia entre i y j, siendo $(i, j) \in A$

(d) u_i: Tiempo de descarga en el nodo $i \in N$, con $u_0 = 0$.

(e) t_{ij}: Tiempo de viaje entre i y j, mas u_i. Asumir que $d_{ij} = t_{ij} - u_i$.

(f) a_{ik}: Tiempo de atención al cliente i mediante el vehículo k.

(g) e_i: Tiempo inicial de la ventana de tiempo en $i \in N$.

(h) l_i: Tiempo final de la ventana de tiempo en $i \in N$.

(i) q_i: Demanda del nodo $i \in N$. Depósito no tiene demanda $q_o = 0$.

(j) m: Cantidad de vehículos disponibles en la flota homogénea.

(k) F: Costo fijo para el funcionamiento de cada vehículo.

(l) Q: Capacidad de cada vehículo.

(m) τ: Tiempo máximo que el chofer conduce el vehículo. Se asume que $\tau = i_0 - e_0$.

(n) p_{ij}: Denota la proximidad entre los nodos (clientes), donde $i, j \in N - \{0\}$.

(ñ) δ: Cantidad de vecindades cercanas, de acuerdo con la medida de proximidad definido por p_{ij}. Se puede validar que un nodo j esta en una δ-vecindad de i.

(o) x_{ijk}: Variable de decisión binaria, donde el valor 1 indica que el nodo j ha sido visitado por i mediante el vehículo k. Caso contrario será 0.

$$min\ Z = F\sum_{k=1}^{m} x_{0jk} + \sum_{k=1}^{m}\sum_{i=0}^{n}\sum_{j=0}^{n} d_{ij}x_{ijk} \tag{2.12}$$

s.t.

$$\sum_{k=1}^{m}\sum_{i=0}^{n} x_{ijk} = 1, \forall j = 1, 2, \ldots, n \tag{2.13}$$

$$\sum_{i=0}^{n} x_{irk} - \sum_{j=0}^{n} x_{rjk} = 0, \forall r = 1, \ldots, N; k = 1, \ldots, m \tag{2.14}$$

$$\sum_{j=1}^{n} x_{0jk} \leq 1, \forall k = 1, 2, \ldots, m \tag{2.15}$$

$$\sum_{i=1}^{n} x_{i0k} = 0, \forall k = 1, 2, \ldots, m \tag{2.16}$$

$$\sum_{i=0}^{n}\sum_{j=0}^{n} q_i x_{ijk} \leq Q, \forall k = 1, 2, \ldots, m \tag{2.17}$$

$$a_{ik} + t_{ij} - a_{jk} \leq (1 - x_{ijk})\tau, \forall i, j \in V; k = 1, 2, \ldots, m \tag{2.18}$$

$$e_i \leq a_{ik} \leq l_i, \forall i \in N; k = 1, 2, \ldots, m \tag{2.19}$$

$$x_{ijk} \in \{0, 1\}, \forall i, j \in N; k = 1, 2, \ldots, m \tag{2.20}$$

La función objetivo (2.12) garantiza que si para un valor apropiado o F es escogido, entonces una ruta óptima contribuye con la disminución de la distancia total recorrida. Las condiciones (2.13) y (2.14) establecen que cada cliente es visitado exactamente una sola vez y después de esta visita el vehículo se aleja en busca de otro cliente. En (2.15) modela el hecho de que cada vehículo es usado solamente una única vez. La restricción (2.16) garantiza que el vehículo después de haber visitado al último cliente, no retornará al depósito.

La restricción (2.17) garantiza que la demanda total generada por los clientes no excede la capacidad del respectivo vehículo. La no generación de subcircuitos queda garantizada en la restricción (2.18). Finalmente, las condiciones (2.19) y (2.20) aseguran que la atención al cliente sucede entre los límites establecidos en la ventana de tiempo y la variable de ruteo de tipo binario.

Recientemente Idzikowski (2020) presentó un estudio de caso para el ruteo de vehículos abierto capacitado con ventana de tiempo, donde la función objetivo esta dado por la suma de los tiempos de viaje de todos los vehículos, para tal efecto se tiene en cuenta las normas para el tráfico en las carreteras. El autor propone un modelo de optimización y para la solución mediante algoritmos presenta un algoritmo guloso y la metaheurística de busca tabú, donde al comparar los resultados, el autor llega a la conclusión que con la metaheurística se obtiene mejores resultados que el algoritmo guloso.

2.1.5. Metaheurísticas

Los problemas de optimización que existen en la vida real son diversos en las distintas áreas del conocimiento ya sea en la ciencia, ingeniería, logística y transporte, entre otros. Tales problemas por lo general son complejos y por lo tanto difíciles de resolver en forma exacta en tiempo computacional aceptable, sobre todo cuando se trabaja con escenarios que tienen cientos de datos (Talbi, 2009). La dificultad se debe a que estos problemas pertenecen a la clase NP, para los cuales es necesario el uso de algoritmos aproximados los cuales son de dos clases, el primero se denomina heurísticas específicas y el segundo metaheurísticas. Este último, según Talbi (2009), son una rama de la optimización en ciencia de la computación y matemática aplicada que esta relacionada con algoritmos y teoría de la complejidad computacional.

En Talbi (2009) se establece que las heurísticas dependen del problema a solucionar, es

decir, que están diseñados para ser aplicados a problemas particulares. La metaheurística es una estrategia algorítmica mas general y por tanto aplicable a una diversidad de problemas de optimización, que iterativamente explora el espacio de soluciones en busca de una respuesta para el problema planteado de forma eficiente, pero aproximada. Por lo tanto, su propósito principal es resolver grandes problemas los mas rápido posible obteniendo soluciones denominadas buenas.

Etimológicamente, la palabra heurística proviene del vocablo griego *heuriskein* que significa el arte de descubrir nuevas estrategias para la solución de problemas. La palabra meta significa metodología superior, por tal motivo, debido a (Glover, 1986) la palabra metaheurística se define como metodologías generales de nivel superior. Diversas áreas del conocimiento se han visto beneficiadas desde hace 30 años, Talbi (2009) y Amodeo et al. (2018), pues las metaheurísticas se han aplicado con éxito para solucionar problemas como:

(a) Dinámica de fluidos, telecomunicaciones, automotriz y robótica.

(b) Aprendizaje automático y minería de datos en bioinformática, biología computacional y finanzas.

(c) Modelado, simulación e identificación de sistemas en química, física y biología; procesamiento de control, señales e imágenes.

(d) Planificación de problemas de ruteo, problemas de horarios y producción, logística y transporte, gestión de la cadena de suministro, medio ambiente, etc.

(e) Red de sensores multimedia, selección de proveedores, embalaje de contenedores, seguimiento de objetos e identificación por radiofrecuencia.

Consideraciones a tener presente antes de diseñar una metaheurística, según Talbi (2009), se muestra en la Figura 2.11, donde el uso de una metaheurística depende de la complejidad

y dificultad del problema que se desea resolver, esto nos ubica en un caso de problemas NP para los cuales es necesario tener en cuenta el tiempo de busca de la solución dentro del respectivo espacio de busca.

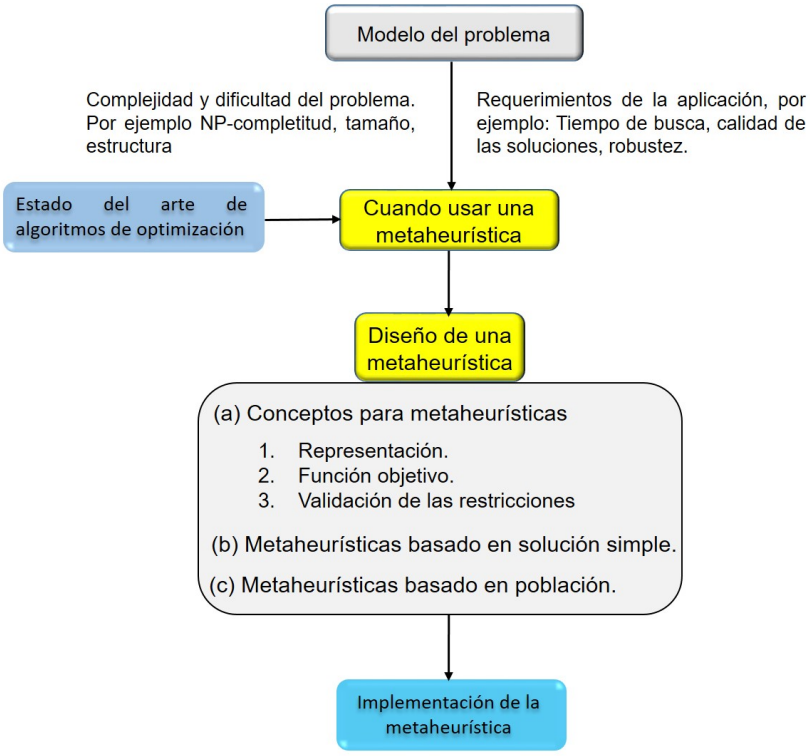

Figura 2.11: Consideraciones para diseñar una metaheurística.
Fuente: Talbi (2009)

Con esas consideraciones y una investigación del estado del arte, el siguiente paso es diseñar una metaheurística para lo cual se debe considerar:

(a) **Conceptos de metaheurísticas:**

Representación: Se refiere al uso de una codificación que puede lineal o no lineal u otras para representar las soluciones del problema. Es muy importante la codificación pues contribuye en la eficiencia y eficacia de una metaheurística para el problema de optimización a resolver.

Función objetivo: Guía la búsqueda de las soluciones ”buenas” en el espacio de búsqueda para obtener un máximo o mínimo.

Validación de restricciones: Las condiciones que se deben cumplir para llegar a la solución del problema es importante tenerlo presente en el diseño de una metaheurística. Las estrategias de manejo de las restricciones actúan sobre la forma como se representa las soluciones o la función objetivo.

(b) **Metaheurísticas basadas en soluciones simples:**

Este tipo de metaheurísticas se interpreta como dar un paseo por una vecindad o buscar trayectorias a través del espacio de busca de un problema dado. Se realiza mediante procedimientos iterativos con los cuales se obtienen soluciones cada vez mejores, aplicando procedimientos de generación y reemplazo a partir de una simple solución actual.

La fase de generación crea un conjunto $C(s)$ de soluciones candidatas, empezando desde la solución actual s, obtenidos por medio de transformaciones de la solución. En la fase reemplazo, la selección se ejecuta desde $C(s)$ para reemplazar la solución actual, es decir, una solución $s' \in C(s)$ es seleccionado como nueva mejor solución. El proceso continua hasta que se cumpla una condición de parada. Ver Figura 2.12 y algoritmo iterativo 1.

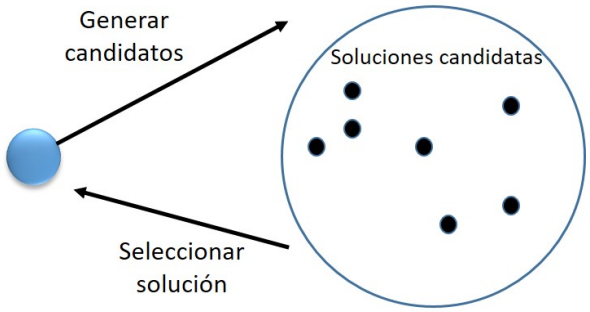

Figura 2.12: Principios para metaheurísticas basada en solución simple.
Fuente: Talbi (2009)

Algorithm 1 Busca solución

Require: Solución inicial s_0.
Ensure: Mejor solución encontrada.
$t = 0$
repeat
Genera $(C(s_t))$ ▷ Generar soluciones candidatas.
$s_{t+1} =$ Seleccionar$(C(s_t))$ ▷ Seleccionar sol. de $C(s)$ y reemplazar sol. actual s_t.
$t = t + 1$

until Que se cumpla el criterio de parada
return La mejor solución.

Vecindad se define como la función $F : S \rightarrow 2^S$ que asigna a cada solución $s \in S$ un conjunto de soluciones $F(s) \subset S$. La Figura 2.13 muestra como representar las vecindades.

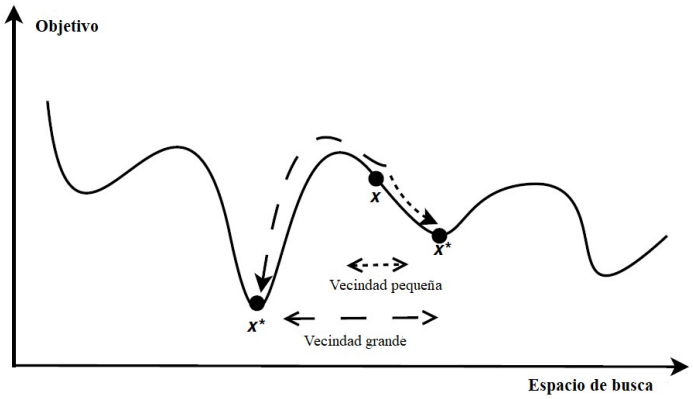

Figura 2.13: Tamaño de las vecindades en una busca local.
Fuente: Talbi (2009)

Tipos de metaheurísticas basada en solución simple conocidos son Busca local; *simulated annealing*; busca tabú; busca local iterada; busca en vecindad variable y busca local guiada.

(c) **Metaheurísticas basadas en poblaciones:**

Estos tipos de estrategias algorítmicas son considerados como la mejora iterativa en una población de soluciones, donde primero la población es inicializada, después se genera una nueva población y finalmente la nueva población es integrado en una solu-

ción actual por medio de algún procedimiento de selección. El proceso termina cuando se cumple con alguna condición o criterio de parada. La Figura 2.14 y el algoritmo 2 de forma general muestran el proceso para obtener soluciones basadas en poblaciones.

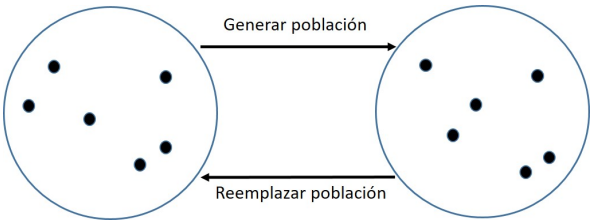

Figura 2.14: Principios para metaheurísticas basada en poblaciones.
Fuente: Talbi (2009)

Algorithm 2 Busca solución

Require: Solución inicial P_0.
Ensure: Mejor solución encontrada.

```
P = P_0                                   ▷ Generación de la población inicial.
t = 0
repeat
    Genera (P'_t)                         ▷ Generar nueva población.
    P_{t+1} = Seleccionar(P_t ∪ P'_t)     ▷ Seleccionar nueva población.
    t = t + 1

until Que se cumpla el criterio de parada return La mejor solución.
```

Ejemplos de metaheurísticas basadas en población son algoritmos genéticos, colonia de hormigas, colonia de abejas, sistemas inmune artificial, etc. Para fines de esta tesis hemos considerado la colonia de hormigas. Discusiones detalladas acerca de los diversos tipos de metaheurísticas son discutidas en Talbi (2009), Gendreau and Potvin (2019) y Kumar and Davim (2019). Finalmente como ya resaltado en la subsección 2.1.4, estas estrategias computacionales son usados para solucionar problemas de ruteo de vehículos por ser un problema NP.

2.1.6. Heurística 2-Opt

Debido a la naturaleza combinatoria y por tanto pertenece a clase de problemas NP-difícil, resolver el problema del agente viajero[4] mediante el paradigma (metodología) de algoritmos programación lineal entera, requiere mucho tiempo computacional para obtener un recorrido, incluido en escenarios de tamaño moderado, pues el proceso debe empezar y terminar en el mismo lugar y con la condición que cada ciudad es visitado exactamente una sola vez. Para tal efecto de debe realizar $\frac{(n-1)!}{2}$ operaciones, siendo n la cantidad de ciudades. Ver Figura 2.15.

Por ello el uso de un paradigma de aproximación a través de una metaheurística es necesario para obtener resultados en tiempo aceptable, sin embargo tales procedimientos algorítmicos al obtener una solución se detienen y no intentan mejorarla (Davendra, 2010). Se estima que estas estas estrategias algorítmicas obtienen soluciones en el rango del 10 $\%$ al 15 $\%$.

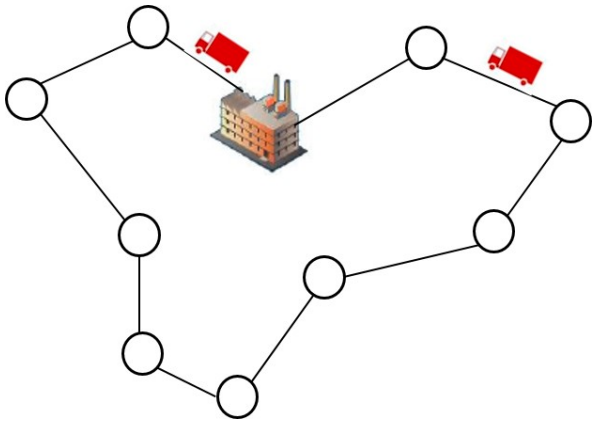

Figura 2.15: Recorrido en el problema del agente viajero.
Fuente: Elaboración propia

Estrategia para mejorar un recorrido es el denominado 2-Opt. El proceso consiste en retirar aleatoriamente dos arcos del recorrido obtenido hasta ese entonces y volver a formar otros dos nuevos arcos conectados a las ciudades, para obtener un nuevo recorrido. Ver Fi-

[4]Traveling salesman problem

gura 2.16. Esto es posible realizar solamente si el nuevo recorrido es más óptimo que en el anterior obtenido en etapa anterior. El proceso continua hasta que ya no sea posible obtener un recorrido mejorado.

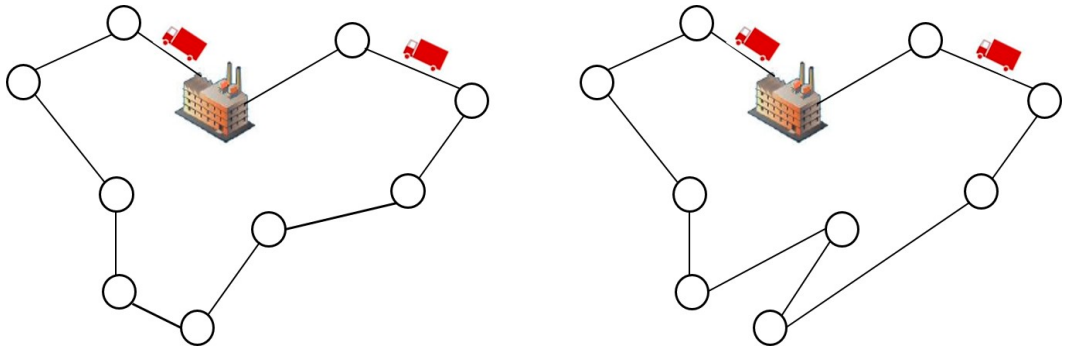

Figura 2.16: Recorrido original y recorrido mejorado.
Fuente: Elaboración propia

El siguiente pseudocódigo debido a Sathyan et al. (2015) muestra como se procede para obtener nuevas soluciones mejoradas para un problema del agente viajero.

Algorithm 3 Algoritmo 2-Opt

Require: n ciudades y sus localizaciones.
Ensure: Mejor solución encontrada.

```
Evaluar la matriz distancia
Definir un recorrido T inicializado aleatoriamente o obtenido mediante otro algoritmo
for i:= 1: n - 2 do
    for j:= i + 2: n do
        Evaluar d1 := longitud total de los dos arcos del recorrido original
        Evaluar d2 := longitud de los dos nuevos arcos del nuevo recorrido
        if d1 > d2 then
            Intercambiar los índices en el recorrido T para obtener un nuevo recorrido
        end if
    end for
end for
```

2.2. Método de la investigación

La investigación para esta tesis esta basada en el uso del método cuantitativo por cuanto se fundamenta en la medición de las características del fenómeno estudiado. Esto supone derivar de un marco conceptual al problema analizado de la distribución de productos y

su transporte en una red logística, es decir, se ha discutido una realidad susceptible de ser cuantificado mediante un modelo de optimización y una metaheurística.

Por lo tanto, la secuencia de pasos seguidos para alcanzar los objetivos trazados son los siguientes:

(a) Estudio de los sistemas logísticos urbanos, incluido la cadena de suministro.

(b) Modelado de los problemas de localización y ruteo de vehículos.

(c) Discusión de estrategias computacionales denominadas metaheurísticas y su uso en ruteo de vehículos según publicaciones en revistas indexadas.

(d) Aplicación de conocimientos teóricos para la solución del problema formulado en la presente tesis.

(e) Discusiones relacionadas con los resultados obtenidos.

(f) Consideraciones finales y referencias bibliográficas.

Capítulo 3

Localización de centros de distribución de productos y su transporte en una red logística

Basado en los conceptos discutidos en los dos primeros capítulos, en esta parte de la tesis se presenta y discuten un modelo de optimización lineal entera binaria y una metaheurística aplicada al tema investigado, teniendo como soporte una red logística donde los datos usados en diversos escenarios fueron prefabricados. De este modo modelo y metaheurística se testearon y validaron, cuyas estructura de implementación y las principales funciones se anexan al presente documento.

3.1. Modelo de optimización para localización-transporte

Planificación y modelamiento de un sistema de logística para que sea aplicado en una área urbana es una fase importante y estratégica. El modelo determina la localización de los centros de distribución (facilidades), así como también el flujo de productos que será transportado a lo largo de la red.

A continuación se presenta el modelo integrado que localiza centros de distribución y mediante un ruteo de vehículos abierto atiende a un conjunto de clientes, donde cada proveedor, cliente y centro de distribución tienen una ubicación geográfica expresada como un par ordenado (m, n), es decir, se optimiza minimizando el tiempo de atención a los clientes y la distancia recorrida por el vehículo seleccionado expresados en unidades de tiempo y longitud.

En relación a la disminución de la contaminación ambiental generado por el proceso de

transporte a través de los vehículos, fue posible ya que por definición de ruteo de vehículos cada cliente es atendido exactamente una vez, es decir, en el recorrido óptimo generado, los vehículos pasan por un arco de la red exactamente una vez en la zona donde se aplique el modelo.

(1) S: Conjunto de proveedores.

(2) I: Conjunto de clientes.

(3) J: Conjunto de centros de distribución.

(4) K: Conjunto de vehículos para el transporte de productos.

(5) SF: Red formado por proveedores y centros de distribución.

(6) P: Red formado por centros de distribución y clientes.

(7) $dist_{sj}$: Distancia entre proveedores $s \in S$ y centros de distribución $j \in J$.

(8) d_{ij}: Distancia entre centros de distribución $i \in P$ y clientes $j \in P$.

(9) t_j: Capacidad de los centros de distribución $j \in J$.

(10) $cost_{ij}$: Costo de transporte entre $i \in P$ y $j \in P$.

(11) ti_{ij}: Tiempo empleado durante el proceso de ruteo $i \in P$ y $j \in P$.

(12) f_j: Costo fijo para el funcionamiento del centro de distribución $j \in J$.

(13) cf_k: Costo fijo para operar la flota de vehículos $k \in K$.

(14) dd_i: Demanda del cliente $i \in I$.

(15) q_k: Capacidad del vehículo $k \in K$.

(16) g: Cantidad de vehículos.

(17) W_{sj}: Cantidad transportada de proveedor $s \in S$ al centro de distribución $j \in J$.

(18) X_j: Variable binaria que denota localización del centro de distribución $j \in J$.

(19) Y_{ij}: Variable binaria que indica demanda atendida por el centro de distribución $j \in J$ al cliente $i \in I$.

(20) Z_{ij}: Variable binaria que representa el proceso de ruteo de vehículos.

(21) U_{ij}: Variable que reporta la entrega de la carga antes de atender al siguiente cliente.

(22) M_k: Variable binaria que informa si el vehículo $k \in K$ esta de servicio.

(23) u_i: Variable que representa la cantidad de clientes $i \in P$ atendidos.

Con la definición de los conjuntos, parámetros y variables ya es posible formular el modelo de optimización basado en el paradigma de programación lineal entera binaria,

$$Min: \sum_{s\in S, j\in J} dist_{sj} W_{sj} + \sum_{j\in J} f_j X_j + \sum_{k\in K} cf_k M_k + \sum_{i\in P, j\in I} (d_{ij} + cost_{ij} + ti_{ij})\, Z_{ij} \quad (3.1)$$

s.t.

$$\sum_{s\in S} W_{sj} \leq t_j X_j, \forall j \in J \quad (3.2)$$

$$\sum_{s\in S} W_{sj} - \sum_{i\in I} dd_i Y_{ij} = 0, \forall j \in J \quad (3.3)$$

$$\sum_{j\in P} Z_{ij} \leq 1, \forall i \in P \quad (3.4)$$

$$\sum_{j\in P} Z_{ij} = 1, \forall i \in P \quad (3.5)$$

$$\sum_{j \in P} u_i - u_j + \mid P \mid Z_{ij} \leq \mid P \mid -1, \forall i, j \in P \tag{3.6}$$

$$\sum_{i \in P, j \in J} Z_{ij} \leq g \tag{3.7}$$

$$\sum_{i \in J, a \in I} Z_{ia} - \sum_{a \in I, j \in J} Z_{aj} = 0 \tag{3.8}$$

$$Z_{ij} \leq M_k, \forall i, j \in P, k \in K \tag{3.9}$$

$$\sum_{m \in P} Z_{im} + \sum_{n \in P} Z_{jn} - Y_{ij} \leq 1, \forall i \in I, j \in J \tag{3.10}$$

$$\sum_{j \in P} U_{ji} - \sum_{j \in P} U_{ij} = dd_i, \forall i \in I \tag{3.11}$$

$$U_{ij} \leq q_k Z_{ij}, \forall i \in P, j \in P, k \in K \tag{3.12}$$

La función objetivo (3.1) bajo ciertas condiciones viabiliza el recorrido en la red logística, localizando los centros de distribución a través de las variables X_j, $j \in J$ y el transporte de productos mediante la variable de ruteo Z_{ij}, $i \in P, j \in I$, expresados en unidades de distancia, monetarias y tiempo respectivamente. Para lograr minimizar esta función objetivo optimiza un grupo de sumatorias, donde la primera minimiza la distancia de atención de los centros de distribución mediante el respectivo proveedor localizado, en el segundo caso se optimiza el costo fijo para el funcionamiento de los respectivos centros de distribución.

De igual modo el tercer bloque de sumatorias optimiza el costo fijo para el funcionamiento de los vehículos que componen la flota. El cuarto bloque de sumatorias encuentra la

mejor ruta de entre todos aquellos recorridos candidatos según las distancias recorridas también optimiza el mejor recorrido en función al costo más apropiado. En este caso el costo de calculado en función a la distancia recorrida, es decir, $cost_{ij} = flete \times d_{ij}/50$, donde $flete$ es el precio a pagar por transportar productos según la distancia, d_{ij} a recorrer por cada 50 km. Además del tiempo empleado en el proceso.

La condición (3.2) modela el proceso en el que el proveedor puede abastecer al centro de distribución siempre que no se exceda la capacidad de tal centro y que esté en funcionamiento. En (3.3) el proveedor atiende al centro de distribución según demanda de la misma. Las restricciones (3.4) y (3.5) establecen que cada cliente tiene dos conexiones, es decir, una entrada a un cliente y otra salida en busca de otro cliente.

La restricción (3.6) evita la formación de subrecorridos en la solución final del proceso de atención a los clientes. La ecuación (3.7) modela la posible atención a los clientes siempre que se tenga vehículos disponibles. En (3.8) s tiene una condición de equilibrio, es decir, vehículo que llega a un cliente hace la entrega del pedido, y luego debe salir en busca del siguiente cliente para atender su demanda respectiva. La condición (3.9), asegura que los clientes serán atendidos por el respectivo vehículo.

En (3.10) garantiza que la demanda del cliente $i \in I$ es atendida por el centro de distribución $j \in J$. En la condición (3.11) se garantiza que la entrega de carga a un cliente sucede antes de atender al siguiente cliente, es decir, la demanda de todos los clientes es atendida por el respectivo centro de distribución. En (3.12) garantiza que ruteo del vehículo para atender a los clientes es posible, siempre que la carga del vehículo este con productos suficientes.

3.1.1. Aplicación del modelo localización-transporte

Con la finalidad de conocer el potencial del modelo de optimización propuesto en esta investigación, se realizó la implementación usando la herramienta computacional GLPK, eje-

cutándose sobre una Laptop HP, Procesador Intel CORE i7, con Sistema Operativo Window 10. A continuación se muestra un escenario ejemplo con datos prefabricados considerándolo como un caso hipotético.

Conjunto de proveedores:

s1, s2, s3

Conjunto de centros de distribución:

f1, f2, f3, f4

Conjunto de clientes :

c1, c2, c3, c4, c5

Conjunto de vehículos :

k1, k2

Ubicación geográfica de proveedores y centros de distribución:

s1 = (35,15) s2 = (100,65) s3 = (7,59)

f1 = (10,15) f2 = (20,50) f3 = (30,70) f4 = (55,105)

Capacidad de los centros de distribución:

f1 = f2 = f3 = f4 = 150

Ubicación geográfica de los centros de distribución y clientes:

f1 = (10,15) f2 = (20,50) f3 = (30,70) f4 = (55,105)

c1 = (3,25) c2 = (20,61) c3 = (31,81) c4 = (40,90) c5 = (80,50)

Demanda de los clientes:

c1 = 4 c2 = 3 c3 = 3 c4 = 1 c5 = 3

Costo fijo de instalacion para el funcionamiento de los centros de distribución:

f1 = f2 = f3 = f4 = 110

Capacidad de los vehículos:

k1 = k2 = 80

Costo fijo para el funcionamiento de los vehiculos:

k1 = k2 = 90

Con la información ingresada la implementación reporto la siguiente solución en un tiempo de 2 segundos de proceso computacional:

Proveedor	C. Distribución	Cant. producto a distribuir	Recorrido(km)
s3	f2	14	15.81

El proveedor s3 fue seleccionado para abastecer al centro de distribución f2 con una cantidad de 14 unidades de productos los mismos que fueron demandados por los clientes. La distancia que separa a s3 de f2 es 15.81 km. el cual es recorrido por el vehículo respectivo para abastecer al centro de distribución seleccionado.

Una vez abastecida el centro de distribución, se procede el ruteo de vehículos abierto el cual consiste en atender la demanda de los clientes con otro vehículo de menor tamaño. A continuación se muestra los resultados obtenidos en el proceso de ruteo, donde las dos primeras columnas representan la salida y llegada del vehículo de transporte. La tercera columna muestra que las cantidades de productos van disminuyendo, pues se esta haciendo la entrega respectiva al cliente. La cuarta columna el recorrido realizado en km. y finalmente el costo.

Cliente	Cliente	Cant. de producto distribuidos	Recorrido(km)	Costo
f2	c1	14	30.23	15.11
c1	c2	10	39.81	19.9
c2	c3	7	22.82	11.41
c3	c4	4	12.72	6.36
c4	c5	3	56.56	28.28

En detalle los resultados obtenidos, muestran que el vehículo sale del centro de distri-

bución f2 con un cargamento de 14 unidades de productos con la cual llega al cliente c1; al descargar 4 unidades en c1, el vehículo sigue su marcha hacia c2 llegando con 10 unidades. Atiende a c2 entregando 3 unidades de productos, para luego seguir el proceso de ruteo llegando al cliente c3 al cual llega con 7 unidades. Atiende a c3 y luego el vehículo continúa hacia c4 al cual llega con 4 unidades pues 3 unidades fueron entregados al cliente c3. Después de atender a c4 el vehículo de transporte llegará al cliente c5 con 3 unidades de productos a quien entregará la demanda de este cliente, con lo cual el vehículo queda vacío. Finalmente, el vehículo de transporte retorna a su depósito, pues ya entregó los productos demandados por los clientes. Ver Figura 3.1.

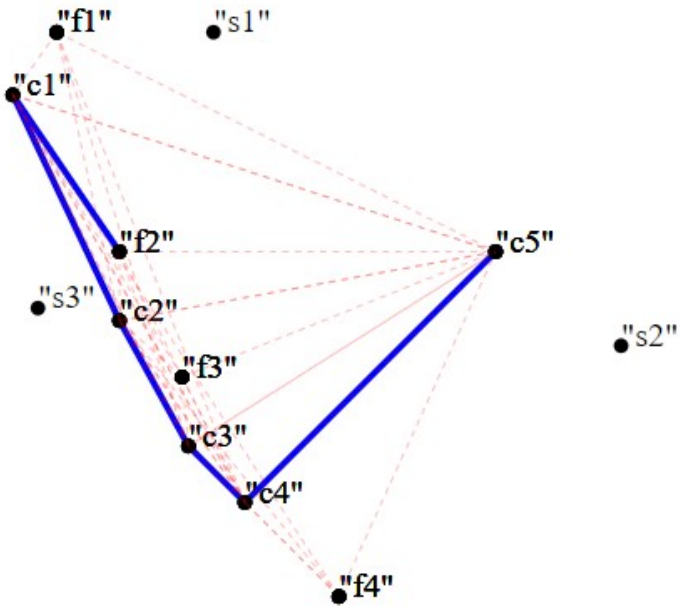

Figura 3.1: Localización del centro de distribución f2 y ruteo del escenario ejemplo .
Fuente: Resultado obtenido con glpk

La Tabla 3.1 es el resultado de haber implementado el modelo propuesto para seis escenarios. Para cada escenario se usaron los mismos conjuntos compuestos de tres proveedores, cuatro facilidades y dos vehículos de transporte con los cuales se hace la entrega de la demanda generada por los clientes, los cuales se van incrementando para cada escenario. Se observa que en la medida en que se incrementa la cantidad de clientes, aumenta el tiempo

(min.) del proceso de modo exponencial, esto se debe a que para hallar el recorrido óptimo, primero se deben testear todos los posibles recorridos para después decidir por aquel de menor costo. La complejidad computacional de n!.

La dificultad por encontrar recorridos óptimos para grandes escenarios en tiempos de proceso moderado, ha determinado la creación de nuevas estrategias algorítmicas, siendo una de ellas la que se discute en la siguiente sección.

Tabla 3.1: Resultados computacionales con el modelo propuesto para diversos escenarios

Escenario	Clientes	Vehículos	Cant. distribuida	Dist.(km.)	Costo	Tiempo
1	5	1	14	162.16	81.08	0.2
2	10	1	34	219.86	109.93	1.4
3	15	1	49	267.69	133.84	11.3
4	20	1	59	347.31	173.65	33.6
5	25	1	67	380.43	190.21	82.3
6	30	1	73	462.51	231.25	1320.0

Fuente: Elaboración propia

3.2. Metaheurística para localización-transporte basado en la colonia de hormigas

El modelo propuesto y su implementación en la sección 3.1 muestran que es posible resolver el problema de abastecer de productos en el sector urbano, cumpliendo con los principios de la logística, es decir, que dado un conjunto de centros de distribución (facilidad), los clientes potenciales pueden hacer una localización de tales centros. Una vez realizada esta etapa, se producen las demandas respectivas de productos los cuales, siguiendo un plan de rutas, son transportados hacia su respectivo destino. Previamente los centros son abastecidas por proveedores.

Por lo tanto, la contribución es de dos niveles; del proveedor al centro de distribución y de esta, aplicando ruteo de vehículos abierto, se transporta las demandas a los clientes de manera óptima. Sin embargo para grandes escenarios determinar el mejor ruteo implica

mucho tiempo de procesamiento, por ello se hace necesario una metaheurística que proporcione solución en menor tiempo de procesamiento, pero con resultados aproximados. Esto es discutido es la presente sección.

A continuación, en las tres subsecciones con sus respectivas descripciones o explicaciones de su funcionamiento, se presenta el algoritmo de colonia de hormigas el cual hace una llamada al algoritmo para generar rutas las cuales a su vez son mejoradas mediante el algoritmo 2-Opt.

3.2.1. Algoritmo de colonia de hormigas

El procedimiento que se presenta a continuación describe la lógica general para obtener un ruteo aplicando optimización mediante la metaheurística colonia de hormigas. Los parámetros que se ingresan al algoritmo son los siguientes:

(1) S: Conjunto de proveedores.

(2) I: Conjunto de facilidades.

(3) J: Conjunto de clientes.

(4) K: Conjunto de vehículos.

(5) SF: Red formada por proveedores y facilidades.

(6) P: Red conformada por facilidades y clientes.

(7) dist: Distancia entre proveedores y facilidades.

(8) d: Distancia entre facilidades y clientes.

(9) t: Capacidades de las facilidades.

(10) flete: Pago por transport.

(11) cost: Costo de transporte en la P.

(12) f: Costo fijo de instalación de las facilidades.

(13) cf: Costo fijo para el funcionamiento de un vehículo.

(14) dmax: Distancias máximas de los vehículos.

(15) dd: Demandas de los clientes.

(16) q: Capacidades de los vehículos.

(17) g: Cantidad de vehículos.

(18) W_{sj}: Cantidad transportada del proveedor a la facilidad.

(19) X_j: Variable binaria, donde $X = 1$ facilidad localizada, caso contrario no localizada.

(20) Y_{ij}: Variable que indica demanda atendida por la facilidad $j \in J$ al cliente $i \in I$.

(21) Z_{ij}: Variable binaria que representa el proceso de ruteo de vehículos.

(22) M_k: Variable binaria que informa si el vehículo $k \in K$ esta de servicio.

Algorithm 4 Colonia de hormigas

Require: S, I, J, K, SF, P, dist, d, t, flete, cost, f, cf, dmax, dd, q, g.
Ensure: RUTEO ÓPTIMO.

```
ruteo optimo:= Ninguno
if longitud(I) ≠ longitud(dd) then
    Escribir(Longitud de I es diferente de longitud de dd. Deben ser iguales.);
end if
if longitud(J) ≠ longitud(t) then
    Escribir(Longitud de J es diferente de longitud de t. Deben ser iguales.);
    if longitud(K) ≠ g then
        Escribir(Longitud de K es diferente del valor de g. Deben ser iguales.);
    end if
    if sumar(t.valores()) < sumar(dd.valores) then
        Escribir(suma de los valores de t no debe ser diferente de suma de valores de dd.);
    end if
    if longitud(K) ≠ longitud(dmax) then
        Escribir(Longitud de dmax es diferente de longitud de K. Deben ser iguales.);
    end if
    if flete < 0 then
        Escribir(Valor de flete debe ser un valor positivo.);
    end if
else
    ruteos:= [ ]; proveedores:= [ ]; costos obj:= [ ]; feromonas:= InicializarFeromonas(P)
    for iterador := 0 : iterador < 100 de 1 en 1 do
        W := { }
        for s en S do
            W[s]:= { }
            for j en J do
                w[s] [j]:= 0
            end for
        end for
        X := { }
        for j en J do
            X[j]:= 0
        end for
```

Algorithm 4 Colonia de hormigas(continua)

```
33:         Y := { }
34:         for i en I do
35:             Y[i]:= 0
36:         end for
37:         Z := { }
38:         for i en P do
39:             Z [i] := { }
40:             for j en P do
41:                 if i ≠ j then
42:                     Z [i] [j]:= 0
43:                 end if
44:             end for
45:         end for
46:         M := { }
47:         for k en K do
48:             M [k] := 0
49:         end for
50:         proveedor:= Ninguno; facilidad:= Ninguno; facilidades descartadas [ ]
51:         band:= Verdadero; ruta:= Ninguno; ruteo aco:= [ ]; ruteo bl:= [ ]
52:         proveedores elegidos:= [ ]; costos rutas aco:= { }; k:= 1; r:= 1
53:         while band = verdadero do
54:             if suma(M.valores()) < g then
55:                 if facilidad es ninguno then
56:                     facilidad:= elegirFacilidad(J, I, d, Z, Y, facilidades descartadas)
57:                     proveedor:= elegirProveedor(facilidad, S, dist)
58:                 end if
59:             end if
60:             if facilidad no es ninguno then
61:                 ruta, pedidos:= generarRuta(k, facilidad, Z, I, feromonas, q[K [k − 1]],
    dmax[K [k − 1]], t[facilidad], dd, d, Y )
62:                 if longitud(ruta) = 1 then
63:                     facilidades descartadas.agregar(facilidad)
64:                     facilidad:= Ninguno
65:                     Ir a While
66:                 end if
```

Algorithm 4 Colonia de hormigas(continua)

```
67:                 ◁ Verificación si aún queda espacio en la facilidad
68:                 if t[facilidad] - W[ proveedor] [facilidad] >pedidos then
69:                     W[proveedor] [ facilidad]:= W[proveedor] [ facilidad] + pedidos
70:                     X[facilidad] := 1
71:                 else
72:                     aux ruta:= ruta.copiar( )
73:                     for i:= longitud(aux ruta): i > 0 de -1 en -1 do
74:                         pedidos:= pedidos - dd[aux ruta [i]]
75:                         Y[aux ruta [i]] := 0
76:                         ruta.eliminar(aux ruta[i] )
77:                         if t[facilidad] - W[ proveedor] [facilidad] >pedidos then
78:                             W[proveedor] [ facilidad]:= W[proveedor] [ facilidad] + pedi-
    dos
79:                             X[facilidad] := 1
80:                             Salir
81:                         end if
82:                     end for
83:                     if longitud(ruta) = 1 then
84:                         facilidades descartadas.agregar(facilidad)
85:                         facilidad:= Ninguno
86:                         Ir a while
87:                     end if
88:                 end if
89:                 if t[facilidad] - W[ proveedor] [facilidad] = 0 then
90:                     facilidad = None
91:                 end if
92:                 ruteo aco.agregar(ruta)                    ▷ Agregar la ruta a ruteo aco.
93:                 proveedores elegidos.agregar(proveedor)        ▷ Agregar el proveedor
    elegido.
94:                 costos rutas aco[r]:= costoDeTransporteDeRuta(ruta, cost)    ▷ Costo de
    transp.
95:                 facilidades descartadas.limpiar( )          ▷ Borrar todas las facilidades
    descartadas.
96:                 ruta:= Ninguno                                          ▷ Reiniciar ruta
97:                 r:= r + 1                                  ▷ Incrementar el indice de ruta
98:                 M[(k,convertirACaracter(k))]:= 1
99:                 k:= k + 1
100:                Verificar si ya no quedan clientes por atender.
101:                if sumar(Y.valores( )) = longitud(I) then
102:                    band:= Falso
103:                end if
```

Algorithm 4 Colonia de hormigas(continua)

```
104:        else
105:            Como no se eligio facilidad, terminar while que generada rutas
106:            band:= Falso
107:        end if
108:        Como no quedan vehículos disponibles, terminar while que generaba rutas.
109:        ruteo aco.limpiar( )
110:        band:= Falso
111:        if longitud(ruteo aco) > 0 then
112:            costo obj aco:= funcionObjetivo(S, J, I, K, dist, d, cost, W, Z, f, X, cf,
    M)
113:            Z2:= Ninguno
114:            costos rutas bl:= Ninguno
115:            costo obj bl:= Ninguno
116:            band:= Falso
117:            Ejecutar algoritmo 2-opt búsqueda local
118:            ruteo inicial:= ruteo aco
119:            costo obj inicial:= costo obj aco
120:            while band ≠ Verdadero do
121:                Z2:= { }
122:                while cada i en P do
123:                    Z2[i]:= { }
124:                    while cada j en P do
125:                        if i ≠ j then
126:                            Z2[i] [j]:= 0
127:                        end if
128:                    end while
129:                end while
130:                [ruteo bl, costos rutas bl]:= busquedaLocal2opt(ruteo inicial, cost,
    Z2)
131:                costo obj final:= funcionObjetivo(S, J, I, K, dist,d, cost,W,Z2, f,X,
    cf,M)
132:                if valor absoluto(costo obj final - costo obj inicial) = 0 then
133:                    costo obj bl:= costo obj final
134:                    band:=True
135:                else
136:                    costo obj inicial:= costo obj final
137:                    ruteo inicial:= ruteo bl
138:                end if
139:            end while
140:            proveedores.agregar(proveedores elegidos)
```

3.2.1.1. Descripción del algoritmo de colonia de hormigas

Asumiendo el caso de que se cumplen todas las condiciones impuestas en el algoritmo, se declaran e inicializan las variables relacionadas al ruteo, abastecimiento por parte de los proveedores a las facilidades, los costos obj y feromonas. De este momento, se inicia el bucle principal y la decisión de seguir o cancelar depende de la variable ”iterador” comenzando con el valor 0 hasta llegar al número máximo de iteraciones menos uno.

Dentro del bucle o por cada iteración, se inicializan las variables W, X, Y, Z y M. Además, se declaran e inicializa las variables ”facilidad” (facilidad elegida), ”proveedor” (proveedor elegido), ”facilidades descartadas”, ”band”, ”ruta”, ”ruteo aco”, ”ruteo bl”, ”proveedores elegidos”, ”costos rutas aco”, ”k” (k-ésimo vehículo) y ”r” (r-ésima ruta). El siguiente paso es generar el ruteo por medio del algoritmo lanzando una hormiga artificial.

El bucle interno, *while*, es la primera parte del algoritmo y es en aquella donde se generará la solución inicial del ruteo empleando colonia de hormigas. La primera instrucción es la evaluación de la disponibilidad de vehículos. Si no hubiesen vehículos disponibles, la ruta inicial es nula y en consecuencia no habrá ruteo; caso contrario, se continúa con las siguientes instrucciones. Para seguir el flujo, la siguiente instrucción es evaluar la existencia de una facilidad elegida. Si no lo hubiera, se elige una facilidad que no sea descartada y también se elige el proveedor más cercano a esta. Posteriormente, se verifica nuevamente la existencia de la facilidad elegida. Si no existe, se le asigna un valor de Falso a ”band” para cancelar el bucle que genera la ruta; caso contrario, se prosigue con las siguientes instrucciones.

Asumiendo la existencia de la facilidad elegida, generar la ruta empleando el algoritmo generarRuta. Luego, se verifica la cantidad de nodos de la ruta. Si esta cantidad o longitud de la ruta es 1, la facilidad elegida se anula y se descarta para no ser elegida y se salta al siguiente ciclo del bucle *while*.

Algorithm 4 Colonia de hormigas(continua)

```
141:                    if costo obj bl < costo obj aco then
142:                        costos obj.agregar(costo obj bl)
143:                        ruteos.agregar(ruteo bl)
144:                        actualizarFeromonas(Z2, feromonas, costos rutas bl)
145:                    else
146:                        costos obj.agregar(costo obj aco)
147:                        ruteos.agregar(ruteo aco)
148:                        actualizarFeromonas(Z, feromonas, costos rutas aco)
149:                    end if
150:                end if
151:            end while
152:        end for
153:        if longitud(ruteos) > 0 then
154:            Hallar el mínimo costo obj, reorganizar los resultados y devolver el ruteo optimo.
155:            minimo objetivo:= minimo(costos obj)
156:            ruteo:= ruteos[ costo obj.index(minimo objetivo)]
157:            proveedores:= proveedores[ costos obj.index(minimo objetivo)]
158:            ruteo optimo:= [ ]
159:            ruteo optimo.agregar({ })
160:            ruteo optimo.agregar(minimo objetivo)
161:            for i:= 0 : i < longitud(proveedores) de 1 en 1 do
162:                s:= proveedores[i]
163:                if Si s no está en list(ruteo optimo[0].claves()) then
164:                    ruteo optimo[0] [ s]:= { }
165:                end if
166:                ruta:= ruteo[i]
167:                if ruta[0] no está en list(ruteo optimo[0] [s].claves( )) then
168:                    ruteo optimo[0] [s] [ruta [0]]:= [ ]
169:                end if
170:                lista:= [ concatenar(k,convertirACaracter(i + 1)), ruta [1 hasta longitud(ruta) - 1]]
171:                ruteo optimo[0] [s] [ruta [0]].agregar(lista)
172:            end for
173:        else
174:            Retornar Ninguno
175:        end if
176:    end if
```

Si la ruta tiene al menos dos nodos, el siguiente paso es verificar si queda espacio en la facilidad elegida para agregar un pedido en la lista W de pedidos de la facilidad que posteriormente realizará a un proveedor. Si queda espacio, se asigna 1 a una región de X con respecto a la facilidad para indicar su instalación y se agrega el pedido a su lista. Si no se da el caso, entonces se procede a retirar clientes de la ruta y verificar si queda espacio y así sucesivamente hasta lograr agregar el pedido a la lista W. Después verificar si la longitud de la ruta modificada es igual a 1, descartar la facilidad y anularla si esto es cierto y saltar al siguiente ciclo del bucle *while*.

Posteriormente, si se da el caso de que se pudo agregar el pedido a la lista W con respecto a la facilidad elegida y/o que su longitud es mayor que 1, se verifica el agotamiento de capacidad de la facilidad. Si se da este caso, ya no podrá servir en las próximas rutas generadas. Luego, se agrega la ruta a la lista ”ruteo aco” así como el proveedor elegido a la lista de proveedores elegidos. También se calcula el costo de transporte de toda la ruta y se agrega a la lista ”costos rutas aco”, se limpia la lista de facilidades descartadas, se reinicia la variable ”ruta” al vacío, la variable ”r” se incrementa para indicar otra ruta al igual que ”k” para indicar el siguiente vehículo.

Finalmente, para terminar la parte del algoritmo de colonia de hormigas, se verifica la cantidad de clientes atendidos. Si todos los clientes han sido atendidos, cancelar el bucle *while*. El siguiente paso es mejorar el ruteo mediante el algoritmo de búsqueda local con dos opciones también denominado 2-Opt.

La segunda parte es mejor la solución inicial obtenida con el algoritmo de colonia de hormigas. Si existe el ruteo inicial, se inicializan las variables necesarias para esta parte como Z2, variable de ruteo alternativa a Z, que será utilizada para la solución con búsqueda local 2-Opt. Teniendo un ruteo inicial y un costo inicial obtenido con colonia de hormigas,

inicializar el ciclo del bucle *while*; donde en una iteración, se obtienen ruta mejorada, costo final y se evalúan la diferencia entre entre costo inicial y final. Si la diferencia es mayor que 0, el ruteo mejorado se convierte en inicial y el costo final se convierte en costo inicial. Luego ejecutar la siguiente iteración hasta que la diferencia de costos sea 0 terminándose la fase de mejoramiento de la solución final.

Después, se comparan los costos del ruteo mejorado y del ruteo inicial. Si el primero es menor, se almacena el ruteo mejorado y su costo en las listas de ruteo y costos objetivos respectivamente; actualizándose la matriz de feromonas teniendo en cuenta la variable Z2. De otro modo, se realiza el mismo procedimiento, para el ruteo inicial y se actualiza la matriz de feromonas teniendo en cuenta la variable Z. Seguidamente, continuar con la siguiente iteración.

3.2.2. Algoritmo para generar rutas

La etapa crucial de la propuesta consiste en generar rutas de modo a optimizar el proceso. A continuación se presenta el respectivo algoritmo en pseudocódigo empleado por la metaheurística de optimización por medio de colonia de hormigas el cual usa las siguientes notaciones.

(1) HK: k ésima hormiga.

(2) NIP: Nodo inicial de la red P formada por facilidades y clientes.

(3) Z: Variable de ruteo.

(4) I: Conjunto de clientes.

(5) FM: Colección clave-valor de feromonas de los caminos.

(6) hq: Capacidad de almacenamiento de pedidos de la hormiga.

(7) DMH: Distancia máxima de recorrido de la hormiga.

(8) tj: Capacidad de almacenamiento de una facilidad j.

(9) dd: Colección clave-valor de las demandas de los clientes.

(10) DP: Colección clave-valor de las distancias de caminos de la red P.

(11) Y: Variable tipo colección clave-valor de los clientes atendidos.

(12) ”Parada”: Variable de decisión de tipo booleano que controla el funcionamiento del algoritmo permitiendo su ejecución y parada según la condición que se especifique.

(13) ”Temp”: Variable denominada temporal usada para el cálculo de la probabilidad para decidir qué nodo o cliente se toma como destino inmediato.

(14) ”Nodo actual”: Variable utilizada para apuntar a un nodo actual, empezando del nodo inicial ingresado como parámetro y luego apuntando a los últimos nodos que pertenecen a la ruta.

(15) ”Nodo siguiente”: Variable que apunta al posible nodo destino inmediato al nodo actual. Nodo apuntado por esta variable es evaluado para determinar si debe pertenecer a la ruta que se está generando.

(16) ”Dist. acumulada”: Variable empleada para acumular las distancias recorridas por la hormiga y luego se compara con su distancia máxima.

(17) ”Demanda acumulada”: Variable utilizada para acumular las demandas de los clientes que asignan a la ruta y luego se compara con la capacidad de carga de la hormiga.

(18) ”Ruta”: Variable que apunta a la ruta que se va generando durante la ejecución del algoritmo.

(19) ”Pedidos acumulados”: Variable que contiene cantidad de pedidos o demandas acumulados según la ruta generada, y se compara con la capacidad de la facilidad en un análisis externo.

Algorithm 5 Generar rutas

Require: HK, NIP, Z, I, FM, hq, DMH, tj, dd, DP, Y.
Ensure: RUTA ÓPTIMA: (ruta, pedidos acumulados).

parada:= Falso
Temp:= Ninguno
Nodo actual:= NIP
Nodo siguiente:= Ninguno
Dist. acumulada:= 0
Demanda acumulada:= 0
Ruta:= NIP ▷ Lista de nodos que conformarán la ruta
Pedidos acumulados:= 0
while Parada es falso **do**
    Nodos destinos:= Obtener nodos destinos (Z, I, Nodo actual)
    if Longitud (Nodos destinos) > 0 **then**
        Temp:= Nuevo vector vacio (Longitud (Nodos destinos))
        for i := 0 : i $<$ Longitud(Nodos destinos) **do**
            f:= FM(Nodo actual, Nodos destinos$[i]$) ▷ f: FM
            v:= $1/$DP(Nodo actual, Nodos destinos$[i]$) ▷ v: Visibilidad
            FM $*$ Visibilidad;
            Temp$[i]$:= f$*$ v
        end for
        Suma:= Sumar(Temp) ▷ Sumar todos los valores del vector temporal Temp
        for i:= 1 : i $<$ Longitud(Nodos destinos) **do**
            Temp$[i]$: Temp$[i]/$ Suma
        end for
        Suma:= 0;
        Numero aleatorio:= Generar numero aleatorio( ) ▷ Generar número aleatorio
        for i := 0 : i $<$ Longitud(Nodos destinos) **do**
            if Numero aleatorio $\leq$ Suma + Temp$[i]$ **then**
                Nodo siguiente:= Nodos destinos$[i]$; Detener PARA
            else
                Suma:= Suma + Temp$[i]$
            end if
        end for
        Dist. acumulada:= Dist. acumulada + DP[Nodo actual] [Nodo siguiente]
        Demanda acumulada:= Demanda acumulada + dd [Nodo siguiente]
        if Dist.acumulada $\leq$ DMH; Demanda acumulada $\leq$ hq; Demanda acumulada $\leq$ tj **then**
            Ruta.AgregarNodo(Nodo siguiente) ▷ Nodo siguiente pasa a ser parte de la ruta

Algorithm 5 Generar rutas (continua)

```
36:             Pedidos acumulados:= Pedidos acumulados + dd[ Nodo siguiente ]
        ▷ Camino (Nodo actual, Nodo siguiente) se activa en Z con el número de hormiga
    > 0
37:             Z[ Nodo actual ] [ Nodo siguiente ] := HK
38:             Nodo actual := Nodo siguiente    ▷ Nos situamos ahora en el nodo siguiente
39:             Y[ Nodo actual ]: = 1                    ▷ Se activa el cliente "Nodo actual"
40:         else
41:             Parada:= Verdadero
42:         end if
43:     else
44:         Parada:= Verdadero
45:     end if
46: end while
```

3.2.2.1. Descripción del algoritmo generador de rutas

El proceso comienza con el ingreso al bucle permitido por la variable "Parada"con valor verdadero. La primera instrucción en el bucle determina el grupo de nodos destinos posibles desde un nodo origen o nodo actual. En esta instrucción se invoca a una función llamada "obtenerNodosDestinos" y los parámetros enviados son la variable de ruteo Z, el conjunto de clientes I y el nodo actual indicado por "Nodo actual". Luego, se evalúa la cantidad de nodos destinos obtenido anteriormente mediante la condición "Longitud(nodos destinos) $>$ 0". Si la condición es verdadera, la ejecución continúa, caso contrario asignar falso a la variable "Parada" finalizando el bucle.

Asumiendo que la condición mencionada sea verdadera, la siguiente instrucción creará un espacio en memoria para la variable "Temp", tomando en cuenta la longitud del grupo de nodos destinos posibles. Después de terminar esta instrucción, el siguiente paso es ejecutar el primer bucle interno en el cuál se procede a encontrar la feromona entre el nodo actual y un nodo del grupo de nodos destino, se calcula la visibilidad entre los nodos mediante el inverso multiplicativo de la distancia entre estos dos.

$$\mathit{Temp}_k = f_{ij}^{\alpha} \times v_{ij}^{\beta}, \forall k \in [0, \mathit{long}(\mathit{Nodos\ destino})] \wedge i = \mathit{Nodo\ actual} \wedge j \in \mathit{Nodos\ destinos}$$

El valor de α (influencia sobre la feromona) y β (influencia sobre la visibilidad) para el algoritmo es 1. Posteriormente, al salir del bucle, se suman todos los valores del arreglo apuntado por "Temp". Luego, en el segundo bucle interno, se procede a calcular la probabilidad por cada valor de "Temp" dividido entre la suma de los valores del mismo.

$$Probabilidad_k = \frac{Temp_k}{\Sigma_{i=0}^{k-1} Temp_i}$$

La probabilidad k-ésima calculada reemplaza al valor almacenado en la k-ésima región de "Temp" y después de realizar todo el procesamiento del segundo bucle interno, "Temp" deberá contener todas las k probabilidades calculadas.

$$Temp_k := Probabilidad_k$$

Luego, se reinicia el valor de la suma en 0 y se calcula el número aleatorio entre 0 y 1. El siguiente paso es ingresar al tercer bucle interno para determinar a qué k-ésima región del espacio entre 0 y 1 pertenece el número aleatorio.

$$Región_k = [suma, suma + Temp_i]$$

Partiendo de $suma = 0$, la primera región es:

$$suma = 0 \rightarrow Región_0 = [0, Temp_0]$$

Las siguientes regiones se obtienen de la siguiente forma:

$$suma = suma_{anterior} + Temp_0 \rightarrow Región_1 = [suma, suma + Temp_1]$$

$$suma = suma_{anterior} + Temp_1 \rightarrow Región_2 = [suma, suma + Temp_2]$$

$$suma = suma_{anterior} + Temp_2 \rightarrow Región_3 = [suma, suma + Temp_3]$$

...

$$suma = suma_{anterior} + Temp_{i-1} \rightarrow Región_k = [suma, suma + Temp_i]$$

Para saber a qué región pertenece el número aleatorio, realizar una comparación de dicho número con el valor máximo del intervalo k (k ésima región), es decir:

$$número\ aleatorio \leq suma + Temp_i$$

Si la condición es falsa, seguir recorriendo las regiones hasta encontrar una que satisfaga la condición y se obtenga un nodo destino i. Cuando el resultado de la comparación es verdadero, se obtiene el nodo destino y se cancela el bucle interno para pasar a las siguientes instrucciones.

El siguiente paso es la acumulación de distancias recorridas y demandas recogidas. Posteriormente, se llega al paso final con la condición que permitirá decidir si el nodo destino seleccionado debe o no pertenecer a la ruta en proceso de generación. Se verifica si la distancia acumulada es menor o igual que la distancia máxima de la hormiga y que la demanda acumulada es mejor o igual que la capacidad de carga de la hormiga.

Si el resultado de estas comparaciones es verdadero, el nodo destino es agregado a la ruta, la demanda del nodo es añadida a los pedidos acumulados de la ruta, el camino entre el nodo actual y el nodo destino es indicado por el número de la hormiga en la variable de ruteo Z, el nodo siguiente se convierte en nodo actual, su atención es indicada con el valor 1 en la variable Y de atención de clientes y como la variable ”Parada” aún tiene el valor de verdadero, se retorna a la primera instrucción del bucle y se ejecuta el mismo procedimiento para tratar de obtener otro nodo destino.

Cuando la condición de parada del paso final da como resultado un valor falso, el nodo destino seleccionado es anulado, el valor de falso es asignado a la variable ”Parada” para cancelar el bucle principal y devolver como resultado del algoritmo una lista compuesta por

la ruta generada y los pedidos acumulados en toda la ruta.

3.2.3. Algoritmo de busca local 2-Opt

Para obtener un mejor ruteo, es decir, con el costo mínimo a continuación se presenta la heurística de búsqueda local 2-Opt. Ver algoritmo 6.

Algorithm 6 Optimización de rutas 2-Opt

Require: ruteo aco, costos p, Z.
Ensure: Rutas optimizadas.

```
1:  ruteo bl:= ruteo aco.copiar()
2:  costos rutas bl:= { }
3:  r:= 1
4:  for cada ruta en ruteo bl do
5:      min cambio:= 0
6:      min i:= 0
7:      min j:= 0
8:      for i:= 0 : i < (longitud(ruta) – 2) de 1 en 1 do
9:          for j:= (i + 2) : j < (longitud(ruta) – 1) de 1 en 1 do
10:             costo actual:= costos p[ruta[i]] [ruta[i + 1]] + costos p[ruta[j]] [ruta[j + 1]]
11:             costo nuevo:= costos p[ruta[i]] [ruta[j]] + costos p[ruta[i + 1]] [ruta[j + 1]]
12:             cambio:= costo nuevo - costo actual
13:             if cambio < min cambio then
14:                 min cambio:= cambio
15:                 min i:= i
16:                 min j:= j
17:             end if
18:         end for
19:     end for
20:     if min cambio < 0 then
21:         ruta[min i + 1 :min j + 1]:= ruta[min i + 1: min j + 1] [:: −1]
22:     end if
23:     for Para i:= 0 : i < (longitud(ruta) – 1) de 1 en 1 do
24:         Z[ruta[i]] [ruta[i + 1]]:= r
25:     end for
26:     costos rutas bl[r]:= costoDeTransporteDeRuta(ruta, costos p)
27:     r:= r + 1
28: end for
29: Retornar [ruteo bl, costos rutas bl].
```

3.2.3.1. Descripción del algoritmo de busca local 2Opt

El algoritmo de búsqueda local de dos opciones (2-Opt) ayuda a mejorar una ruta eliminando los cruces sobre sí misma. Los parámetros que recibe son el ruteo con colonia de

hormigas, costos de las rutas y la variable de ruteo. El ruteo con búsqueda local se inicializa con la copia del ruteo con colonia de hormigas y la nueva lista de costos se inicializa como vacía.

La mejora del ruteo se procede ruta por ruta, donde para cada ciclo del bucle principal se emplean las variables ”min cambio” (cambio mínimo), ”min i” y ”min j”. El primer paso para mejorar una ruta es tomar dos nodos $(ruta_i, ruta_j)$ no consecutivos $(ruta_k, ruta_{k+2})$, respectivamente y calcular los costos de la siguiente manera:

$$\text{costo actual} = costosP_{ruta_i, ruta_{i+1}} + costosP_{ruta_j, ruta_{j+1}}$$

$$\text{costo nuevo} = costosP_{ruta_i, ruta_j} + costosP_{ruta_{i+1}, ruta_{j+1}}$$

Luego, se calcula la diferencia de estos costos a la cual se le denomina ”cambio”.

$$\text{cambio} = \text{costo nuevo} - \text{costo actual}$$

Después se compara si el cambio es menor que el cambio mínimo. Si el resultado de la condición es verdadero, entonces el cambio mínimo es el cambio calculado por la diferencia de costos y los valores de i y j se convierten en índices mínimo y nuevamente se procede con el siguiente ciclo hasta que el cambio sea igual que el cambio mínimo.

Posteriormente, se evalúa si el cambio mínimo es negativo. Si esto resulta ser cierto, entonces se produce el intercambio de posiciones de los nodos en la ruta deshaciendo los cruces sobre sí misma y tratando de minimizar su costo. La siguiente instrucción coloca el número de la ruta en los caminos que forman parte de ella. La próxima instrucción calculará el costo de transporte de la ruta y por último el cambio de número de ruta para pasar a la siguiente hasta terminar de mejorar todas las rutas pertenecientes al ruteo.

Finalmente, se procede a devolver una lista que ha de contener tanto el ruteo mejorado denominado ”ruteo bl” así como los costos de transporte de sus rutas.

3.2.4. Aplicación de la metaheurística propuesta

Considerando la misma información usada para testear el modelo de optimización propuesto y con el mismo equipo informático, sección 3.1.1., a continuación se presentan los resultados obtenidos cuando se implementó la metaheurística basado en colonia de hormigas usando el lenguaje de programación Python, al cual se ha agregado cuatro nuevos escenarios haciendo un total de 50 clientes quienes generan la respectiva demanda de productos.

La Tabla 3.2 muestra que en cada escenario el tiempo de procesamiento, en minutos, empleado para localizar centros de distribución y atención a los clientes se incrementa gradualmente, a pesar de que desde el escenario 7 la cantidad de vehículos se ha aumentado. El incremento del tiempo de proceso empleado por la metaheurística propuesta, se debe a que la cantidad de clientes ha aumentado siendo que para determinar una solución se tuvieron que realizar una cantidad grande de posibilidades, es decir, siendo n la cantidad de clientes se tiene que realizar $\frac{(n-1)!}{2}$ operaciones para hallar igual cantidad de rutas candidatas, para finalmente decidir por una de ellas.

Tabla 3.2: Resultados computacionales con la metaheurística para diversos escenarios

Escenario	Clientes	Vehículos	Cant. distribuida	Dist.(km)	Costo	Tiempo
1	5	1	14	162.5	81.08	0.5
2	10	1	34	219.86	109.93	1.4
3	15	1	49	267.7	133.85	3.8
4	20	1	59	347.32	173.66	6.4
5	25	1	67	385.1	192.55	10.4
6	30	1	73	551.93	275.97	13.7
7	35	3	165	883.46	441.73	18.3
8	40	3	185	839.16	419.58	22.5
9	45	4	222	1181.54	590.77	27.7
10	50	4	241	1244.71	622.36	31.8

Fuente: Elaboración propia

La Figura 3.2 muestra la localización del centro distribución y el proceso de ruteo de vehículos abierto para atender las demandas de 50 clientes usando cuatro vehículos, donde

cada color identifica el recorrido realizado por el respectivo vehículo. En este caso para obtener un recorrido bueno se realizaron

$$\frac{(50-1)!}{2} = \ldots = 3{,}041409320171337804361260816606 5e+62$$

operaciones.

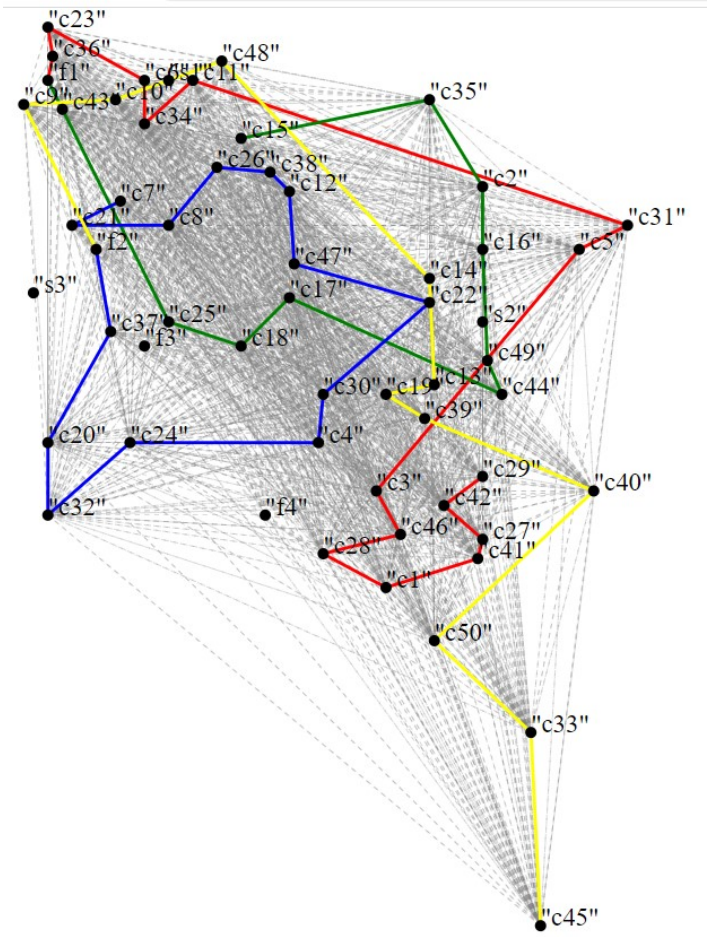

Figura 3.2: Localización de centros de distribución y ruteo en escenario 10.
Fuente: Resultados computacionales obtenidos con glpk

Capítulo 4

Resultados y discusión de la tesis

La investigación culmina con las propuestas del modelo de optimización basado en el paradigma de programación lineal entera binaria y una metaheurística basado en colonia de hormigas, llegándose a resultados interesantes del punto de vista computacional. Estos resultados muestran que se contrasta la hipótesis planteada durante el proceso de elaboración del plan de investigación, es decir, que se logró demostrar la relación entre las variables de estudio formuladas en la investigación.

4.1. Resultados computacionales

Para determinar el potencial de la propuesta realizada donde inicialmente es necesario localizar centros de distribución de productos y su transporte mediante un ruteo de vehículos en la variante abierto sobre una red logística, fue posible implementar el modelo de optimización y la metaheurística, respectivamente. Las pruebas realizadas para ambas propuestas se efectuaron con los mismos datos y en diferentes escenarios, donde la cantidad de clientes se fueron incrementando en cinco unidades hasta un total de cincuenta clientes haciendo un total de diez escenarios. Ver Tabla 3.2.

La Tabla 4.1 muestra la comparación de los tiempos, en segundos, empleados por el modelo y metaheurística propuestos cuyos resultados se mostraron en las Tablas 3.1 y 3.2. Se observa que desde el escenario tres los tiempos empleados para localizar los centros de distribución y hallar el mejor ruteo de vehículos abierto se empiezan a diferenciar grandemente. Debido a esta diferencia de tiempo, el modelo de optimización solamente fue probado hasta un escenario seis. La metaheurística fue probada hasta el escenario diez.

Tabla 4.1: Comparación de tiempos de ejecución modelo versus metaheurística propuestos

Escenario	Clientes	Tiempo con modelo (seg.)	Tiempo con metaheurística (seg.)
1	5	0.2	0.5
2	10	1.4	1.5
3	15	11.3	3.8
4	20	33.6	6.4
5	25	82.3	10.4
6	30	1320.0	13.7
7	35	-	18.3
8	40	-	22.5
9	45	-	27.7
10	50	-	31.8

Fuente: Elaboración propia

El tiempo para hallar los resultados mediante el modelo de programación entera binaria va creciendo de modo exponencial a pesar que la solución obtenida es óptima. La metaheurística también resuelve el problema pero en menor tiempo, sin embargo la solución es buena por ser una estrategia algorítmica de aproximación, pero no deja de ser interesante. Similarmente la Figura 4.1, con datos de la Tabla 4.1, ilustra los tiempos empleados por las propuestas de esta tesis para los diez escenarios considerados en este estudio.

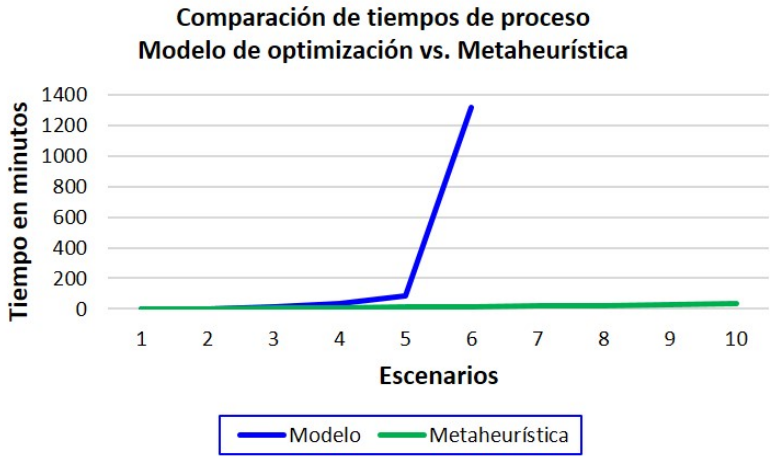

Figura 4.1: Tiempo Modelo vs. Metaheurística
Fuente: Elaboración propia

4.2. Discusión

La tesis se propuso dar solución al problema de localizar centros de distribución por parte de los clientes quienes generan una demanda por algún producto; posteriormente se produce el transporte de los mismos mediante un proceso de ruteo de vehículos con la característica que el vehículo al culminar la atención del último cliente, no regresa al respectivo centro, es decir, que el vehículo se irá a su respectivo depósito.

Para hacer realidad la propuesta se procedió a discutir los fundamentos teóricos necesarios para sustentar la investigación tal como se muestran en el capítulo 2 denominado materiales y métodos. En esa etapa se concluye que la propuesta está dentro de los problemas computacionales denominados NP, para los cuales es posible obtener solución a la problemática mediante dos paradigmas de algoritmos (metodologías); el primero a través del paradigma de programación lineal entera binaria, para el cual se diseñó un modelo de optimización y su implementación para tener una experiencia computacional en diversos escenarios de tamaño moderado, y de este modo conocer hasta que escenario se puede obtener resultados óptimos en tiempo de procesamiento aceptable.

Siendo que el problema a resolver en este tesis es NP, entonces también se optó usar el paradigma de aproximación siendo la metaheurística usada la llamada colonia de hormigas. Desarrollada esta metaheurística se procedió a implementar con los mismos datos usados en la implementación del modelo de optimización. Incluso se agregaron otros escenarios hasta un total de 50 clientes. Como indica el nombre de este paradigma, la solución obtenida es aproximada por ello los resultados obtenidos se denominan solución buena, es decir, no es óptima pero no deja de ser una respuesta aceptable a la problemática que se resuelve en esta investigación.

Ambos paradigmas de algoritmo usados en la tesis resuelven el problema formulado, sin embargo la diferencia está en el tiempo de procesamiento empleado para llegar a las respuestas, localización y ruteo de vehículos pertenecen a la clase NP, en particular el ruteo que tiene un comportamiento de tipo $\frac{(n-1)!}{2}$ operaciones que deben ser realizadas, hace necesario el uso de una metaheurística para tener soluciones en menor tiempo de procesamiento. Ver Tabla 4.1.

Capítulo 5

Consideraciones finales

5.1. Conclusiones

La investigación bibliográfica realizada revela que realmente existe una preocupación por mantener un desarrollo sustentable, pues es necesario preservar la salud de la población y el medio ambiente. Esto es posible si hacemos uso de los conocimientos computacionales aplicados en el contexto de la logística.

Conocida la realidad problemática y por tanto la complejidad, NP-difícil, para resolver el problema planteado, se deducía que era necesario el uso de conocimientos relacionados con optimización y metaheurísticas; en consecuencia se mostró mediante la aplicación de los conceptos de localización, ruteo de vehículos, modelo de programación lineal entera binaria y metaheurística basada en colonia de hormigas, su utilidad para alcanzar el objetivo trazado.

Basado en los paradigmas o metodologías de algoritmos, se desarrolló e implementó un modelo de optimización, basado en la programación entera binaria, y una metaheurística, basado en colonia de hormigas, para viabilizar el recorrido en una red logística tal como muestran los resultados obtenidos, donde el modelo es mas apropiado para escenarios pequeños y la metaheurística para escenarios de mayor tamaño.

5.2. Trabajos futuros

Conocer la realidad problemática para el cual se propusieron metodologías de solución consiguiéndose resultados interesantes, nos abre las puertas para considerar nuevas oportunidades para contribuir con el desarrollo sustentable. Una ampliación de nuestra propuesta consiste en considerar los denominados costos ambientales, es decir, que se debe conocer

la cantidad de emisiones de CO2, azufre, entre otros, que se generan durante el transporte para atender la demanda de los clientes. Esto es posible de ser agregado en un nuevo modelo de optimización, para ello una fórmula matemática que mide la emisión de gases de los vehículos debe ser relacionada con la variable de ruteo.

Dada la complejidad del problema y considerando el párrafo anterior, es posible hacer uso de otras metaheurísticas tales como busca tabú, *simulated annealing*, algoritmo genético y colonia de hormigas. Luego se debe comparar los resultados obtenidos por estas estrategias computacionales para determinar el tiempo de proceso empleado y la calidad de los resultados obtenidos. Similarmente, es posible implementar la propuesta de esta tesis con otras metaheurísticas citadas en este párrafo. La comparación entre metaheurísticas es importante para la toma de decisiones, pues la calidad de los resultados es importante tener presente, no solamente se debe pensar en disminuir el tiempo de proceso, pues esto podría afectar la calidad de la atención al cliente.

Otro trabajo futuro es asumir que los centros de distribución sean considerados como plataformas (grandes depósitos ubicados en la afueras de las ciudades), los cuales atenderán a las facilidades (centros comerciales). Este último se entiende que actúa como cliente. Para estos casos los vehículos a usar deben ser de mayor tonelaje ya que la demanda generada por una facilidad es mayor. Además el proceso de ruteo no debería realizarse en horas de mucha congestión vehicular. Es recomendable que las facilidades estén ubicadas en lugares con amplio espacio.

Los trabajos futuros deben ser enfocados en el contexto de la cadena de suministro sostenibles para así mantener una sostenibilidad urbana mediante una adecuada gestión con protección del medio ambiente. Actualmente se le denomina logística urbana o logística de las ciudades.

Referencias bibliográficas

Amodeo, L., El-Ghazali, T., and Yalaoui, F. (2018). *Recent Developments in Metaheuristics*. Springer, Switzerland AG.

Bouchery, Y., Corbett, C. J.and Fransoo, J., and Tan, T. (2017). *Sustainable supply chains: A research - based textbook on operations and strategy*. Springer, Switzerland.

Braekers, K., Ramaekers, K., and Van Nieuwenhuyse, I. (2016). The vehicle routing problem: State of the art classification and review. *Computers & Industrial Engineering, Elsevier.*, 99(12):300–313.

Brandão, J. (2018). Iterated local search algorithm with ejection chains for the open vehicle routing problem with time windows. *Computer & Industrial engineering, Elsevier*, 120(12):146–159.

Bugliarello, G. (2006). Urban sustainability: Dilemmas, challenges and paradigms. *Technology in society, Elsevier.*, 28(22):19–26.

Clarke, G. and Wright, J. (1964). Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research, INFORMS.*, 12(4):568–581.

Cormen, T., Leiserson, C., Rivest, R., and Stein, C. (2009). *Introduction to algorithms, third edition*. MIT Press., Cambridge, Massachusetts.

Dantzig, G. and Ramser, J. (1959). The truck dispatching problem. *Management science.*, 6(1):80–91.

Daskin, M., LV., S., and Berger, R. (2005). *Facility location in supply chain design*. Langevin A, Riopel D (eds) Logistics systems design and optimization, Springer, New York.

Davendra, D. (2010). *Traveling salesman problem, theory and applications*. InTech, India.

El-Sherbeny, N. A. (2010). The vehicle routing with time windows: An overview of exact, heuristic and metaheuristic methods. *Journal of King saud University*, 2010(22):123–131.

Farahani, R. Z. and Hekmatfar, M. (2009). *Facility location: Conceptos, models, algorithms and case studies*. Springer-Verlag., Berlin, Heidelberg.

Gendreau, M. and Potvin, J. (2019). *Handbook of metaheuristics, Third edition*. Springer, New York Dordrecht Heidelberg London.

Ghiani, G., Laporte, G., and Musmanno, R. (2004). *Introduction to logistics systems planning and control*. Jhon Wiley & Sons.

Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. *Computer & Operations Research*, 13(5):533–549.

GurpreetSingh, E. and Dhir, V. (2014). Open vehicle routing problem by ant colony optimization. *International Journal of Advanced Computer Science and Applications - IJACSA*, 5(3):63–68.

Gutiérrez, F., Navarro, V., Rodríguez, J., Lujan, E., and F.., G. (2020). Vehicle routing applied to natural disasters in south américa: state of art. *Selecciones matemáticas*, 7(2):340–353.

Idzikowski, R. (2020). Capacitated open vehicle routing problem with time couplings. *Theory and Applications of Dependable Computer Systems, Advances in Intelligent Systems and Computing, Springer*, 1173:273–282.

Jungnickel, D. (2013). *Graphs, networks and algorithms, fourth edition*. Algorithms ad computation in mathematics, Vol 5, Springer., Verlag Berlin Heidelberg.

Kumar, K. and Davim, J. P. (2019). *Optimization using evolutionary algorithms and metaheuristics: Applications in engineering (Science, Technology, and Management)*. CRC Press.

Kumar, S. and Panneerselvam, R. (2012). A survey on the vehicle routing problem and its variants. *Intelligent Information Management. Scientific research*, 4(3):66–74.

Laporte, G. (1992). The vehicle routing problem: An overview of exact and approximate algorithms. *European Journal of Operational Research*, 4(59):345–358.

Liong Choong, Y., Wan Rosmanira, I., Khairuddin, O., and Z., M. (2008). Vehicle routing problem: Models and solutions. *Journal of quality measurement and analysis*, 4(1):205–218.

McConnell, J. (2008). *Analysis of algorithms: An active learning approach, second edition*. Jones and Bartlett publishers, United States of America.

Melquiades, J. A. R. (2015). *Modelagem para a roteirização do processo de coleta e transporte dos resíduos sólidos urbanos*. T.D. 003 2015 Departamento de engenharia civil e ambiental, Universidade de Brasília - Brasília, DF, Brasília DF, Brasil.

Monteiro da Costa Cruz, M. and De Alvarenga Rosa, R. (2009). *Operações e logística*. Universidad aberta do Brasil, Brasil.

Neumann, F. and Witt, C. (2010). *Bioinspired computation in combinatorial optimization: Algorithms and their computational complexity*. Natural computing series, Springer., Switzerland AG.

Pichka, K., Ashjari, B., Ziaeifar, A., and Nickbeen, P. (2014). Open vehicle routing problem optimization under realistic assumptions. *International Journal of research in Industrial Engineering*, 3(2):46–55.

Ruiz, E., Soto, V., Ruiz, A., and Reyes, R. (2019). Solving the open vehicle routing problem with capacity and distance constraints with a biased random key genetic algorithm. *Computers & Industrial Engineering, Elsevier*, 133:207–219.

Sathyan, A., Boone, N., and Cohen, K. (2015). Comparison of approximate approaches to solving the traveling salesman problem and its application to uav swarming. *International Journal of Unmanned Systems Engineering (IJUSEng).*, 3(1):1–16.

Schneider, M. and Drexl, M. (2017). *A survey of the standard location-routing problem*. Springer science + business media, New York.

Stadtler, H., Kilger, C., and Meyr, H. (2015). *Supply chain management and advanced planning: Concepts, models, software and case studies, 5th edition*. Springer, Verlag, Berlin Heidelberg.

Talbi, E.-G. (2009). *Metaheuristics: From design to implementation*. John Wiley and Sons, France.

Tanguay, G., Rajaonson, J., Lefevre, J., and Lanoie, P. (2010). Measuring the sustainability of cities: An analysis of the use of local indicators. *Ecological Indiators, Elsevier.*, 26(22):100–115.

Utama, D. M., Dewi, S., Wahid, A., and Santoso, I. (2020). The vehicle routing problem for perishable goods: A systematic review. *Cogent engineering, Computer science.*, 7(1):24.

Apéndice A

Estructura principal de la implementación del modelo de optimización

```
set S ;
set I ;
set J ;
set K ;
set SF := S union J ;
set P := I union J ;
param dist { s in SF , j in SF : s != j };
param d { i in P , j in P : i != j };
param t { j in J };
param flete := 25 ;
param cost { i in P , j in P : i != j }:= flete * d
[ i , j ]/ 50 ;
param f { j in J };
param cf { k in K };
param dd { i in I };
param q { k in K };
param g ;
var W { s in S , j in J : s != j } >= 0 ;
var X { j in J }, binary ;
var Y { i in I , j in J : i != j }, binary ;
var Z { i in P , j in P : i != j }, binary ;
var U { i in P , j in P } >= 0 ;
var M { k in K }, binary ;
var u { i in P } >= 0 ;

minimize CB : sum { s in S , j in J } dist [ s , j ]* W
[ s , j ] + sum { j in J } f [ j ]* X [ j ] + sum { k
in K } cf [ k ]* M [ k ] + sum { i in P , j in I : i
!= j } d [ i , j ]* Z [ i , j ] + sum { i in P , j in
I : i != j } cost [ i , j ]* Z [ i , j ];

s . t .R1 { j in J }: sum { s in S } W [ s , j ] <= t
[ j ]* X [ j ];

s . t .R2 { j in J }: sum { s in S } W [ s , j ] - sum
```

```
{ i in I } dd [ i ]* Y [ i , j ] = 0 ;

s . t .R3 { j in P : j != 'C. Distribucion' }: sum { i in
P : i != j } Z [ i , j ] = 1 ;

s . t .R4 { i in P }: sum { j in P : j != 'C. Distribucion'
and i != j } Z [ i , j ] <= 1 ;

s . t . R5 { i in P : i = 'C. Distribucion' and i !=
'C. Distribucion' }: sum { j in P : j != 'C. Distribucion'
} Z [ i , j ] = 1 ;

s . t . R6 { i in P , j in P : i != j and i !=
'C. Distribucion' and i != 'C. Distribucion' }: u [ i ] -
u [ j ] + card ( P )* Z [ i , j ] <= card ( P ) - 1 ;


s . t .R7 : sum { i in P , j in J : i != j } Z [ i , j
] <= g ;

s . t .R8 : sum { i in J , a in I : i != a } Z [ i , a
] - sum { a in I , j in J : a != j } Z [ a , j ] = 0
;

s . t . R9 { i in P , j in P , k in K : i !=
'C. Distribucion' and i != 'C. Distribucion' and i !=
j }: Z [ i , j ] <= M [ k ];

s . t .R10 { i in I , j in J }: sum { m in P : i != m
} Z [ i , m ] + sum { n in P : n != j } Z [ j , n ] -
Y [ i , j ] <= 1 ;

s . t .R11 { i in I }: sum { j in P : i != j } U [ j ,
i ] - sum { j in P : i != j } U [ i , j ] = dd [ i ];


s . t .R12 { i in P , j in P , k in K : i != j }: U
[ i , j ] <= q [ k ]* Z [ i , j ];
```

Apéndice B

Principales funciones de la implementación de la metaheurística

```
def generarRuta(hormiga_k,nodo_inicio_P,Z,I,feromonas,hq,dist_max_h,
tj,dd,distancias_P,Y):
    parada = False
    temp = None
    nodo_actual = nodo_inicio_P
    nodo_siguiente = None
    distancia_acum = 0
    demanda_acum = 0
    ruta = [nodo_inicio_P]
    pedidos_acum = 0

    while not parada:
        nodos_destinos = obtenerNodosDestinos(Z, I, nodo_actual, Y)
        if len(nodos_destinos) > 0:
            temp = np.empty((len(nodos_destinos),))
            for i in range(len(nodos_destinos)):
                temp[i] = f * v
            suma = np.sum(temp)
            for i in range(len(nodos_destinos)):
                temp[i] = temp[i]/suma
            suma = 0
            numero_aleatorio = random.random()

            for i in range(len(nodos_destinos)):
                if numero_aleatorio <= (suma + temp[i]):
                    nodo_siguiente = nodos_destinos[i]
                    break
                else:
                    suma += temp[i]
            distancia_acum += distancias_P[nodo_actual][nodo_siguiente]
            demanda_acum += dd[nodo_siguiente]

            if distancia_acum <= dist_max_h and demanda_acum <= hq
            and demanda_acum <= tj:
                ruta.append(nodo_siguiente)
                pedidos_acum += dd[nodo_siguiente]

                Z[nodo_actual][nodo_siguiente] = hormiga_k
                nodo_actual = nodo_siguiente
                Y[nodo_actual] = 1
            else:
                parada = True
```

```
        else:
            parada = True

    return [ruta, pedidos_acum]


def obtenerDistanciasSF(SF, coord_SF)

def obtenerDistanciasP(P, coord_P)

def obtenerCostosDeTransporte(flete, dist)

def inicializarFeromonas(redP)

def actualizarFeromonas(var_ruteo, feromonas, costosPorRuta)

def generarRuta(hormiga_k, nodo_inicio_P, Z, I, feromonas, hq, dist_max_h
    , tj, dd, distancias_P, Y)

def obtenerNodosDestinos(var_ruteo, clientes, nodo_origen, Y)

def elegirProveedor(facilidad, proveedores, distanciaSF)

def elegirFacilidad(facilidades, clientes, distanciasP, var_ruteo, Y,
    facilidades_descartadas=[])

def costoDeTransporteDeRuta(ruta, costos_p)

def busquedaLocal2opt(ruteo_aco, costos_p, Z)

def funcionObjetivo(S, J, I, K, distancias_sf, distancias_p, costos_p, W,
     Z, f, X, cf, M)

def generarRuteo(S, I, J, K, SF, P, dist, d, t, flete, cost, f, cf, dmax,
     dd, q, g)
```

UNIVERSIDAD NACIONAL DE TRUJILLO

DECLARACION JURADA
RR-384-2018/UNT

Los autores suscritos en el presente documento DECLARAMOS BAJO JURAMENTO que somos los responsables legales de la calidad y originalidad del contenido del Proyecto de Investigación Científica, así como, del Informe de Investigación Científica realizado.

Titulado: Modelo de optimización y metaheurística para localizar centros de distribución de productos y su transporte usando ruteo de vehículos abierto en una red logística

PROYECTO DE INVESTIGACIÓN CIENTÍFICA		**INFORME FINAL DE INVESTIGACIÓN CIENTÍFICA**	
PROY DE TRABAJO DE INVESTIGACIÓN (PREGRADO)	**( )**	**TRABAJO DE INVESTIGACIÓN (PREGRADO)**	**( )**
PROYECTO DE TESIS PREGRADO	**( )**	**TESIS PREGRADO**	**(X)**
PROYECTO DE TESIS MAESTRÍA	**( )**	**TESIS MAESTRIA**	**( )**
PROYECTO DE TESIS DOCTORADO	**( )**	**TESIS DOCTORADO**	**( )**

El equipo investigador Integrado por:

N°	APELLIDOS Y NOMBRES	FACULTAD/DEPARTAMENTO	CATEGORIA DOCENTE ASESOR	CÓDIGO Docente Numero Matricula del estudiante	Autor Coautor Asesor
1	Cruzado Berrú Luis Enrique	CFyM/Informática		1512701214	Autor
2	Rodríguez Melquiades José Antonio	CFyM/Informática	Asociado TC	4891	Asesor

Trujillo, 07 de julio de 2021

…… 70861004

FIRMA DNI

…… 17809561

UNIVERSIDAD NACIONAL DE TRUJILLO

CARTA DE AUTORIZACIÓN DE PUBLICACIÓN DE TRABAJO DE INVESTIGACIÓN EN REPOSITORIO DIGITAL RENATI-SUNEDU
RR-384-2018/UNT

Trujillo, 07 de julio de 2021

Los autores suscritos del INFORME FINAL DE INVESTIGACIÓN CIENTIFICA

Titulado: Modelo de optimización y metaheurística para localizar centros de distribución de productos y su transporte usando ruteo de vehículos abierto en una red logística

AUTORIZAMOS SU PUBLICACIÓN EN EL REPOSITORIO DIGITAL INSTITUCIONAL, REPOSITORIO RENATI-SUNEDU, ALICIA-CONCYTEC CON EL SIGUIENTE TIPO DE ACCESO:

A. Acceso Abierto: [x]
B. Acceso Restringido [ ] (datos del autor y resumen del trabajo)
C. No autorizo su Publicación [ ]

Si eligió la opción restringido o NO autoriza su publicación sírvase justificar:

ESTUDIANTES DE PREGRADO: TRABAJO DE INVESTIGACIÓN [ ] TESIS [x]
ESTUDIANTES DE POSGRADO : TESIS MAESTRIA [ ] TESIS DOCTORADO [ ]
DOCENTES: INFORME DE INVESTIGACIÓN [ ]

El equipo investigador Integrado por:

N°	APELLIDOS Y NOMBRES	FACULTAD	CONDICIÓN DOCENTE (NOMBRADO, CONTRATADO, EMÉRITO, estudiante, OTROS)	CÓDIGO Docente Numero Matricula del estudiante	Autor Coautor Asesor
1	Cruzado Berrú Luis Enrique	Ciencias físicas y matemáticas		1512701214	Autor
2	Rodríguez Melquiades José Antonio	Ciencias físicas y matemáticas	Asociado TC	4891	Asesor

..... 70861004
FIRMA DNI

..... 17809561