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

Resumen iii

Abstract v

Dedicatoria vii

Agradecimientos ix

1 Introducción 1

2 Marco Teórico 3
2.1 Antencedentes . . . 3
2.2 Conceptos Clave . . . 3
2.2.1 El Problema en Optimización . . . 3
2.2.1.1 Optimización Continua . . . 4
2.2.1.2 Optimización Discreta o Combinatoria . . . 4
2.2.2 Complejidad Computacional y Algorítmica . . . 4
2.2.3 El Problema P vs NP . . . 5
2.2.3.1 Clases de Complejidad Básicas . . . 5
2.2.3.2 NP-Completo y NP-Difícil . . . 6
2.2.4 Problemas de Optimización Combinatoria . . . 7
2.2.4.1 Problema del Agente Viajero (TSP) . . . 7
2.2.4.2 Problema de Ruteo de Vehículos con Capacidades (CVRP) . . . 9
2.2.4.3 Problema de Ruteo de Vehículos con Capacidades y Ventanas de Tiempo (CVRPTW) . . . 9
2.2.5 Algoritmos Exactos . . . 9
2.2.6 Heurísticas . . . 9
2.2.6.1 Motivos para emplear métodos heurísticos . . . 9
2.2.6.2 Características de un buen algoritmo heurístico . . . 10
2.2.7 Matehurísticas . . . 10
2.2.7.1 Criterios de Evaluación para Métodos Metaheurísticos . . . 11
2.3 Referencias Relevantes . . . 11

3 Materiales y Métodos 13

4 Resultados 15

5 Conclusiones **17**

Apéndices **21**

Índice de figuras

Índice de tablas

Índice de fórmulas

2.1 Función objetivo TSP: minimizar el costo total del recorrido 8
2.2 Restricción de salida única de cada nodo para TSP 8
2.3 Restricción de entrada única a cada nodo para TSP 8
2.4 Restricción de eliminación de subciclos y variables binarias para TSP 8

Capítulo 1

Introducción

Capítulo 2

Marco Teórico

2.1. Antencedentes

2.2. Conceptos Clave

2.2.1. El Problema en Optimización

Los problemas de optimización se dividen de manera natural en dos categorías: problemas de optimización con variables continuas y problemas de optimización con variables discretas. A estos últimos se les llama problemas de optimización combinatoria [2].

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.

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:

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

[2]

2.2.1.1. Optimización Continua

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} [2].

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 [2].

2.2.2. Complejidad Computacional y Algorítmica

La complejidad computacional estudia los recursos necesarios para resolver un problema, independientemente del algoritmo utilizado. A diferencia de la complejidad algorítmica,

que se enfoca en el tiempo o espacio que un algoritmo específico consume [5].

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 [5].

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 [5].

Por contraste, un problema es *decidible* cuando existe, o al menos podría existir, algún algoritmo capaz de resolverlo. En otras palabras, un problema indecidible es aquel respecto al cual no se sabe, ni se puede llegar a saber, si se detendrá o cuándo se detendrá [5].

Un problema es decidible cuando es compresible; la compresibilidad se refiere a la existencia de un algoritmo o una fórmula mediante los cuales un problema, un lenguaje o un programa pueden ser condensados, reducidos, comprimidos o expresados. Por el contrario, los problemas indecidibles son incompresibles [5].

2.2.3.1. Clases de Complejidad Básicas

Los problemas decidibles se dividen en dos categorías: los problemas P y los problemas NP [5].

Clase P:

Se dice que un problema D pertenece a la clase de complejidad P si cumple con los siguientes criterios:

- D es un problema de decisión.
- D se puede resolver en tiempo polinomial.

El término "tiempo polinomial" indica que el problema puede resolverse mediante un algoritmo que se ejecuta en un tiempo relativamente rápido. De manera formal, que un algoritmo para D, cuyo comportamiento depende del tamaño de la entrada n, corra en tiempo polinomial significa que:

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

donde $p(n)$ es un polinomio.

Esto puede expresarse utilizando la notación habitual:

$$D = O\left(n^k\right).$$

Así, se dice que un algoritmo D corre en tiempo polinomial.

Se considera que un algoritmo que se ejecuta en tiempo polinomial es "eficiente"; sin embargo, un proceso limitado por un polinomio de grado muy elevado (por ejemplo, 10,000) difícilmente puede considerarse rápido o eficiente. En general, los problemas que se resuelven en tiempo polinomial lo hacen con un grado relativamente pequeño. Un ejemplo de este tipo de problema es la multiplicación de dos números naturales, que se abordó previamente en este artículo, donde el tiempo de ejecución estaba acotado por un polinomio de grado uno [3].

Clase NP:

Se dice que un problema E pertenece a la clase de complejidad NP si cumple con los siguientes criterios:

- E es un problema de decisión.
- E es verificable en tiempo polinomial.

El término "verificar" se refiere a comprobar si el certificado emitido por el algoritmo satisface los requisitos del problema. Un problema se encuentra en la clase NP si el algoritmo utilizado para verificar su certificado se ejecuta en tiempo polinomial.

Como ejemplo de un problema que pertenece a la clase de complejidad NP se puede mencionar el problema de ordenamiento tratado anteriormente, en el que se mostró que la verificación del resultado está acotada por un polinomio de grado uno.

Por otra parte, entre todos los algoritmos de ordenamiento, se observa que mientras algunos se ejecutan en tiempo polinomial, como el conocido *bubble sort*, en principio podría definirse un algoritmo de ordenamiento que opere en tiempo exponencial. Sin embargo, basta con que exista al menos un algoritmo que lo resuelva en tiempo eficiente para que dicho problema pertenezca a la clase P [3].

Surge aquí una dificultad fundamental relacionada con los problemas P y NP. Esta cuestión puede plantearse de varias formas: ¿existen más problemas en P que en NP, o sucede lo contrario? Es decir, ¿puede decirse que $P \geq NP$ o que $NP \geq P$? También se ha propuesto pensar esta relación en términos de pertenencia o inclusión: si los problemas P están contenidos en NP, como en $P \in NP$ o $P \subseteq NP$, o si la relación inversa podría ser cierta, es decir, $NP \in P$. De manera más radical, la cuestión se resume en determinar si ambos conjuntos son iguales o distintos: $P = NP$ o bien $P \neq NP$ [5].

2.2.3.2. NP-Completo y NP-Difícil

Es importante señalar que la clase de problemas NP incluye todos aquellos que pueden resolverse en tiempo polinomial. Sin embargo, los problemas denominados *NP-difíciles*

(o *NP-hard*) son aquellos que se abordan y se resuelven en un tiempo polinomial no determinista. A su vez, los problemas *NP-completos* son aquellos que pertenecen a NP y que, al mismo tiempo, son NP-difíciles.

En este caso, si bien es posible encontrar soluciones a los problemas NP-completos, no se pueden ubicar con precisión en términos de una solución rápida. De manera más exacta, el tiempo necesario para resolver este tipo de problemas crece tan rápidamente como lo hace el tamaño del propio problema. En términos simplificados, a diferencia de otros casos, el análisis general no resulta posible, y a medida que el problema crece, también aumenta proporcionalmente el tiempo requerido para su solución.

Los problemas NP-difíciles pueden presentarse como problemas de decisión, problemas de búsqueda o problemas de optimización [5].

2.2.4. Problemas de Optimización Combinatoria

De manera complementaria a la definición formal presentada en la Subsubsección 2.2.1.2 por [2], Papadimitriou y Steiglitz [7] indica 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 con Capacidades y Ventanas de Tiempo (CVRPTW)** se ha consolidado como uno de 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 [4].

Según [8], el Problema del Agente Viajero se define sobre una red $G = [N, A, C]$, donde N es el conjunto de nodos, A es el conjunto de arcos y $C = [c_{ij}]$ 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) A continuación, se describe la formulación matemática más común:

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

sujeto a:

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

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

$$\begin{cases} u_i - u_j + N x_{ij} \leq N - 1 & \forall i \neq j,\ i, j = 2, \ldots, N \\ x_{ij} \in \{0, 1\} & \forall i, j \end{cases} \tag{2.4}$$

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 [8].

La función objetivo (2.1) proporciona la distancia total de los arcos que conforman un tour. Las restricciones en (2.2) garantizan que, al salir de una ciudad, sólo sea posible dirigirse a un único nodo. Las restricciones en (2.3) aseguran que a cada ciudad únicamente se llegue una sola vez. La restricción (2.4) es fundamental en el planteamiento, pues mediante ella se evita que cualquier conjunto de x_{ij} que forme una subruta sea factible, de modo que sólo los conjuntos que constituyen un recorrido completo sean viables [8].

Complejidad del TSP Como se explica en [7], el Problema del Agente Viajero es NP-hard, y su versión de decisión es NP-completa. Esto implica que la búsqueda de un algoritmo eficiente para resolverlo es un desafío fundamental en la teoría de la computación, pues una solución polinomial para el TSP resolvería eficientemente toda la clase NP. Este carácter NP-hard justifica la utilización de algoritmos aproximados y metaheurísticos en su resolución, dado que las técnicas exactas sólo resultan prácticas para instancias de tamaño reducido.

2.2.4.2. Problema de Ruteo de Vehículos con Capacidades (CVRP)

2.2.4.3. Problema de Ruteo de Vehículos con Capacidades y Ventanas de Tiempo (CVRPTW)

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 [6]), 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" [6].

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* [6].

Para ilustrar este concepto, cabe mencionar la definición propuesta por J.P. Kelly et al., quien es citado por [6]:

> *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

Capítulo 4

Resultados

Capítulo 5

Conclusiones

Bibliografía

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

[2] 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.

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

[4] 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.

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

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

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

[8] 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.

Apéndices