Universidad Autónoma
Metropolitana

unidad cuajimalpa

Resolución del Problema de Rutas Vehiculares con Capacidades y Ventanas de Tiempo mediante un Algoritmo Híbrido entre Colonia de Hormigas y Recocido Simulado

Proyecto Terminal

que presenta:
alejandro martínez guzmán

Licenciatura en Ingeniería en Computación

Departamento de Matemáticas Aplicadas e Ingeniería

División de Ciencias Naturales e Ingeniería

Asesor:
EDWIN MONTES OROZCO

Junio 2025

Declaración

Yo, ALEJANDRO MARTÍNEZ GUZMÁN, declaro que este trabajo titulado *«Resolución del Problema de Rutas Vehiculares con Capacidades y Ventanas de Tiempo mediante un Algoritmo Híbrido entre Colonia de Hormigas y Recocido Simulado»* es de mi autoría. Asimismo, confirmo que:

- Este trabajo fue realizado en su totalidad para la obtención de grado en esta Universidad.
- Ninguna parte de esta tesis ha sido previamente sometida a un examen de grado o titulación en esta u otra institución.
- Todas las citas han sido debidamente referenciadas y atribuidas a sus autores.

Firma: _____

Fecha: _____

Resumen

Aquí va el resumen en español. Este apartado debe sintetizar brevemente el objetivo del trabajo, la metodología empleada y los resultados más relevantes.

Abstract

Here goes the abstract in English. Briefly describe the goal of your project, methodology, and key results.

Dedicatoria

A mis padres y profesores, por su apoyo incondicional.

Agradecimientos

Aquí van los agradecimientos a las personas e instituciones que contribuyeron al desarrollo de este proyecto.

Índice general

Índice de figuras

Índice de tablas

Índice de fórmulas

Capítulo 1

Introducción

1.1. Planteamiento del Problema

1.2. Justificación

1.3. Objetivos

1.3.1. Objetivo General

Diseñar, implementar, probar y evaluar un enfoque híbrido que combine la Optimización por Colonia de Hormigas (ACO), el Recocido Simulado (SA) con múltiples heurísticas locales y un Algoritmo Evolutivo Diferencial (ADE), con el objetivo de resolver de manera eficiente el Problema de Ruteo de Vehículos con Ventanas de Tiempo (VRPTW), utilizando el TSP y el VRP como base introductoria y estructural para la construcción progresiva del modelo.

1.3.2. Objetivos Específicos

1. Estudiar y analizar los problemas TSP, VRP y VRPTW para identificar sus similitudes estructurales y diferencias, estableciendo así una base conceptual que permita abordar de forma progresiva el problema principal (VRPTW).

2. Diseñar un enfoque híbrido que combine la Optimización por Colonia de Hormigas (ACO) para la construcción de soluciones iniciales, el Recocido Simulado (SA) con distintas heurísticas de búsqueda local para la mejora de dichas soluciones, y un Algoritmo Evolutivo Diferencial (ADE) para intensificar la exploración del espacio de búsqueda.

3. Implementar computacionalmente el enfoque híbrido propuesto, asegurando su modularidad y capacidad de adaptación para resolver los tres problemas (TSP, VRP y VRPTW).

4. Evaluar el rendimiento del enfoque híbrido en términos de calidad de la solución, tiempo de ejecución y robustez, utilizando instancias de prueba estándar y métricas aceptadas en la literatura.

5. Analizar el impacto individual y combinado de las técnicas ACO, SA y ADE en el rendimiento del sistema, y proponer recomendaciones para su aplicación práctica en problemas logísticos reales.

1.4. Alcances y Limitaciones

Alcances

- El trabajo se centra en la resolución de los problemas TSP, VRP y VRPTW, siendo este último el problema principal de estudio.
- Se desarrolla un enfoque híbrido que integra tres técnicas metaheurísticas: Optimización por Colonia de Hormigas (ACO), Recocido Simulado (SA) con múltiples heurísticas locales, y un Algoritmo Evolutivo Diferencial (ADE), aplicadas de manera complementaria.
- La implementación incluye una arquitectura modular que permite la reutilización de componentes entre los tres problemas, facilitando la comparación y evolución de los modelos.
- El enfoque es probado con instancias benchmark conocidas en la literatura, permitiendo una validación objetiva y reproducible de los resultados obtenidos.
- Se analizan métricas relevantes como la calidad de la solución, el tiempo de ejecución y la estabilidad del algoritmo ante múltiples ejecuciones.

Limitaciones

- El enfoque propuesto está diseñado para instancias estáticas de los problemas de ruteo; no se contempla el tratamiento de versiones dinámicas o en tiempo real.
- La calidad de los resultados depende en gran medida de la correcta calibración de los parámetros de ACO, SA y ADE, lo cual puede implicar un proceso costoso de ajuste manual o semiautomático.
- No se consideran restricciones adicionales como múltiples depósitos, flotas heterogéneas, restricciones de carga o prioridades diferenciadas entre clientes.

- Las pruebas están limitadas a conjuntos de datos artificiales y benchmarks académicos, sin validación en entornos productivos reales.
- Al tratarse de métodos metaheurísticos, no se garantiza la obtención del óptimo global, sino soluciones cercanas al óptimo dentro de un tiempo razonable.

1.5. Estructura del Documento

Capítulo 2

Marco Teórico / Estado del Arte

2.1. Antencedentes

2.2. Conceptos Clave

2.2.1. El Problema en Optimización

Los **problemas de optimización** constituyen una herramienta fundamental en ciencias computacionales, ingeniería, logística y muchas otras disciplinas. En el contexto de este trabajo de investigación, resultan especialmente relevantes porque permiten modelar situaciones en las que se busca obtener el **mejor resultado posible** bajo ciertas restricciones, como **minimizar costos**, **distancias** o **tiempos**.

De acuerdo con [4], los problemas de optimización se dividen de manera natural en dos categorías: problemas con **variables continuas** y problemas con **variables discretas**. A estos últimos se les conoce como problemas de **optimización combinatoria**.

La función $f : S \to \mathbb{R}$, donde el conjunto S es un subconjunto de $\mathbb{R}^n$, se denomina **función objetivo**, **función de costo** o **beneficio**, mientras que el conjunto S se identifica como el **conjunto factible** o el conjunto de **soluciones posibles** [4].

Como definición formal, se puede decir que el problema general de optimización consiste en encontrar un elemento $x \in S$ que **optimice** la función objetivo. En el caso de un problema de **minimización**, se expresa de la siguiente manera:

$$\min_{x \in S} f(x).$$

Por su parte, en un problema de **maximización**, se utiliza la notación:

$$\max_{x \in S} f(x).$$

[4]

2.2.1.1. Optimización Continua

En la **optimización continua** para problemas de minimización, el objetivo es encontrar un punto x_{opt} dentro del conjunto factible S tal que el valor de la función objetivo en ese punto sea menor o igual que el valor en cualquier otro punto del conjunto, es decir,

$$f(x_{opt}) \leq f(x), \quad \forall x \in S.$$

Para problemas de maximización, se busca un punto x_{opt} en S donde la función objetivo tome un valor mayor o igual al de cualquier otro punto factible, esto es,

$$f(x_{opt}) \geq f(x), \quad \forall x \in S.$$

A este punto se le denomina **solución óptima global**, y el valor correspondiente $f(x_{opt})$ se conoce como **costo óptimo**. Además, el conjunto de todas las soluciones óptimas se representa por S_{opt} [4].

2.2.1.2. Optimización Discreta o Combinatoria

Una instancia de un problema de **optimización combinatoria** puede representarse mediante una pareja (S, f), donde S es un conjunto **finito** que contiene todas las soluciones posibles, y f es una función real, denominada **función objetivo** o **función costo**, definida como

$$f : S \to \mathbb{R}.$$

Para problemas de minimización, se busca una solución $i_{opt} \in S$ tal que

$$f(i_{opt}) \leq f(i), \quad \forall i \in S,$$

mientras que para problemas de maximización, se requiere que

$$f(i_{opt}) \geq f(i), \quad \forall i \in S.$$

La solución i_{opt} se denomina **solución óptima global**, y el valor $f(i_{opt})$ representa el **costo óptimo**. El conjunto de todas las soluciones óptimas se denota como S_{opt}. Un

problema de optimización combinatoria se define entonces como el conjunto de todas sus instancias [4].

Comprender esta distinción es esencial para abordar el problema central de este trabajo **(VRPTW)**, el cual se enmarca dentro de la optimización combinatoria. En este tipo de problemas, el número de soluciones posibles es **finito**, y el desafío consiste en identificar cuál de ellas representa la **mejor opción** según un criterio específico.

Esta base teórica permite introducir los problemas clásicos de enrutamiento, como el **Problema del Agente Viajero (TSP)**, el **Problema de Enrutamiento de Vehículos (VRP)** y sus variantes, como el **Problema de Enrutamiento de Vehículos con Ventanas de Tiempo (VRPTW)**, los cuales serán analizados más adelante por su relación directa con la planificación eficiente de rutas en contextos reales.

2.2.2. Complejidad Computacional y Algorítmica

El estudio de la complejidad en ciencias computacionales busca comprender qué tan difícil es resolver un problema, tanto desde su estructura teórica como desde los recursos requeridos para su solución. Esta dificultad se aborda desde dos enfoques: la **complejidad computacional**, que clasifica los problemas según el crecimiento de recursos necesarios en función del tamaño de la entrada, y la **complejidad algorítmica**, que evalúa el desempeño de algoritmos específicos al aplicarse sobre dichas entradas [6, 11].

De forma simplificada, la complejidad computacional se refiere al análisis general del problema sin importar el algoritmo, mientras que la algorítmica se enfoca en el tiempo y espacio consumido por un algoritmo concreto [9].

Esta distinción es clave al abordar problemas de optimización combinatoria como el **Problema del Agente Viajero (TSP)**, el **Problema de Ruteo de Vehículos (VRP)** y su variante principal en este trabajo: el **Problema de Enrutamiento de Vehículos con Ventanas de Tiempo (VRPTW)**. Estos problemas presentan una explosión combinatoria de soluciones conforme crece el número de clientes.

2.2.3. El Problema P vs NP

Todos los problemas, tanto en la vida como en la ciencia, pueden clasificarse en dos grupos principales: **problemas decidibles** y **problemas indecidibles**. Se dice que un problema es ***indecidible*** cuando no existe ningún algoritmo que permita resolverlo, incluso si se dispusiera de tiempo y recursos ilimitados. En consecuencia, no es posible determinar si dicho problema se detendrá ni en qué momento lo hará. Por contraste, un problema es ***decidible*** cuando existe, o al menos podría existir, algún algoritmo capaz de resolverlo [9].

Además, un problema se considera decidible cuando es compresible; es decir, cuando existe una fórmula o algoritmo que permite condensarlo o expresarlo de forma finita. Por

el contrario, los problemas indecidibles se consideran incompresibles, ya que no pueden reducirse a una descripción finita ni predecirse computacionalmente [9].

En el contexto de este trabajo, los problemas abordados —el **Problema del Agente Viajero (TSP)**, el **Problema de Ruteo de Vehículos (VRP)** y su variante con restricciones temporales, el **Problema de Ruteo de Vehículos con Ventanas de Tiempo (VRPTW)**— son considerados **problemas decidibles**. A continuación, se analizan las clases de complejidad que resultan relevantes para comprender con mayor profundidad la dificultad computacional asociada a su resolución.

2.2.3.1. Clases de Complejidad Básicas

Dentro de los problemas decidibles, se distinguen dos categorías fundamentales en la teoría de la complejidad: los problemas ***P*** y los problemas ***NP*** [9].

Clase P Un problema pertenece a la clase P si puede resolverse mediante un algoritmo que se ejecuta en tiempo polinomial con respecto al tamaño de la entrada. Es decir, el número de pasos que requiere el algoritmo está acotado por un polinomio de grado k. De manera formal:

$$(\exists\, \alpha > 0)\,(\exists\, n_0 > 0)\ :\ D(n) \leq \alpha \cdot p(n) \quad (\forall\, n \geq n_0),$$

donde $p(n)$ es un polinomio. En notación asintótica, esto se expresa como:

$$D = O(n^k).$$

Aunque se dice que un algoritmo que se ejecuta en tiempo polinomial es "eficiente", si el grado del polinomio es muy alto (por ejemplo, n^{10000}), su aplicabilidad práctica puede verse limitada. En general, se espera que los algoritmos eficientes tengan un grado bajo. Un ejemplo típico de problema en P es la multiplicación de dos números naturales, cuyo algoritmo básico opera en tiempo lineal [5].

Clase NP Por otro lado, un problema pertenece a la clase NP si, aunque no se sepa cómo resolverlo eficientemente, sí puede *verificarse* en tiempo polinomial. Es decir, si se proporciona una posible solución (llamada certificado), puede comprobarse su validez en un tiempo razonable. Un ejemplo clásico es el ordenamiento de elementos: si bien existen algoritmos polinómicos para ordenarlos, el proceso de verificar si una secuencia está ordenada también toma tiempo lineal [5].

Aunque algunos algoritmos de ordenamiento, como el *bubble sort*, operan en tiempo polinomial, en teoría podría existir un algoritmo con tiempo exponencial. Sin embargo, si al menos uno de los algoritmos conocidos resuelve el problema en tiempo polinomial, el problema pertenece a la clase P [5].

La gran incógnita: ¿P = NP? Una de las cuestiones abiertas más fundamentales en ciencias computacionales es determinar si **P** = **NP** o **P** ≠ **NP**. Esta pregunta implica saber si todo problema cuya solución puede verificarse rápidamente, también puede resolverse rápidamente. Es decir, si los problemas que se pueden verificar en tiempo polinomial también pueden resolverse en ese mismo tipo de tiempo. Este dilema sigue sin resolverse y tiene profundas implicaciones teóricas y prácticas [9].

2.2.3.2. NP-Completo y NP-Difícil

Dentro de los problemas de clase *NP*, existe una subcategoría especialmente importante: los problemas **NP-completos**. Estos son problemas que pertenecen a *NP* y que, además, son al menos tan difíciles como cualquier otro problema en *NP*. En otras palabras, si se lograra resolver uno de estos problemas de forma eficiente, también podrían resolverse todos los demás problemas en *NP* [9].

Por otra parte, los **problemas NP-difíciles (NP-hard)** son aquellos que pueden ser incluso más complejos que los *NP-completos*, ya que no necesariamente pertenecen a *NP* (es decir, podrían no tener soluciones verificables en tiempo polinomial), pero son al menos tan difíciles como los problemas más complicados de *NP*. Este tipo de problemas puede presentarse como problemas de decisión, de búsqueda o de optimización [9].

El crecimiento exponencial del tiempo requerido para resolver este tipo de problemas, como el **Problema del Agente Viajero (TSP)**, el **Problema de Ruteo de Vehículos (VRP)** y el **Problema de Ruteo de Vehículos con Ventanas de Tiempo (VRPTW)**, en función del tamaño de la entrada, los ubica dentro de la clase *NP*. Más adelante se explicará con mayor detalle a qué subcategoría pertenece cada uno de estos problemas.

2.2.4. Problemas de Optimización Combinatoria

De manera complementaria a la definición formal presentada en la Subsubsección 2.2.1.2 por [4], Papadimitriou y Steiglitz [12] indican que la optimización combinatoria como el estudio de problemas de optimización en los que el conjunto de soluciones factibles es discreto o puede hacerse discreto mediante un proceso de enumeración, y en los que se busca una solución que optimice una función objetivo definida sobre ese conjunto.

Los problemas de optimización combinatoria constituyen una clase amplia de modelos que encuentran aplicación en numerosas áreas, especialmente en la planificación y enrutamiento de vehículos. Dentro de este campo, destacan el **Problema del Agente Viajero (TSP)**, que representa la forma más básica de la optimización de rutas, y distintas extensiones que incorporan restricciones adicionales propias de entornos reales, tales como las limitaciones de capacidad de los vehículos o las ventanas de tiempo en las que debe efectuarse la entrega de los productos. En este contexto, el **Problema de Ruteo de Vehículos (VRP)** y el **Problema de Ruteo de Vehículos con Ventanas de Tiempo**

(VRPTW) se han consolidado los desafíos más estudiados por su relevancia práctica y su elevada dificultad computacional. A continuación, se describen estos problemas en detalle:

2.2.4.1. Problema del Agente Viajero (TSP)

El **Problema del Agente Viajero**, conocido por sus siglas en inglés como *TSP* (*Travelling Salesman Problem*), es uno de los problemas más reconocidos y complejos dentro de las ciencias computacionales. Ha sido estudiado desde diversas ramas de la ingeniería y por múltiples motivos. Su aplicación principal consiste en determinar rutas desde diferentes enfoques, ya sea en procesos que requieren una secuencia específica o en operaciones logísticas relacionadas con el transporte, con el objetivo de encontrar la ruta óptima considerando criterios de minimización de distancia o de costo [8].

Según [13], el **Problema del Agente Viajero** se define sobre un grafo $G = [N, A, C]$, donde N es el conjunto de nodos, A es el conjunto de arcos y $C = [c_{ij}]$ es la matriz de costos (distancias) entre nodos i y j. El objetivo es encontrar un ciclo hamiltoniano de costo mínimo que recorra todos los nodos una sola vez y regrese al punto de partida.

Modelo Matemático del Agente Viajero (TSP)

$$\min \sum_{i=1}^{N} \sum_{j=1}^{N} c_{ij}\, x_{ij} \tag{2.1}$$

sujeto a:

$$\sum_{j=1}^{N} x_{ij} = 1, \quad \forall i = 1, \ldots, N \tag{2.2}$$

$$\sum_{i=1}^{N} x_{ij} = 1, \quad \forall j = 1, \ldots, N \tag{2.3}$$

$$u_i - u_j + N x_{ij} \leq N - 1, \quad \forall i, j = 2, \ldots, N,\ i \neq j \tag{2.4}$$

$$x_{ij} \in \{0, 1\}, \quad \forall i, j = 1, \ldots, N \tag{2.5}$$

donde:

- $x_{ij} = 1$ si la ruta va de i a j, y 0 en caso contrario;
- u_i son variables auxiliares usadas para eliminar subciclos, con $i = 2, \ldots, N$ [13].

La **función objetivo** (2.1) representa la suma total de los costos (o distancias) de los arcos que conforman el recorrido. La restricción en (2.2) garantizan que, al salir de cada nodo, sólo se dirija a un único nodo siguiente. La restricción en (2.3) aseguran que a cada nodo únicamente se llegue una vez. La restricción (2.4) es fundamental, ya que previene la formación de subciclos que no incluyan a todos los nodos, asegurando así un recorrido único y completo.Finalmente, la restricción (2.5) indica que las variables x_{ij} son **binarias**, es decir, sólo pueden tomar los valores 0 o 1, lo que permite representar la inclusión o exclusión de una ruta entre nodos [13].

Complejidad del TSP Como explica [12], el **Problema del Agente Viajero** es *NP-hard* (NP-difícil), y su versión de decisión es *NP-completa*. Esto implica que la búsqueda de un algoritmo eficiente para resolverlo representa un desafío fundamental en la teoría de la computación, pues encontrar una solución polinomial para el **TSP** permitiría resolver eficientemente toda la clase *NP*. Esta dificultad justifica la utilización de algoritmos aproximados y metaheurísticos, dado que las técnicas exactas sólo resultan prácticas para instancias

El **Problema del Agente Viajero (TSP)** ha sido tradicionalmente considerado como el punto de partida para el estudio de problemas más complejos de enrutamiento, como el *Problema de Ruteo de Vehículos (VRP)* y sus variantes. Su análisis proporciona una base conceptual sólida para comprender aspectos fundamentales como la construcción de rutas, la minimización de costos y las restricciones operativas. Estudiar el **TSP** permite introducir gradualmente nuevas variables y condiciones, lo que lo convierte en un buen comienzo para abordar modelos más realistas en logística.

2.2.4.2. Problema del Problema Ruteo de Vehículos (VRP)

El problema de ruteo de vehículos, conocido por sus siglas en inglés como *VRP* (Vehicle Routing Problem). Este problema puede interpretarse como la convergencia de dos problemas clásicos de optimización combinatoria: el problema del agente viajero (TSP) mencionado en Subsubsección 2.2.4.1, en el que se asume una capacidad infinita de los vehículos, y el problema de empaquetamiento en compartimentos (BPP) [3].

Según [14], el Problema de Ruteo de Vehículos (VRP) se define sobre un grafo $G = (V, A)$, donde $V = \{0, 1, \ldots, n\}$ representa el conjunto de nodos, con el nodo 0 correspondiente al depósito y los nodos restantes a los clientes, A es el conjunto de arcos, y $C = [c_{ij}]$ es la matriz de costos asociados a los arcos (i, j). El objetivo es determinar un conjunto de rutas de costo mínimo para una flota de vehículos idénticos, de modo que: cada cliente sea visitado exactamente una vez por un solo vehículo, cada ruta comience y termine en el depósito, y la demanda total de los clientes en cada ruta no exceda la capacidad del vehículo.

Modelo Matemático del Problema de Ruteo de Vehículos (VRP)

$$\text{mín} \sum_{i \in V} \sum_{j \in V} c_{ij} x_{ij} \tag{2.6}$$

sujeto a:

$$\sum_{i \in V} x_{ij} = 1, \quad \forall j \in V \setminus \{0\} \tag{2.7}$$

$$\sum_{j \in V} x_{ij} = 1, \quad \forall i \in V \setminus \{0\} \tag{2.8}$$

$$\sum_{i \in V} x_{i0} = K \tag{2.9}$$

$$\sum_{j \in V} x_{0j} = K \tag{2.10}$$

$$\sum_{i \in S} \sum_{j \in S} x_{ij} \geq r(S), \quad \forall S \subseteq V \setminus \{0\},\ S \neq \emptyset \tag{2.11}$$

$$x_{ij} \in \{0, 1\}, \quad \forall i, j \in V \tag{2.12}$$

donde:

- V es el conjunto de nodos (clientes más depósito), con el depósito representado por el nodo 0;
- c_{ij} es el costo (o distancia) de viajar del nodo i al nodo j;
- $x_{ij} = 1$ si se viaja directamente del nodo i al nodo j, 0 en otro caso;
- K es el número de vehículos disponibles en el depósito;
- S es un subconjunto de clientes, y $r(S)$ representa el número mínimo de vehículos necesarios para satisfacer la demanda total de los clientes en S [14].

La **función objetivo** (2.6) minimiza el costo total de todas las rutas. Las restricciones (2.7) y (2.8) aseguran que cada cliente sea visitado exactamente una vez. Las restricciones (2.9) y (2.10) controlan el número de vehículos que entran y salen del depósito. La restricción (2.11) evita la formación de subrutas que no incluyan al depósito, garantizando conectividad. Finalmente, la restricción (2.12) indica que las variables de decisión son binarias [14].

Complejidad del VRP Como señalan Laporte y Nobert [7], el **Problema de Ruteo de Vehículos (VRP)** es *NP-hard*, incluso en su versión más simple con un solo vehículo (es decir, el TSP). Esto significa que no se conoce ningún algoritmo que lo resuelva de forma exacta en tiempo polinomial, y su complejidad crece exponencialmente con el número de clientes. Por ello, la mayoría de las instancias prácticas requieren el uso de técnicas heurísticas o metaheurísticas, ya que los métodos exactos son computacionalmente inviables para problemas de tamaño medio o grande.

El **Problema de Ruteo de Vehículos (VRP)** representa la extensión natural del TSP hacia escenarios logísticos más realistas y complejos, incorporando restricciones operativas que reflejan las limitaciones del mundo real. Su estudio es fundamental para comprender cómo las restricciones de capacidad, múltiples vehículos y la gestión de flotas impactan en la optimización de rutas. El **VRP** sirve como base conceptual para abordar problemas logísticos avanzados como la distribución con ventanas de tiempo, el ruteo con múltiples depósitos y la optimización de última milla en el comercio electrónico. Analizar el **VRP** permite desarrollar una comprensión profunda de los trade-offs entre minimización de costos, utilización eficiente de recursos y satisfacción de restricciones operativas, estableciendo los fundamentos teóricos necesarios para abordar las complejidades de los sistemas de distribución modernos.

2.2.4.3. Problema de Ruteo de Vehículos con Ventanas de Tiempo (VRPTW)

El **Problema de Ruteo de Vehículos con Ventanas de Tiempo (VRPTW)** es una extensión del VRP descrito en Subsubsección 2.2.4.2, donde cada cliente debe ser atendido dentro de una ventana temporal específica. Además, el vehículo debe permanecer en la ubicación del cliente durante el tiempo de servicio requerido. Las ventanas pueden ser rígidas, donde no se permite iniciar el servicio fuera del intervalo definido, o flexibles, admitiendo retrasos con penalización. Si el vehículo llega antes del inicio de la ventana, debe esperar hasta el comienzo del servicio [14].

Modelo Matemático del VRPTW

$$\text{mín} \sum_{k \in K} \sum_{i \in V} \sum_{j \in V} c_{ij} x_{ijk} \tag{2.13}$$

Sujeto a:

$$\sum_{k \in K} \sum_{j \in V} x_{ijk} = 1, \quad \forall i \in N \tag{2.14}$$

$$\sum_{j \in V} x_{0jk} = 1, \quad \forall k \in K \tag{2.15}$$

$$\sum_{i \in V} x_{i,n+1,k} = 1, \quad \forall k \in K \tag{2.16}$$

$$\sum_{j \in V} x_{ijk} - \sum_{j \in V} x_{jik} = 0, \quad \forall k \in K, \forall i \in N \tag{2.17}$$

$$w_{jk} \geq w_{ik} + s_i + t_{ij} - M(1 - x_{ijk}), \quad \forall k \in K, \forall i, j \in V \tag{2.18}$$

$$a_i \leq w_{ik} \leq b_i, \quad \forall k \in K, \forall i \in V \tag{2.19}$$

$$\sum_{i \in V} \sum_{j \in V} d_i x_{ijk} \leq C, \quad \forall k \in K \tag{2.20}$$

$$x_{ijk} \in \{0, 1\}, \quad w_{ik} \geq 0, \quad \forall k \in K, \forall i, j \in V \tag{2.21}$$

donde:

- $V = \{0, 1, \ldots, n, n+1\}$ es el conjunto de nodos, donde 0 y $n+1$ representan el depósito inicial y final;
- $N = V \setminus \{0, n+1\}$ es el conjunto de clientes;
- K es el conjunto de vehículos disponibles;
- A es el conjunto de arcos permitidos entre nodos;
- c_{ij} es el costo o distancia de viajar del nodo i al nodo j;

- x_{ijk} es variable binaria, igual a 1 si el vehículo k viaja directamente del nodo i al nodo j, 0 en otro caso;
- w_{ik} es el tiempo de inicio del servicio del vehículo k en el nodo i;
- s_i es el tiempo de servicio requerido en el nodo i;
- t_{ij} es el tiempo de viaje del nodo i al nodo j;
- $[a_i, b_i]$ es la ventana de tiempo para el nodo i;
- d_i es la demanda del cliente i;
- C es la capacidad máxima del vehículo;
- M es un número grande usado para linearizar las restricciones condicionales.

La **función objetivo** (2.13) busca minimizar el costo total de las rutas. La restricción (2.14) garantiza que cada cliente sea visitado exactamente una vez por algún vehículo. Las restricciones (2.15) y (2.16) aseguran que cada vehículo comience y termine su ruta en el depósito. La restricción (2.17) mantiene el balance de flujo en cada cliente para cada vehículo. La restricción (2.18) impone el orden y respeto a las ventanas de tiempo mediante la relación entre tiempos de servicio y viajes, usando el parámetro M. La restricción (2.19) asegura que el servicio comience dentro de la ventana permitida. La restricción (2.20) controla la capacidad máxima que puede transportar cada vehículo. Finalmente, la restricción (2.21) define los dominios de las variables de decisión [14].

Complejidad del VRPTW Sabemos que la complejidad del **Problema de Ruteo de Vehículos (VRP)** es *NP-hard*, como se señala en la Subsubsección 2.2.4.2 y en [7]. El **Problema de Ruteo de Vehículos con Ventanas de Tiempo (VRPTW)** es una generalización del VRP clásico, y también pertenece a la clase de problemas *NP-hard*. De hecho, incluso si se ignoran las ventanas de tiempo, el problema sigue siendo difícil de resolver. Al incorporar restricciones temporales, el VRPTW se vuelve aún más complejo, tanto desde el punto de vista computacional como algorítmico. Según Bräysy y Gendreau [2] y Toth y Vigo [14], las ventanas de tiempo reducen significativamente el conjunto de soluciones viables, aumentando así la dificultad del problema.

El **Problema de Ruteo de Vehículos con Ventanas de Tiempo (VRPTW)** representa una evolución del VRP orientada a entornos donde las restricciones temporales son determinantes para la calidad del servicio. Esta variante refleja de forma más precisa los desafíos logísticos actuales, donde los clientes no solo deben ser visitados, sino también atendidos dentro de intervalos de tiempo específicos, lo cual impone una coordinación precisa entre la planificación de rutas, la gestión de recursos y el cumplimiento de niveles de servicio.

2.2.5. Algoritmos Exactos

2.2.6. Heurísticas

Heurística es un concepto cuyo origen se remonta a la Grecia clásica, derivado de la palabra griega *heuriskein*, que significa encontrar o descubrir. Según la historia, se asocia con *eureka*, la famosa exclamación atribuida a Arquímedes [1].

Para ejemplificar este concepto, se expone la siguiente definición:

> Según Zanakis et al. (citado en [10]), las heurísticas son *"procedimientos simples, a menudo basados en el sentido común, que se supone que obtendrán una buena solución (no necesariamente óptima) a problemas difíciles de un modo sencillo y rápido"*.

2.2.6.1. Motivos para emplear métodos heurísticos

Los problemas de decisión que pertenecen a la clase NP corresponden a aquellos para los que no se puede garantizar encontrar una mejor solución en un tiempo polinómico razonable.

En este contexto, los métodos heurísticos se convierten en procedimientos eficientes para hallar buenas soluciones, aunque no se pueda demostrar que estas sean óptimas. En estos métodos, la rapidez con que se obtiene el resultado (*que siempre es menor que el tiempo requerido por otros métodos*) es tan relevante como la calidad de la solución encontrada.

Según [1], es posible emplear métodos heurísticos cuando un problema de optimización, sea determinístico o no, presenta alguna de las siguientes características:

a. El problema tiene una naturaleza tal que no se conoce ningún método exacto para resolverlo.

b. Aunque exista un método exacto para su resolución, su uso resulta computacionalmente muy costoso o inviable.

c. El método heurístico posee mayor flexibilidad que un método exacto, permitiendo, por ejemplo, incorporar condiciones de difícil modelización.

d. El modelo matemático es demasiado extenso, excesivamente no lineal o muy complejo desde el punto de vista lógico.

e. Realizar suposiciones o aproximaciones para simplificar el problema tiende a destruir estructuras del modelo que son esenciales en el contexto real, haciendo que la solución obtenida no sea viable.

f. El método heurístico se emplea como parte de un procedimiento global que garantiza el óptimo de un problema. Existen dos posibilidades:

- El método heurístico proporciona una buena solución inicial de partida.
- El método heurístico interviene en una etapa intermedia del procedimiento; un ejemplo de esto son las reglas que determinan la selección de la variable que entra en la base en el método Simplex.

2.2.6.2. Características de un buen algoritmo heurístico

Según [1], un buen algoritmo heurístico debe poseer las siguientes propiedades:

a. **Ser eficiente**. Que el esfuerzo computacional sea realista y adecuado para obtener la solución.

b. **Ser bueno**. La solución debe estar, en promedio, cerca del óptimo.

c. **Ser robusto**. La probabilidad de obtener una mala solución (lejos del óptimo) debe ser baja.

2.2.7. Matehurísticas

El término *metaheurística* fue introducido por Fred Glover en 1986 [1]. Etimológicamente, deriva de la composición de dos palabras de origen griego, que son "meta" y "heurística". El segundo término ha sido descrito en la sección anterior, mientras que el prefijo meta puede traducirse como "más allá de" o "en un nivel superior" [10].

Con este término, Glover pretendía definir un *procedimiento maestro de alto nivel que guía y modifica otras heurísticas para explorar soluciones más allá de la simple optimalidad local* [10].

Para ilustrar este concepto, cabe mencionar la definición propuesta por J.P. Kelly et al., quien es citado por [10]:

> *Las metaheurísticas son métodos aproximados especialmente diseñados para abordar problemas complejos de optimización combinatoria donde los heurísticos tradicionales resultan ineficaces. Estas técnicas ofrecen un marco flexible para desarrollar algoritmos híbridos que integran conceptos provenientes de la inteligencia artificial, la evolución biológica y procesos estadísticos.*

Las metaheurísticas constituyen un enfoque metodológico que permite el desarrollo de algoritmos híbridos innovadores mediante la integración de principios provenientes de múltiples disciplinas, incluyendo la genética, biología, inteligencia artificial, matemáticas, física y neurología [1].

2.2.7.1. Criterios de Evaluación para Métodos Metaheurísticos

La valoración del desempeño de las metaheurísticas se fundamenta en propiedades específicas que determinan su utilidad tanto práctica como teórica. Es importante reconocer que la optimización simultánea de todas estas características no es factible debido a la naturaleza contradictoria de algunas de ellas [1]:

- **Simplicidad:** La metodología debe fundamentarse en principios claros y accesibles que faciliten su comprensión.
- **Precisión:** Cada etapa y procedimiento debe estar definido mediante términos específicos y concretos.
- **Coherencia:** Los componentes metodológicos deben derivarse lógicamente de los principios fundamentales.
- **Eficiencia:** Debe optimizar el uso de recursos computacionales, tanto en tiempo de procesamiento como en memoria.
- **Generalidad:** La aplicabilidad debe extenderse a diversos tipos de problemas manteniendo un rendimiento satisfactorio.
- **Adaptabilidad:** Capacidad de ajustarse a diferentes escenarios de aplicación y modificaciones sustanciales del modelo.
- **Robustez:** El comportamiento debe mantenerse estable ante variaciones menores en el modelo o contexto.
- **Interactividad:** Debe facilitar la incorporación del conocimiento del usuario para optimizar el rendimiento.
- **Multiplicidad:** Capacidad de generar múltiples alternativas de solución de alta calidad para la selección del usuario.
- **Autonomía:** Funcionamiento independiente con parámetros mínimos o autoconfiguración.

2.3. Referencias Relevantes

Capítulo 3

Materiales y Métodos

3.1. Materiales

3.2. Métodos

Capítulo 4

Resultados

Capítulo 5

Conclusiones

Bibliografía

Bibliografía

[1] Orlando Antonio Suárez. Una aproximación a la heurística y metaheurísticas. *INGE@UAN – Tendencias en la Ingeniería*, 1(2), 2014.

[2] Olli Bräysy and Michel Gendreau. Vehicle routing problem with time windows, part i: Route construction and local search algorithms. *Transportation Science*, 39(1):104–118, 2005.

[3] Julio Mario Daza, Jairo R. Montoya, and Francesco Narducci. Resolución del problema de enrutamiento de vehículos con limitaciones de capacidad utilizando un procedimiento metaheurístico de dos fases. *Revista EIA*, (12):23–38, diciembre 2009.

[4] Sergio Gerardo de los Cobos Silva, John Goddard Close, Miguel Ángel Gutiérrez Andrade, and Alma Edith Martínez Licona. *Búsqueda y exploración estocástica*. Libros CBI, Universidad Autónoma Metropolitana Iztapalapa, México, 2010.

[5] Xavier Alejandro Flores Cabezas. Problemas del milenio: P vs np. *Revista de Divulgación Amarun*, 1:1–15, 2014.

[6] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

[7] Gilbert Laporte and Yves Nobert. Exact algorithms for the vehicle routing problem. *Annals of Discrete Mathematics*, 31:147–184, 1987.

[8] Erasmo López, Óscar Salas, and Álex Murillo. El problema del agente viajero: Un algoritmo determinístico usando búsqueda tabú. *Revista de Matemática: Teoría y Aplicaciones*, 21(1):127–144, 2014.

[9] Carlos Maldonado. Un problema fundamental en la investigación: Los problemas p vs. np. *Revista Logos, Ciencia & Tecnología*, 4(2):10–20, 2013.

[10] Abraham Duarte Muñoz, Juan José Pantrigo Fernández, and Micael Gallego Carrillo. *Metaheurísticas*. Dykinson, Madrid, España, 2007.

[11] Christos H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.

[12] Christos H. Papadimitriou and Kenneth Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Dover Publications, New York, 1998.

[13] J. Torres-Jiménez, A. Aguilar-Melendéz, and M. G. Cedillo-Campos. El problema del agente viajero con restricciones de ventanas de tiempo: revisión y clasificación de métodos de solución. *Revista Iberoamericana de Automática e Informática Industrial*, 15(4):440–452, 2018.

[14] Paolo Toth and Daniele Vigo. *Vehicle Routing: Problems, Methods, and Applications*. SIAM (Society for Industrial and Applied Mathematics), 2nd edition, 2014.

Apéndices