UNIVERSIDAD NACIONAL DE PIURA

ESCUELA DE POSTGRADO
SECCIÓN DE CIENCIAS

PROGRAMA DE MAESTRÍA EN CIENCIAS, CON MENCIÓN EN MATEMÁTICA APLICADA

TÍTULO

METAHEURÍSTICA PARA EL PROCESO DE ENTREGA ÓPTIMA DE PRODUCTOS MEDIANTE RUTEO DE VEHÍCULOS APLICADO EN ZONAS DE EMERGENCIA

TESIS

PARA OPTAR EL GRADO ACADÉMICO DE MAESTRO EN CIENCIAS, CON MENCIÓN EN MATEMÁTICA APLICADA

LIC. GIANCARLO VLADIMIR NAVARRO CASTRO

PIURA – PERÚ

JULIO – 2021

UNIVERSIDAD NACIONAL DE PIURA

ESCUELA DE POSTGRADO
SECCIÓN DE CIENCIAS

PROGRAMA DE MAESTRÍA EN CIENCIAS, CON MENCIÓN EN MATEMÁTICA APLICADA

TÍTULO

METAHEURÍSTICA PARA EL PROCESO DE ENTREGA ÓPTIMA DE PRODUCTOS MEDIANTE RUTEO DE VEHÍCULOS APLICADO EN ZONAS DE EMERGENCIA

TESIS

PARA OPTAR EL GRADO ACADÉMICO DE MAESTRO EN CIENCIAS, CON MENCIÓN EN MATEMÁTICA APLICADA

LIC. GIANCARLO VLADIMIR NAVARRO CASTRO

PIURA – PERÚ

JULIO – 2021

UNIVERSIDAD NACIONAL DE PIURA

ESCUELA DE POSTGRADO
SECCIÓN DE CIENCIAS

PROGRAMA DE MAESTRÍA EN CIENCIAS, CON MENCIÓN EN MATEMÁTICA APLICADA

TÍTULO

METAHEURÍSTICA PARA EL PROCESO DE ENTREGA ÓPTIMA DE PRODUCTOS MEDIANTE RUTEO DE VEHÍCULOS APLICADO EN ZONAS DE EMERGENCIA

APROBADA EN ESTILO Y CONTENIDO POR

Dr. ETHELWOLDO LÓPEZ COBA
Presidente Jurado de Tesis

MSc. JUAN ESPINOZA ARÉVALO
Secretario Jurado de Tesis

MSc. ANDRÉS AUGUSTÍN CASTILLO MOSCOL
Vocal Jurado de Tesis

UNIVERSIDAD NACIONAL DE PIURA

ESCUELA DE POSTGRADO
SECCIÓN DE CIENCIAS

PROGRAMA DE MAESTRÍA EN CIENCIAS, CON MENCIÓN EN MATEMÁTICA APLICADA

TÍTULO

METAHEURÍSTICA PARA EL PROCESO DE ENTREGA ÓPTIMA DE PRODUCTOS MEDIANTE RUTEO DE VEHÍCULOS APLICADO EN ZONAS DE EMERGENCIA

EL SUSCRITO DECLARO QUE ESTE TRABAJO DE TESIS ES ORIGINAL EN SU CONTENIDO Y FORMA

Lic. GIANCARLO VLADIMIR NAVARRO CASTRO
Ejecutor

Dr. FLABIO ALFONSO GUTIERREZ SEGURA
Patrocinador

DEDICATORIA

Con toda la humildad que mi corazón puede emanar, dedico principalmente este trabajo a Dios todo poderoso, el que me ha dado fortaleza para continuar cuando a punto de caer he estado y llevarme de la mano por el camino del bien.

De igual forma, dedico esta tesis, a mis hijas Saylim Damara y Ashley Dariana, las causantes de mi anhelo de salir adelante, progresar y culminar con éxito esta tesis, dedico a ellas cada esfuerzo que realicé en la construcción de esta; agradezco a Dios por darme tan hermosa compañía y motivación para cada día ser mejor.

También dedico esta tesis a mis padres, por haberme forjado como la persona que soy, fundamentalmente soy quien soy gracias a ustedes.

Giancarlo Vladimir Navarro Castro

PRÓLOGO

La elaboración de un plan de ruteo de vehículos para un vehículo que debe abastecer dos o tres clientes puede parecer para muchas personas innecesario, ya sea porque la respuesta a esta es trivial o no representa grandes variaciones en el presupuesto. Esta situación no sucede si se trata de una flota de vehículos donde la dificultad de hallar una ruta óptima es más complicada y la minimización de gastos es ahora un factor importante, la búsqueda de una solución óptima se vuelve complicada, ya que los problemas de ruteo de vehículos son clasificados como problemas combinatorios, pero es importante y necesario elaborar un plan de ruteo que permita minimizar gastos de transporte como: combustible, desgaste de neumáticos, tiempo de reparto, pago de personal, etc.

La idea de implementar un modelo de ruteo de vehículos para el reparto de productos en lugares de emergencia resulta necesaria, si bien la minimización de gastos de transporte no es lo primordial, pero sí la de minimizar el tiempo de reparto a las zonas afectadas. El modelo de ruteo de vehículos permitirá la distribución oportuna de productos de primera necesidad como: agua,

medicinas, productos enlatados; reduciendo muertes por falta de alimentos y atención médica. El Instituto Nacional de Defensa Civil (INDECI) sugiere un tiempo de respuesta no mayor a 72 horas.

AGRADECIMIENTOS

A través de estas líneas quiero expresar mi más sincero agradecimiento a todas las personas que con su apoyo han colaborado en la realización de esta tesis.

Quiero expresar mi más sincero agradecimiento, a mi asesor de tesis, el Dr. Flabio Gutierrez Segura, por su paciencia, generosidad al brindarme la oportunidad de recurrir a su capacidad y experiencia científica en un marco de confianza, afecto y amistad, fundamentales para la concreción de este trabajo.

Asimismo, agradezco infinitamente al Dr. José Rodríguez Melquiades, quien estuvo guiándome académicamente con su experiencia y profesionalismo.

Quiero agradecer de igual manera al Fondo Nacional de Desarrollo Científico y Tecnológico (**FONDECYT**), por el financiamiento de esta tesis mediante el proyecto (**125-2018-FONDECYT-BM-IADT-AV**), **MODELOS Y META HEURÍSTICAS PARA EL RUTEAMIENTO DEL PROCESO DE DISTRIBUCIÓN DE PRODUCTOS Y COLECTA TRANSPORTE DE RESIDUOS SÓLIDOS EN EL CONTEXTONDE LA LOGÍSTICA URBANA.**

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

RESUMEN

Este trabajo aborda el problema de ruteo de vehículos para la distribución de productos aplicado en zonas de emergencia, dada la complejidad computacional de este tipo de problemas y la necesidad de obtener respuestas en un tiempo limitado, se implementan computacionalmente dos metaheurísticas: Los Algoritmos Inspirados en Colonias de Hormigas (**ACO**) y Algoritmos Genéticos (**AG**). Finalmente se evalúan estas metaheurísticas y se comparan con los resultados obtenidos del modelo de programación lineal mixta implementados en **GLPK**, con datos recolectados de la ciudad de Piura, frente a un eventual fenómeno del niño costero, como el ocurrido en el año 2017, concluyendo que los Algoritmos Inspirados en Colonias de Hormigas muestran mejores resultados. *Python* es el lenguaje de programación en el que se han implementado estas metaheurísticas.

Palabras Clave

Ruteo de vehículos, optimización, heurísticas, metaheurísticas, algoritmos genéticos, algoritmos inspirados en colonias de hormigas.

ABSTRACT

This work addresses the problem of vehicle routing for the distribution of products applied in emergency zones, given the computational complexity of this type of problems and the need to obtain answers in a limited time, two metaheuristics are computationally implemented: Algorithms Inspired in Ant Colonies (**ACO**) and Genetic Algorithms (**AG**). Finally, these metaheuristics are evaluated and compared with the results obtained from the mixed linear programming model implemented in **GLPK**, with data collected from the city of Piura, in the face of an eventual phenomenon of the coastal child, such as the one that occurred in the year 2017, concluding that Ant Colonies Inspired Algorithms show better results. *Python* is the programming language in which these metaheuristics have been implemented

Keywords

Vehicle routing, optimization, heuristics, metaheuristics, genetic algorithms, algorithms inspired by ant colonies.

INTRODUCCIÓN

Nuestro mundo está lleno de amenazas de desastres naturales. En particular, hay un número creciente de desastres graves ocurridos en los últimos años, como el terremoto de magnitud 7.5 y tsunami ocurrido en Indonesia el 28 de septiembre del 2018, el huracán Katrina - Estados Unidos el 2005, el terremoto ocurrido el 15 de agosto de 2007 en la provincia de Pisco o el fenómeno del niño costero en el verano del 2017 que afectó al norte del Perú, generando el desborde del río Piura e inundando muchos caseríos y centros poblados.

Desde un punto de vista matemático, las operaciones de alivio en caso desastres implican una amplia variedad de problemas complejos de optimización, por ejemplo, localización de lugares para reparto óptimo de ayudas a los damnificados, rutas óptimas para el reparto de productos de primera necesidad; este último es un problema de ruteo de vehículos. Los problemas de localización, ruteo de vehículos son problemas de optimización combinatoria.

En general, un problema de optimización consiste en encontrar una solución factible que maximice (minimice) una función objetivo. Al enfrentarse con un problema de optimización, la primera preocupación de los científicos de matemática y computación es: si el problema puede resolverse mediante algunos algoritmos eficientes (y en particular en tiempo polinomial). Muchos problemas de optimización combinatoria son NP-duros, es decir no se pueden resolver en un tiempo polinomial. Además, en condiciones de emergencia, las soluciones a los problemas deben desarrollarse en un tiempo muy limitado. Por lo tanto, los métodos exactos tradicionales tienen una aplicabilidad limitada.

Las metaheurísticas son una clase de métodos de optimización que debido a su simplicidad, flexibilidad y aplicabilidad general resultan adecuados para abordar una amplia gama de problemas computacionalmente intratables. No siempre garantizan la solución óptima exacta en una única ejecución, pero en la mayoría de los casos pueden obtener soluciones satisfactorias en un tiempo de cómputo aceptable.

La investigación sobre los métodos de optimización para las operaciones de alivio en casos de desastre, en comparación con otras subáreas de investigación de operaciones, es limitada, pero, ha comenzado a evolucionar en los últimos años. Una variedad de metaheurísticas (Khouadjia et al., 2013), incluidos el algoritmo genético (GA), la optimización de enjambre de abejas, la optimización de colonias de hormigas, han encontrado un número creciente de aplicaciones en problemas de ruteo de vehículo. Este trabajo se enfocará en la aplicación de dos metaheurísticas: algoritmos genéticos y metaheurísticas inspiradas en colonias de hormigas, además se utilizará una herramienta computacional para resolver el modelo, que es un problema de programación lineal entera mixta.

CAPÍTULO I

PRELIMINARES

Este capítulo contiene los conceptos teóricos que dan sustento al desarrollo de la investigación que se presenta en esta tesis: Algoritmos, teoría de grafos, optimización, programación lineal y algunas variantes de problemas de ruteo de vehículos.

1.1. Algoritmos

Los modelos de ruteo de vehículo son por lo general problemas de optimización combinatoria, es decir, a medida que la complejidad del espacio de búsqueda incrementa, el costo de ejecución de dichos algoritmos puede aumentar de forma exponencial, convirtiendo la resolución en prácticamente imposible.

Definición 1.1.1. *Un algoritmo es un conjunto de instrucciones o reglas definidas y no ambiguas, ordenadas y finitas que permite, solucionar un problema, realizar un cómputo, procesar datos y llevar a cabo otras tareas o actividades. Dados un estado inicial y una entrada, siguiendo los pasos sucesivos se llega a un estado final y se obtiene una solución.*

Una instrucción es no ambigua cuando la acción a ejecutar está perfectamente definida.

1.1.1. Eficiencia y complejidad de algoritmos

Cuando se tiene que resolver un problema, es posible disponer de varios algoritmos. Evidentemente, se desearía seleccionar el mejor. Esto plantea la pregunta, *¿cómo decidir entre varios algoritmos cuál es el más beneficioso?*. Si se tuviera que resolver unos cuantos casos pequeños de un problema sencillo, quizá no importe demasiado que algoritmo emplear; en este caso se podría

seleccionar el que sea más fácil de programar, sin preocupación por sus propiedades teóricas. Sin embargo, si se tiene que resolver muchos casos, o si el problema es difícil, quizá se tenga que seleccionar de forma más cuidadosa (Brassard et al., 1997).

Definición 1.1.2. ***eficiencia algorítmica*** *es usado para describir aquellas propiedades de los algoritmos que están relacionadas con la cantidad de recursos utilizados por el algoritmo. Un algoritmo debe ser analizado para determinar el uso de los recursos que realiza. un algoritmo es eficiente si, cuando implementado, se ejecuta rápidamente en instancias de entrada reales.*

Entre otras definiciones se puede considerar que un algoritmo es eficiente si su consumo de recursos está en la media o por debajo de los niveles aceptables. "aceptable" significa: que el algoritmo se ejecuta en un tiempo razonable en un computador. Desde mediados del siglo XX hasta la actualidad las computadoras han tenido un avance significativo tanto para procesar datos como en la capacidad de memoria disponible, lo que indica que los niveles aceptables de eficiencia hace unas décadas serían inaceptables en la actualidad.

Existen muchas maneras para medir la cantidad de recursos empleados por un algoritmo: las dos medidas más frecuentes son la complejidad temporal y espacial; otras medidas a tener en consideración podrían ser la velocidad de transmisión, uso temporal del disco duro, así como uso del mismo a largo plazo, consumo de energía, tiempo de respuesta ante los cambios externos, etc. Muchas de estas medidas dependen del tamaño de la entrada del algoritmo; además podrían depender de la forma en que los datos están organizados.

Definición 1.1.3. ***La complejidad de los algoritmos*** *estudia formalmente la dificultad de los problemas algorítmicos. Así, se definen varias clases de complejidad (**P**, **NP**, etc) pudiéndose entonces clasificar los algoritmos según sus características.*

1.1.2. Consumo de tiempo de los algoritmos

Diremos que un algoritmo resuelve un problema si al recibir la descripción de una instancia del problema devuelve una solución de la instancia o informa que la instancia no tiene solución. Para cada instancia del problema, el algoritmo consume una cantidad de tiempo diferente. Digamos que el algoritmo consume $T(I)$ unidades de tiempo para procesar la instancia I. La relación entre $T(I)$ y el tamaño de la instancia I proporciona una medida de la eficiencia del algoritmo. Por ejemplo, es más difícil ordenar la lista $\{3, 5, 2, 9, 1\}$ que la lista $\{1, 3, 5, 6, 7\}$, como la segunda ya está ordenada. Lo que se quiere es tal caracterización de la calidad de un algoritmo que nos facilita hacer comparaciones entre algoritmos.

1.1.2.1. Conceptos de peor caso y mejor caso

Dado un algoritmo A para un problema y un número natural n, sea:

$$T^*(n) = \text{máx}\{T(I)/I \text{ es instancia de tamaño } n \text{ para el problema}\}$$

Decimos que $T^*(n)$ mide el consumo de tiempo del algoritmo A en el peor caso; en este caso también diremos que el algoritmo realiza el mayor esfuerzo.

De modo similar podemos definir

$$T_*(n) = \text{mín}\{T(I)/I \text{ es instancia de tamaño } n \text{ para el problema}\}$$

El cual mide el consumo de tiempo en el mejor caso, es decir cuando el algoritmo realiza el menor esfuerzo.

1.1.3. Análisis asintótico

La meta del análisis de algoritmos es evaluar la calidad de un algoritmo en comparación con otros algoritmos o en comparación a la complejidad del problema o alguna cota de complejidad conocida. Sin embargo, típicamente en el conteo de "pasos de computación" falta precisión en el sentido que no es claro que cosas se considera operaciones básicas. Por eso normalmente se caracteriza la calidad de un algoritmo por la clase de magnitud de la función de complejidad y no la función exacta misma. Tampoco son interesantes los tiempos de computación para instancias pequeñas, sino las instancias grandes. En una instancia pequeña, normalmente todos los algoritmos producen resultados rápidamente (Schaeffer, 2011).

Para definir clases de magnitudes, necesitamos las definiciones siguientes. Para funciones $f : \mathbb{Z}^+ \to \mathbb{R}$ y $g : \mathbb{Z}^+ \to \mathbb{R}$.

- $f(n) \in \mathcal{O}(g(n))$ si $\exists\ c > 0$ tal que $|f(n)| \leq c \cdot |g(n)|$ para valores de n suficientemente grandes.
- $f(n) \in \Omega(g(n))$ si $\exists\ c > 0$ tal que $|f(n)| \geq c \cdot |g(n)|$ para valores de n suficientemente grandes.
- $f(n) \in \Theta(g(n))$ si $\exists\ c, c' > 0$ tal que $c \cdot |g(n)| \leq |f(n)| \leq c' \cdot |g(n)|$ para valores de n suficientemente grandes.
- $f(n) \in o(g(n))$ si $\lim\limits_{n\to\infty} \dfrac{f(n)}{g(n)} = 0$.

La función $\mathcal{O}(f(n))$ es una cota superior asintótica al tiempo de ejecución, mientras la función $\Omega(f(n))$ es una cota inferior asintótica. La tercera definición quiere decir que las dos funciones crecen asintóticamente iguales.

Por lo general, $\mathcal{O}(f(n))$ es la notación de complejidad asintótica más comúnmente utilizado.

Dadas dos funciones $f(n)$ y $g(n)$ diremos que $f(n)$ es más compleja que $g(n)$ cuando

$$\lim_{n\to\infty}\left(\frac{f(n)}{g(n)}\right) = \infty$$

Diremos que $f(n)$ es menos compleja que $g(n)$ cuando

$$\lim_{n\to\infty}\left(\frac{f(n)}{g(n)}\right) = 0$$

Y diremos que $f(n)$ es equivalente a $g(n)$ cuando

$$\lim_{n\to\infty}\left(\frac{f(n)}{g(n)}\right) = K$$

siendo $K \neq 0$ y $K \neq \infty$

Nótese que esta definición nos permite un análisis algorítmico conociendo la formulación de la función.

1.1.4. Orden de complejidad

Tenemos los siguientes órdenes de complejidad:

- Constante: $\mathcal{O}(1)$.
- Lineal: $\mathcal{O}(n)$.
- Logarítmica: $\mathcal{O}(\log n)$.
- Cuadrada: $\mathcal{O}(x^2)$, Cúbica: $\mathcal{O}(x^3)$, ..., de orden k: $\mathcal{O}(x^k)$.
- Combinación en producto de distintos tipos de complejidades.
  Por ejemplo, cuasilineal $\mathcal{O}(n \log n) = \mathcal{O}(n) \times \mathcal{O}(\log n)$.
- Exponencial: $\mathcal{O}(k^n)$. Por ejemplo: $\mathcal{O}(2^n)$, $\mathcal{O}(3^n)$, ...
- Factorial: $\mathcal{O}(n!)$.

El hecho de que exista un caso de un problema en el cual la solución es inmediata, no provoca que el coste de este algoritmo sea constante. Nunca podemos considerar el coste de un algoritmo en el caso más fácil puesto que este puede estar sujeto a ciertas condiciones que no se pueden dar siempre.

Existen una serie de reglas de complejidad a tener en cuenta, que facilitan la tarea de calcular el orden de complejidad:

- Se ignoran las constantes multiplicativas o aditivas.
- La base del logaritmo no importa.
- Suma de complejidades es el máximo de entre todas ellas.
- Composición de complejidades es el producto de las complejidades.

La complejidad que se usa es la del algoritmo menos complejo que pueda resolver el problema. Evidentemente algoritmos más complejos pueden resolver problemas, pero se buscará el algoritmo menos complejo.

Los problemas según puedan ser resueltos, es decir, lograr una solución óptima, se pueden clasificar según la complejidad de los algoritmos que la logran.

Si la complejidad de los algoritmos es de las nombradas anteriormente, excepto la exponencial, se dice que los problemas que se resuelven son polinómicos, (categoría P). Si no somos capaces de resolverlos por algoritmos de esa complejidad, es decir, no podemos encontrar la solución en un tiempo razonable, pertenecen a la categoría NP (Non deterministic Polynomial time).

1.1.5. Clases de Complejidad

1.1.5.1. Complejidad de Clase P

Contiene aquellos problemas de decisión que pueden ser resueltos en tiempo polinómico por un algoritmo determinístico. Los problemas de complejidad polinómica son tratables ya que se pueden resolver en la práctica en tu tiempo razonable. Aquellos problemas para los que la mejor solución que se conoce es de complejidad superior a la polinómica, se dice que son problemas intratables.

1.1.5.2. Complejidad de Clase NP

NP(non deterministic Polinomial Time) es una de las clases más fundamentales, contiene los problemas de decisión que son resueltos por un algoritmo no determinístico en tiempo polinómico. Algunos de estos problemas intratables pueden caracterizarse por el curioso hecho de que puede aplicarse un algoritmo de verificación de tiempo polinómico para comprobar si una posible solución es válida o no. Esta característica lleva a un método de resolución no determinística consistente en aplicar heurísticas para obtener soluciones que se van desestimando a ritmo polinómico.

La clase P se encuentra contenida en NP, pero NP contiene muchos problemas importantes, llamados también NP-Completos, para el cual no se conoce algoritmos de tiempo polinomial.

1.1.5.3. Complejidad de Clase NP-Completos

Se conoce una amplia variedad de problemas de tipo NP, de los cuales destacan algunos de ellos de extrema complejidad. Son problemas NP y son los peores problemas posibles de clase NP y es probable que no formen parte de la clase de complejidad P. NP-completo es el conjunto de los problemas de decisión en NP tal que todo problema en NP se puede reducir en cada uno de los problemas de NP-completo.

El problema de suma de subconjuntos puede ser un buen ejemplo ya que la verificación de alguna respuesta es fácil, pero no se conoce mejor solución que explorar todos los $2^n - 1$ subconjuntos posibles hasta encontrar uno que cumpla con la condición.

Definición 1.1.4. *Un problema de decisión C es NP-completo si:*

1. $C \in NP$
2. *Todo problema de NP se puede transformar polinomialmente en C.*

Un problema que satisface la segunda condición, pero no la primera se le denomina NP-duro

1.1.5.4. Complejidad de Clase NP-Duro

La Clase NP-Duro es el conjunto de los problemas de decisión que contiene los problemas M tales que todo problema L en NP puede ser transformado polinomicamente en M. Esta clase puede ser descrita como los problemas de decisión que son al menos tan difíciles como un problema de NP.

Asumiendo que M es NP-completo, es decir:

1. L está en NP.
2. $\forall L'$ en $NP, L' \leq L$.

En el conjunto NP-Duro se asume que L satisface la propiedad 2, pero no la propiedad 1

La relación entre las clases de complejidad NP y P es una pregunta por primera vez formulada por el científico computacional Stephen Cook que la teoría de la complejidad computacional aún no ha podido responder. En esencia, la pregunta ¿es P = NP completo? significa: si es posible "verificar" rápidamente soluciones positivas a un problema del tipo SI/NO (donde "rápidamente" significa "en tiempo polinómico"), ¿es que entonces también se pueden "obtener" las respuestas rápidamente? Se considera el problema más importante en este campo, el Clay Mathematics Institute ha ofrecido un premio de un millón de dólares estadounidenses para quien desarrolle la primera demostración correcta.

1.2. Teoría de Grafos

1.2.1. Introducción a la teoría de grafos

La teoría de grafos tuvo inicio con el problema de los puentes de Königsberg, también llamado problema de los siete puentes de Königsberg, es un célebre problema matemático, resuelto por Leonhard Euler en 1736 y cuya resolución dio origen a la teoría de grafos.

Esta ciudad es atravesada por el río Pregel, en ruso «Pregolya», el cual se bifurca para rodear con sus brazos a la isla Kneiphof, dividiendo el terreno en cuatro regiones distintas, las que entonces estaban unidas mediante siete puentes llamados Puente del herrero, Puente conector, Puente verde, Puente del mercado, Puente de madera, Puente alto y Puente de la miel. El problema fue formulado en el siglo $XVIII$ y consistía en encontrar un recorrido para cruzar a pie toda la ciudad, pasando sólo una vez por cada uno de los puentes, y regresando al mismo punto de inicio.

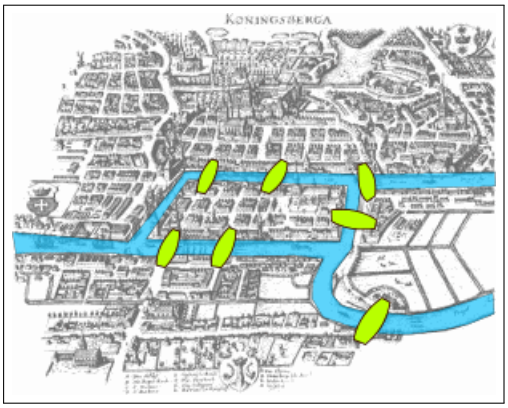


Figura 1.1: Mapa de Konigsberg en la época de Leonhard Euler.

Fuente: https://es.wikipedia.org/wiki/Problema_de_los_puentes_de_Königsberg.

1.2.2. Grafos

Los grafos se presentan con frecuencia en la vida real, tal es el caso de una red de carreteras que enlace un cierto grupo de ciudades; aquí los nodos de la red o ciudades representan los vértices del grafo, las carreteras que unen las ciudades representan los arcos o aristas; así a cada arco se asocia una información tal como la distancia entre ciudades, consumo de gasolina, costo de mantenimiento, etc (Caicedo et al., 2010).

Definición 1.2.1. ***Un grafo****, es un par $G = (V; E)$ de conjuntos que satisfacen $E \subseteq [V]^2$; Así, los elementos de E son subconjuntos de 2 elementos de V. Para evitar ambigüedades notacionales, siempre supondremos tácitamente que $V \cap E = \emptyset$; Los elementos de V son los vértices, nodos, o puntos del grafo G, los elementos de E son las aristas (o líneas).*

Nota 1.2.1. *La forma habitual de representar un grafo es dibujando un punto o círculo para cada vértice y uniendo dos de estos elementos por una línea. La forma en que se dibujan estos es irrelevante: lo único que importa es la información de qué pares de vértices forman una arista y cuáles no.*

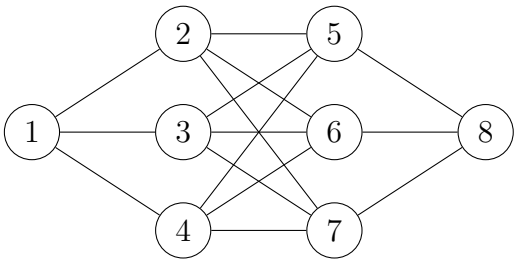


Figura 1.2: Ejemplo de grafo con $V = \{1, 2, 3, 4, 5, 6, 7, 8\}$

Definición 1.2.2. *Un vértice v incide con una arista e, si $v \in e$; entonces e es una arista en v. Los dos vértices que inciden con una arista son llamados extremos.*

Definición 1.2.3. *En un grafo, dos vértices son adyacentes, o vecinos, si están conectados por una arista. En tal caso, cada uno de estos vértices son incidentes a dicha arista.*

Definición 1.2.4. ***Grafo Dirigido*** *Si en un grafo los arcos tienen una dirección, el grafo se llama grafo dirigido u orientado.*

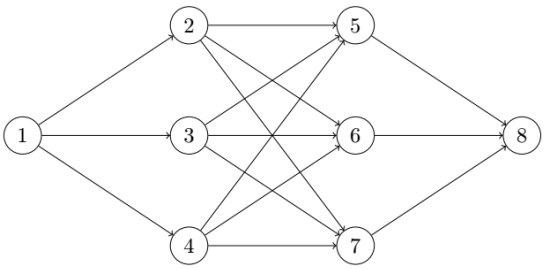


Figura 1.3: Ejemplo de grafo dirigido.

En un grafo orientado cada arco se representa por medio de un par ordenado (v_i, v_j) donde el primer elemento es el nodo origen o fuente y el segundo es el nodo destino de ese arco, por lo tanto se puede decir que el arco va desde v_i hasta v_j y que v_j es adyacente a v_i. Un grafo orientado también se llama un digrafo. Si los arcos del grafo no indican una dirección, el grafo es no dirigido y en él cada arco se puede representar nombrando los nodos que lo forman sin importar el orden es decir:

$$(v_i, v_j) = (v_j, v_i)$$

Los nodos de un grafo se pueden usar para representar los objetos y los arcos para representar relaciones entre esos objetos. En el ejemplo anterior los nodos podrían ser ciudades y los arcos las rutas aéreas entre esas ciudades. A los nodos y arcos de un grafo se pueden asignar nombres o valores (etiquetas), creándose así lo que se llama un grafo etiquetado.

Definición 1.2.5. *Una arista no dirigida es arista una donde no se distingue un vértice inicial ni uno final.*

Definición 1.2.6. *Una arista dirigida es una arista que tiene una dirección asociada consigo, esto es, posee un vértice inicial y un vértice final.*

Definición 1.2.7. *Sea $G = (V, E)$ un grafo, un subgrafo de G es cualquier grafo $H = (W, F)$, de modo que $W \subseteq V$ y $F \subseteq E$. Un subgrafo se obtiene eliminando algunas aristas y/o vértices. Si se suprime un vértice, se suprimen todas las aristas que tienen por origen o fin dicho vértice. Si F contiene todos los lados*

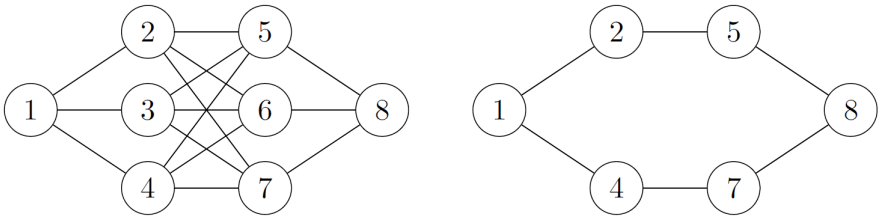


Figura 1.4: Ejemplo de subgrafo.

Definición 1.2.8. *El número de nodos de un grafo se llama orden del grafo y se denota como:*

$$ord(g) = |N(G)| = N$$

El número de aristas o arcos es $|A(g)| = A = e(G)$

Otra notación: G^n = grafo de orden n.

Definición 1.2.9. ***Grado de un Vértice*** *El grado de un vértice x es el número de vértices adyacentes a él, esto es, el número de arcos incidentes a x y se denota $grad(x)$.*

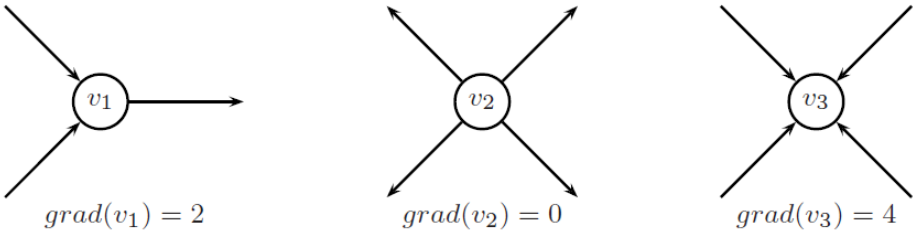


Figura 1.5: Visualización gráfica del grado de un vértice

Fuente: Caicedo (2010) Introducción a la teoría de grafos.

Definición 1.2.10. *Un grafo completo, es un grafo simple donde cada par de vértices está conectado por una arista.*

Un grafo completo de n vértices tiene $n(n-1)/2$ aristas, y se nota K_n. Es un grafo regular con todos sus vértices de grado $n-1$.

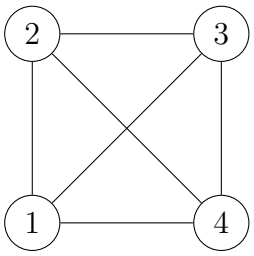


Figura 1.6: Ejemplo de grafo completo K_4

Definición 1.2.11. *Una* ***trayectoria*** *T en un grafo es una secuencia de nodos $v_1, v_2, \ldots, v_n$ tal que $(v_1, v_2), (v_2, v_3), \ldots, (v_{n-1}, v_n)$ son arcos.*

Cuando los vértices de una trayectoria son todos distintos, excepto posiblemente el primero y el último, la trayectoria se define como una trayectoria simple. Una trayectoria que comienza y termina en el mismo nodo se llama un ciclo o circuito. Los ciclos de longitud 1 se llaman bucles o lazos porque salen de un nodo e inciden en el mismo nodo sin pasar por ningún otro.

Definición 1.2.12. *Un grafo G se denomina simple o sencillo si cumple:*

1. *No tiene lazos.*
2. *No existe más que un arco para cada par de nodos j.*

Un grafo que no es simple se le llama grafo múltiple o multigrafo.

Definición 1.2.13. *Multigrafo, es el que acepta más de una arista entre dos vértices. Estas aristas se llaman múltiples o lazos (loops en inglés). Los grafos simples son una subclase de esta categoría de grafos.*

Definición 1.2.14. *Sean $G = (V, E)$ y $G' = (V', E')$ dos grafos. Llamamos a G y G' isomorfos, y escribimos $G \simeq G'$, si existe una biyección $\varphi : V \to V'$ con $xy \in E \Leftrightarrow \varphi(x)\varphi(y) \in E'$ para todas las $x, y \in V$, es decir que preserva la relación de adyacencia. Donde φ se llama isomorfismo; Un isomorfismo de un gráfico G en sí mismo se llama un automorfismo de G. Normalmente no distinguimos entre los grafos isomorfos. Por lo tanto, normalmente escribimos $G = G'$ en lugar de $G \simeq G'$.*

Definición 1.2.15. *Sea G un grafo. Se dice que G es conexo, si dados u y v dos vértices de G existe al menos un camino de u a v.*

Definición 1.2.16. *Un **árbol** es un grafo conexo simple acíclico. Algunas veces, un vértice del árbol es distinguido llamándolo raíz.*

Definición 1.2.17. *Un **bosque** es un conjunto de árboles; o de forma equivalente, un bosque es un grafo acíclico.*

1.2.3. Grafos de Euler

Cuando se posee una red no dirigida conexa se puede diseñar un circuito que recorra todos sus nodos, partiendo de un nodo específico y regresando a él, pero sin pasar más de una vez por cada arco, además escogiendo el circuito de forma tal que el recorrido sea mínimo.

Este tipo de problema se resuelve hallando un circuito euleriano de longitud mínima.

Definición 1.2.18. ***Circuito Euleriano*** *en una red conexa, un circuito que recorre todos los nodos y todos los arcos pasando una vez por cada arco, se llama circuito euleriano.*

Si la red no es conexa puede ocurrir que no exista un circuito euleriano.

Definición 1.2.19. ***Trayectoria Euleriana*** *Una trayectoria euleriana es una trayectoria que incluye cada arco exactamente una sola vez.*

Definición 1.2.20. *Una **red par** es aquella en la cual el número de arcos que inciden a todos y cada uno de los nodos es par.*

Corolario 1.2.1. *Un grafo conexo contiene una trayectoria euleriana, pero no un ciclo euleriano, si y solo sí exactamente dos nodos tienen grado impar. La trayectoria debe comenzar en un nodo impar y terminar en otro nodo impar.*

Proposición 1.2.1. *Sea G un grafo. Entonces si G tiene un circuito de Euler, el grado de cada vértice es par, mientras que, si G tiene un camino de Euler, G tiene exactamente dos vértices de grado impar (exactamente los vértices donde empieza y termina el camino).*

Teorema 1.2.1. *Sea G un grafo conexo. Entonces G es un grafo de Euler si, y sólo si, el grado de cada vértice es par.*

Corolario 1.2.2. *Sea G un grafo conexo. Entonces G tiene un camino de Euler si, y sólo si, G tiene exactamente dos vértices de grado impar.*

1.2.4. Grafos de Hamilton

En la sección anterior estudiamos cuando en un grafo podíamos encontrar un camino que recorriera todas las aristas una sola vez. En esta, pretendemos estudiar como recorrer todos los vértices una sola vez.

Cuando se tiene un grafo conexo no dirigido $G = (N, A)$ puede ocurrir que se desea hacer un recorrido por el grafo, que pase por todos los nodos, sin repetir nodo y regresar al nodo de partida. Tal tipo de recorrido se define como un ciclo o circuito hamiltoniano.

El nombre de este tipo de ciclo se ha dado en honor al matemático irlandés Sir William Rowan Hamilton $(1805 - 1865)$, quien fue el primero que los estudió en 1857.

Si el grafo G tiene asociados a sus arcos valores que pueden ser las distancias entre los nodos o el costo de recorrer dichas distancias, es de interés hallar un circuito que recorra todos los nodos una sola vez, y que tenga distancia mínima o costo mínimo. Este problema ha sido formulado desde hace mucho tiempo con el nombre de: "Problema del Agente Viajero".

Definición 1.2.21. *Un **ciclo hamiltoniano** en un grafo es un ciclo que visita cada vértice una y sólo una vez.*

Definición 1.2.22. *Un camino hamiltoniano es una sucesión de aristas adyacentes, que visita todos los vértices del grafo una sola vez. Si además el último vértice visitado es adyacente al primero, el camino es un ciclo hamiltoniano.*

Definición 1.2.23. *Sea G un grafo. Un camino de Hamilton es un camino que recorre todos los vértices una sola vez.*

- *Un circuito de Hamilton es un camino cerrado que recorre todos los vértices una sola vez (salvo los extremos).*
- *Un grafo con un circuito de Hamilton se denomina grafo de Hamilton o grafo hamiltoniano.*

Observación 1.2.1.

- *Puesto que a la hora de buscar un camino o circuito de Hamilton no podemos pasar dos veces por un mismo vértice, no es posible que el camino contenga dos lados paralelos, ni que contenga lazos. Supondremos por tanto en esta sección que todos los grafos que intervienen no tienen ni lazos ni lados paralelos.*
- *Hemos visto en el ejemplo anterior, que, si hay un vértice de grado 1, entonces el grafo no es de Hamilton.*
- *Por otra parte, si un grafo con n vértices es de Hamilton, en el circuito de Hamilton intervienen n lados. Por tanto, un grafo de Hamilton con n vértices tiene al menos n lados.*
- *Intuitivamente, cuantos más lados tenga un grafo con un número de vértices fijado, más fácil será poder encontrar un circuito de Hamilton. Veremos a continuación que, si tenemos el número suficiente de lados, entonces tenemos garantizada la existencia de circuitos de Hamilton.*

1.3. Optimización

La optimización es la acción de desarrollar una actividad lo más eficientemente posible, es decir, con la menor cantidad de recursos y en el menor tiempo posible. La optimización, en general, implica lograr el mejor funcionamiento de algo, usando de la mejor forma los recursos.

Desde el punto de vista matemático, la optimización, es el proceso que consiste en hallar los mínimos y máximos de una función, algunas ramas de la optimización son: (lineal, no lineal, entera, combinatoria, estocástica, multiobjetivo) (Ramos et al., 2010).

Los problemas de optimización se componen generalmente de tres partes:

- **Función objetivo:** Es la medida cuantitativa del funcionamiento del sistema que se desea optimizar (maximizar o minimizar). Como ejemplo de funciones objetivo se pueden mencionar: la minimización del tiempo de ruteo de un vehículo, la maximización de los beneficios netos de venta de ciertos productos, la minimización de emisión de CO_2 de una flota de vehículos, etc.
- **Variables:** Representan las decisiones que se pueden tomar para afectar el valor de la función objetivo. Desde un punto de vista funcional se pueden clasificar en variables dependientes y variables independientes.
- **Restricciones:** Representan el conjunto de relaciones (expresadas mediante ecuaciones e inecuaciones) que ciertas variables están obligadas a satisfacer. Por ejemplo, la capacidad de un vehículo, tiempo de llegada de un vehículo a su depósito, etc.

Resolver un problema de optimización consiste en encontrar el valor que deben tomar las variables para hacer óptima la función objetivo satisfaciendo el conjunto de restricciones.

Los métodos de optimización los podemos clasificar en: métodos clásicos y métodos metaheurísticos (que aparecieron ligados a lo que se denominó inteligencia artificial e imitan fenómenos sencillos observados en la naturaleza). Dentro de los primeros se encuentra la optimización lineal, lineal entera mixta, no lineal, combinatoria, estocástica, etc. En el segundo grupo se incluyen los algoritmos evolutivos, metaheurísticas inspiradas en colonias de hormigas, etc.

1.3.1. Métodos clásicos de optimización

- **Programación Lineal:** Una de las áreas más empleadas de la optimización es la programación lineal (PL). Estos problemas se basan en la optimización (minimización o maximización) de una función lineal conocida como función objetivo, sujeta a una serie de restricciones lineales de igualdad o desigualdad.

  Definición 1.3.1. *Un programa lineal es la maximización o minimización de una función lineal sujeta a restricciones lineales (por ejemplo, ecuaciones lineales o desigualdades lineales).*

  Matricialmente, un problema de PL (con igualdades) se puede expresar como:

$$\begin{aligned} \text{máx} \quad & z = cx \\ \text{s.a.} \quad & Ax = b \\ & x \geq 0 \end{aligned} \tag{1.1}$$

donde cx es la función objetivo a maximizar (minimizar), $x \in \mathbb{R}^n$ representa el vector de variables a determinar, $c \in \mathbb{R}^n$ es el vector de costos asociado a las variables, $A \in \mathcal{M}_{m\times n}$ es la matriz de coeficientes y $b \in \mathbb{R}^m$ el vector de términos independientes asociado a las restricciones. Es decir, el problema (1.1) es equivalente al problema (1.2) en notación extendida:

$$\begin{aligned} \text{máx} \quad & z = c_1x_1 + c_2x_2 + ... + c_nx_n \\ \text{sujeto a } & a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = b_1 \\ & a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n = b_2 \\ & \vdots \qquad\qquad\qquad\quad = \vdots \\ & a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n = b_m \\ & x_1, x_2, \ldots, x_n \geq 0 \end{aligned} \tag{1.2}$$

Definición 1.3.2. *Se llama **región factible** del problema al conjunto de posibles valores que satisfacen todas las restricciones, $\mathcal{R} = \{x \in \mathbb{R}^n : Ax = b, x \geq 0\}$. Una solución del problema (1.1) se dice **solución factible** si satisface todas las restricciones del mismo, es decir, $x \in \mathcal{R}$. Una solución factible se dice **solución óptima** si proporciona el valor más favorable a la función objetivo, es decir, $x^* \in \mathcal{R}$ es óptima si $\forall x \in \mathcal{R}$, $cx^* \geq cx$.*

La función objetivo puede representar un problema de maximización o minimización. En un problema concreto, las restricciones pueden venir dadas en términos de igualdad o desigualdad ($\geq$ o $\leq$) y las variables pueden ser no negativas ($x \geq 0$), no restringidas ($x \in \mathbb{R}$) o acotadas ($l \leq x \leq u$). Sin embargo, dichas variantes pueden transformarse en el problema dado en forma estándar en (1.1) añadiendo variables adicionales (Merino, 2015).

$$\begin{aligned} \text{máx} \quad & z = cx \\ \text{s.a.} \quad & Ax \leq b \\ & x \geq 0 \end{aligned} \tag{1.3}$$

y su versión equivalente en notación extendida:

$$\begin{aligned} \text{máx} \quad & z = c_1x_1 + c_2x_2 + ... + c_nx_n \\ \text{sujeto a } & a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n \leq b_1 \\ & a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n \leq b_2 \\ & \vdots \qquad\qquad\qquad\quad \leq \vdots \\ & a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n \leq b_m \\ & x_1, x_2, \ldots, x_n \geq 0 \end{aligned} \tag{1.4}$$

Se puede transformar la función objetivo de un problema de programación lineal de maximización a minimización, se puede encontrar una teoría más detallada en Merino (2015).

- **Programación no lineal:** En matemáticas, programación no lineal (PNL) es el proceso de resolución de un sistema de igualdades y desigualdades sujetas a un conjunto de restricciones sobre un conjunto de variables reales desconocidas, con una función objetivo a maximizar (o minimizar), siendo alguna de las restricciones o la función objetivo no son lineales.

- **Problemas enteros:**, si todas sus variables de decisión son enteras, es decir, toman valores del conjunto de los números enteros $\mathbb{Z}$. También se conocen como problemas enteros puros lineales, y se denotan por PE. En particular, se tiene un problema entero $0-1$, si todas sus variables son binarias, es decir, toman valores en el conjunto $\{0,1\}$.

- **Problemas mixtos:** si tienen variables de decisión tanto continuas como enteras. También se conocen como problemas enteros mixtos lineales, y se denotarán por PM. En particular, un problema mixto es aquel que contiene tanto variables continuas como variables binarias.

- **Optimización multiobjetivo:** problemas que requieren la optimización simultánea de más de un objetivo. Habrá que optimizar por tanto una función de la forma $f : S \to T$, donde $S \subset \mathbb{R}^n$ y $T \subset \mathbb{R}^k$. Pero el problema está en que normalmente no existe un elemento de S que produzca un óptimo de forma simultánea para cada uno de los k objetivos que componen f. Esto se deberá a la existencia de conflictos entre objetivos, que harán que la mejora de uno de ellos dé lugar a un empeoramiento de algún otro.

  La mayor parte de los problemas de optimización del mundo real son naturalmente multiobjetivo. Esto es, suelen tener dos o más funciones objetivo que deben satisfacerse simultáneamente y que posiblemente están en conflicto entre sí. Sin embargo, a fin de simplificar su solución, muchos de estos problemas tienden a modelarse como mono objetivo usando sólo una de las funciones originales y manejando las adicionales como restricciones.

  La solución a estos problemas implica encontrar un conjunto de soluciones (en lugar de solo una) que represente las mejores compensaciones posibles entre los objetivos, de modo que ningún objetivo pueda mejorarse sin empeorar a otro.

  El problema de optimización multiobjetivo general puede formularse como:

Encontrar el vector $\vec{x}^* = [x_1^*, x_2^*, \ldots, x_n^*]^T$ que satisfaga las m restricciones de desigualdad:

$$g_i(\vec{x}) \geq 0 \quad i = 1, 2, \ldots, m \tag{1.5}$$

las p restricciones de igualdad

$$h_i(\vec{x}) = 0 \quad i = 1, 2, \ldots, p \tag{1.6}$$

y que optimice

$$\vec{f}(\vec{x}) = [f_1(\vec{x}), f_2(\vec{x}), \ldots, f_k(\vec{x})]^T \tag{1.7}$$

Para tratar el problema del conflicto entre objetivos se han utilizado diversos métodos:

- Métodos basados en el concepto de eficiencia de Pareto.
- Métodos basados en la combinación de objetivos.
  - método de la suma ponderada.
  - método de la programación por metas.

Todos los métodos anteriores han sido utilizados por distintos autores en combinación con los algoritmos evolutivos, que se han mostrado como una herramienta muy adecuada para resolver este tipo de problemas, por ejemplo (Mandal et al., 2010).

Óptimo de Pareto

Decimos que un punto $\vec{x}^* \in \Omega$ es un óptimo de Pareto si para toda $\vec{x} \in \Omega$ e $I = \{1, 2, \ldots, k\}$ ya sea,

$$\forall_{i \in I} \left(f_i(\vec{x}) = f_i\left(\vec{x}^*\right) \right)$$

o hay al menos una $i \in I$ tal que

$$f_i(\vec{x}) > f_i\left(\vec{x}^*\right)$$

Método de la suma ponderada

Método de la suma ponderada, en el que se optimizará el valor obtenido mediante la suma de los valores correspondientes a los distintos objetivos, multiplicados cada uno por un coeficiente de peso. Estos coeficientes de peso establecerán la importancia relativa de cada objetivo. El problema de optimización multiobjetivo se transforma así en otro de optimización escalar, que para el caso de la minimización será de la forma

$$\text{mín} \sum_{i=1}^{k} w_i f_i(x)$$

donde $w_i \geq 0$ es el coeficiente de peso correspondiente al objetivo i.

Método de la programación por metas

En el que se establece una meta para cada objetivo y lo que se suma ahora (multiplicado por el correspondiente coeficiente) es la distancia de cada objetivo a su meta. Para un caso de minimización sería

$$\text{mín} \sum_{i=1}^{k} w_i \left| f_i(x) - M_i \right|$$

donde M_i representa la meta del i-ésimo objetivo.

- **Optimización Combinatoria:** La Optimización Combinatoria es una rama de la optimización. Su dominio se compone de problemas de optimización donde el conjunto de posibles soluciones es discreto o se puede reducir a un conjunto discreto.

  A medida que la complejidad del espacio de búsqueda aumenta, el coste de ejecución de dichos algoritmos puede aumentar de forma exponencial, convirtiendo la resolución en prácticamente inviable.

  Otra posibilidad para afrontar este tipo de problemas, consiste en buscar una solución subóptima, pero en un tiempo razonable.

1.4. Ruteo de vehículos

El idear un modelo de ruteo para un conjunto de vehículo puede representar entre un 10 % y 20 % del costo final de los productos como se menciona en (Olivera, 2004). Toma mayor importancia si idear un modelo de ruteo de vehículos se traduce en minimizar el número de deceso de vidas humanas frente a un desastre.

Se ha visto un incremento de investigaciones sobre problemas de ruteo de vehículos aplicados a desastres naturales, Guzman et al. (2020) muestra una revisión de problemas de ruteo de vehículos aplicados a Desastres naturales.

1.4.1. Problema del agente viajero (TSP)

El problema del agente viajero (TSP) se basa en encontrar la ruta de un vendedor que comienza desde una ubicación de origen, visita un conjunto de ciudades prescritas y regresa a la ubicación original de tal manera que la distancia total recorrida sea mínima y cada ciudad sea visitada exactamente una vez. Este problema ha captado la atención a lo largo de los años debido en parte a su amplia aplicabilidad en la práctica (Gutin and Punnen, 2006). Una de las características más interesantes de este problema es que existen muchas formulaciones conocidas para el mismo problema. Una de ellas es la siguiente.

Sea el conjunto de ciudades $0, 1, 2, \ldots, n$

$$x_{ij} = \begin{cases} 1, & \text{si va de la ciudad } i \text{ a la ciudad } j \\ 0, & \text{en otro caso} \end{cases}$$

Sean u_i para $i = 1, ..., n$ variables artificiales y c_{ij} la distancia desde la ciudad i a la ciudad j. Entonces el modelo de programación lineal en enteros puede ser escrito como:

$$\text{mín} \sum_{i=0}^{n} \sum_{j \neq i, j=0}^{n} c_{ij} x_{ij} \tag{1.8}$$

sujeto a

$$0 \leq x_{ij} \leq 1 \qquad i, j = 0, \ldots n \tag{1.9}$$

$$x_{ij} \text{ entero} \qquad i, j = 0, \ldots n \tag{1.10}$$

$$\sum_{i=0, i \neq j}^{n} x_{ij} = 1 \qquad j = 0, \ldots, n \tag{1.11}$$

$$\sum_{j=0,j\neq i}^{n} x_{ij} = 1 \qquad i = 0, \ldots, n \tag{1.12}$$

$$u_i - u_j + nx_{ij} \leq n - 1 \qquad 1 \leq i \neq j \leq n \tag{1.13}$$

La restricción (1.11) asegura que cada ciudad $0, ..., n$ de salida llegue exactamente a una ciudad, y (1.12) aseguran que desde cada ciudad $1, ..., n$ se salga exactamente hacia una ciudad (ambas restricciones también implican que exista exactamente una salida desde la ciudad 0), (1.11) y (1.11) aseguran los valores binarios para $x_{i,j}$, (1.12) obliga a que un solo camino cubra todas las ciudades y no dos o más caminos disjuntos cubran conjuntamente todas las ciudades.

1.4.2. Problema de ruteo de vehículo (VRP)

Estos (VRP) generalmente se modelan a través de programas enteros mixtos basados en un grafo matemático. Estos grafos consisten en un conjunto de vértices y un conjunto de arcos de conexión. Los vértices representan clientes o depósitos. Los arcos representan los caminos entre clientes y depósitos. Los arcos generalmente contienen costos, estos costos representan los tiempos de viaje o las distancias entre los clientes. En el VRP clásico, el tiempo de conducción, tiempo de servicio y los clientes se conocen de antemano. Para los VRP, una solución generalmente se define mediante la asignación de clientes a los vehículos y la secuencia de los clientes asignados para cada vehículo (Ulmer, 2017). El VRP clásico se puede definir de la siguiente manera:

Sea $G = (V, A)$ un grafo donde $V = \{1, \ldots, n\}$ es un conjunto de vértices que representan ciudades con el depósito ubicado en el vértice 1, y A es el conjunto de arcos. Con cada arco (i, j) $i \neq j$ se asocia un matriz no negativa de distancia $C = (c_{ij})$. En algunos contextos, c_{ij} puede interpretarse como un costo de viaje o como un tiempo de viaje. Cuando C es simétrico, a menudo es conveniente reemplazar A por un conjunto E de bordes no dirigidos.

La formulación del VRP se puede presentar de la siguiente manera: Sea E el conjunto de arcos, x_{ij} una variable entera que puede tomar valores $\{0, 1\}, \forall\{i, j\} \in E \setminus \{\{0, j\} : j \in V\}$, 1 si el vehículo va del vértice i al vértice j y cero en otro caso, valores $\{0, 1, 2\}, \forall\{0, j\} \in E, j \in V$. Tenga en cuenta que $x_{0j} = 2$ cuando se selecciona una ruta que incluye el único cliente j en la solución.

El VRP puede formularse como la siguiente programación entera:

$$\text{mín} \sum_{i \neq j} c_{ij} x_{ij} \tag{1.14}$$

sujeto a

$$\sum_{j \in V} x_{ij} = 1, \quad \forall i \in V \tag{1.15}$$

$$\sum_{i \in V} x_{ij} = 1, \quad \forall j \in V \tag{1.16}$$

$$\sum_{i \in V} x_{ij} \geq |S| - V(S), \{S : S \subseteq V \setminus \{1\}, |S| \geq 2\} \tag{1.17}$$

$$x_{ij} \in \{0, 1\}, \quad \forall \{i, j\} \in E; i \neq j \tag{1.18}$$

La función objetivo busca minimizar el costo total como se muestra en (1.14), generalmente representado por la distancia total recorrida. Las restricciones (1.15) y (1.16) limitan a solo una entrada y salida de cada nodo garantizando que todas sean visitadas. La restricción (1.17) son restricciones de eliminación de subtour: $v(S)$ es un límite inferior apropiado en la cantidad de vehículos necesarios para visitar todos los vértices de S en la solución óptima. Finalmente, la restricción (1.18) define al modelo como uno entero binario que vale 1 cuando por el arco (i, j) circula un vehículo, o cero en caso contrario Zirour (2008).

1.4.3. Problema de ruteo de vehículos con ventanas de tiempo (VRPTW)

El (VRPTW) es una generalización del conocido VRP. Se puede revisar como un problema combinado de ruteo y programación de vehículos que a menudo surge en muchas aplicaciones del mundo real. Se trata de optimizar el uso de una flota de vehículos que deben realizar varias paradas para atender a un conjunto de clientes, y especificar qué clientes deben ser atendidos por cada vehículo y en qué orden para minimizar el costo, sujeto a la capacidad del vehículo y restricciones de tiempo de servicio. El problema implica la asignación de vehículos a viajes, de modo que el costo de asignación y el costo de ruta correspondiente son mínimos (Zirour, 2008).

- $G = (C, A)$ es un grafo completo no dirigido.
- $C = \{0, 1, \ldots, n\}$ conjunto de nodos (clientes), donde 0 es el depósito.

- $A = \{(i, j) : i, j \in C, i \neq j\}$; conjunto de aristas que unen a los clientes i y j.
- c_{ij} es el costo de moverse desde el cliente i hacia el cliente j.
- $d_i \in \mathbb{Z}^+$ demanda del cliente i.
- k es número de vehículos pertenecientes a la flota.
- $Q \in \mathbb{Z}^+$ es la capacidad de los vehículos.
- Y_{iv} es la cantidad entregada por el vehículo v al cliente i.
- b_i^v es la hora de llegada del vehículo v al cliente i.
- $x_{ij}^v = \begin{cases} 1, & \text{si el vehículo } v \text{ viaja del cliente } i \text{ al cliente } j \\ 0, & \text{en otro caso} \end{cases}$

En esta variante del problema, como se detalla en la sección anterior, además de capacidades sobre los vehículos, cada cliente $i \in C \backslash 0$ tiene asociada una ventana de tiempo $[e_i, l_i]$ que establece un horario de servicio permitido para que un vehículo arribe a él y un tiempo de servicio o demora s_i (tiempo de descarga o entrega de mercancía).

Por ejemplo, si el camino entre los clientes (i, j) es parte de una solución, b_i y b_j son las horas de arribo a los clientes i y j respectivamente, entonces las ventanas de tiempo implican que debe cumplirse $b_i \leq l_i$ y $b_j \leq l_j$. Además, se consideran las constantes t_{ij}, que tienen como valor, la distancia existente entre los clientes i y j.

La formulación del (VRPTW) se puede presentar de la siguiente manera (Montes Orozco et al., 2017):

$$\text{mín} \sum_{i=0}^{n} \sum_{j=0}^{n} \sum_{v=1}^{k} c_{ij} x_{ij}^v \tag{1.19}$$

sujeto a

$$\sum_{i=0}^{n} \sum_{v=1}^{k} x_{ij}^v = 1; \quad \forall j = 1, 2, \ldots, n \tag{1.20}$$

$$\sum_{j=1}^{n} x_{0j}^v = 1; \quad \forall v = 1, 2, \ldots, k \tag{1.21}$$

$$\sum_{i=1}^{n} x_{i0}^v = 1; \quad \forall v = 1, 2, \ldots, k \tag{1.22}$$

$$\sum_{i=0}^{n} x_{ip}^{v} - \sum_{j=1}^{n} x_{pj}^{v} = 0; \quad \forall p = 1, 2, \ldots, n; \quad v = 1, 2, \ldots, k \tag{1.23}$$

$$Y_{iv} = d_i \sum_{j=0}^{n} x_{ji}^{v}; \quad \forall i = 1, 2, \ldots, n; \quad v = 1, 2, \ldots, k \tag{1.24}$$

$$\sum_{i=0}^{n} Y_{iv} \leq Q; \quad \forall v = 1, 2, \ldots, k \tag{1.25}$$

$$e_i \leq b_i^v \leq l_i, \quad \forall i, j = 1, 2, \ldots, n; \quad v = 1, 2, \ldots, k \tag{1.26}$$

$$x_{ij}^{v}(b_i^v + s_i + t_{ij} - b_j^v) \leq 0; \quad \forall i = 1, 2, \ldots, n; \quad v = 1, 2, \ldots, k \tag{1.27}$$

$$x_{ij}^{v} \in \{0, 1\}, \text{ y } Y_{iv} \geq 0; \quad \forall i = 1, 2, \ldots, n; \quad v = 1, 2, \ldots, k \tag{1.28}$$

El conjunto de restricciones (1.20) asegura que cada cliente debe ser visitado sólo por un vehículo. Las restricciones (1.21) aseguran que cada vehículo parta solo una vez del depósito, mientras que el conjunto de restricciones (1.22) asegura que cada vehículo regrese al depósito. Las restricciones (1.23) hacen que cada vehículo al arribar a la ubicación de un cliente, salga de esta (restricción de transbordo). El conjunto de restricciones (1.24) impone que la cantidad entregada al cliente i por el vehículo v sea satisfecha en su totalidad. La restricción (1.25) impone que la cantidad entregada en cada ruta no exceda la capacidad del vehículo. La restricción (1.26) asegura que los límites de las ventanas de tiempo son respetados. La restricción (1.27) asegura que cada vehículo v no pueda iniciar el servicio a cierto cliente j, si la suma del tiempo de transporte entre los clientes i y j, la duración del servicio para el cliente i y el tiempo de llegada al cliente i, es mayor que la ventana de tiempo para el cliente j. La restricción (1.28) define el tipo de las variables utilizadas. Finalmente, la función objetivo (1.19) busca minimizar el costo total de las rutas asignadas al conjunto de vehículos.

La siguiente tabla muestra la notación y descripción de varios modelos de ruteo de vehículos existentes.

Modelo de ruteo de vehículo	Descripción
VRP	El problema de ruteo de vehículos VRP, por sus siglas en inglés, hace referencia a la asignación adecuada de las rutas de distribución o abastecimiento, a una flota de carros destinada a transportar la mercancía de un punto de depósito a los clientes.
CVRP	(Capacited VRP), es el VRP más general y consiste en uno o varios vehículos con capacidad limitada y constante encargados de distribuir los productos según la demanda de los clientes.
MDVRP	(Multi-Depot VRP), o VRP con múltiples depósitos es un caso de ruteo de vehículos en el que existen varios depósitos (cada uno con una flota de vehículos independiente) que deben servir a todos los clientes.
PVRP	(Period VRP), contempla en su planteamiento un horizonte de operación de M días, periodo durante el cual cada cliente debe ser visitado una vez.
SDVRP	(Split Delivery VRP), o VRP de entrega dividida, donde se permite que un cliente pueda ser atendido por varios vehículos si el costo total se reduce, lo cual es importante si el tamaño de los pedidos excede la capacidad de un vehículo.
SVRP	(Stochastic VRP), se trata de un VRP en que uno o varios componentes son aleatorios; clientes, demandas y tiempos estocásticos son las principales inclusiones en este tipo de problemas.
VRPPD	(VRP Pickup and Delivery), o VRP con entrega y recogida, es aquel en el que cabe la posibilidad de que los clientes pueden devolver determinados bienes, por tanto, se debe tener presente que estos quepan en el vehículo. Esta restricción hace más difícil el problema de planificación y puede causar una mala utilización de las capacidades de los vehículos, un aumento de las distancias recorridas o a un mayor número de vehículos.
MFVRP	(Mix Fleet VRP), es un VRP en el que se suponen vehículos con distintas capacidades o capacidad heterogénea, por lo que es necesario considerar estas capacidades en la ruta que seguirá cada recurso, ya que un camión más grande podrá realizar una ruta más larga o que tenga mayor concentración de demanda.

VRPTW	(VRP with Time Windows), es aquel en el que se incluye una restricción adicional en la que se asocia a cada cliente una ventana de tiempo, es decir, cada cliente sólo está dispuesto a recibir el bien o servicio durante un intervalo de tiempo predeterminado.
VRPB	VRP con red de retorno: El cliente pude demandar o entregar la mercancía, de esta manera se generan dos subconjuntos de clientes al interior de las rutas, los consumidores y los vendedores; ocasionando una distribución mixta, lo cual representa la minimización de costos totales asociados.
DCVRP	VRP con restricciones de capacidad y distancia: La capacidad de los vehículos es limitada y la longitud de los arcos que se realizan en una ruta, es decir las distancias.
OVRP	VRP abierto: En este caso, el vehículo no está obligado a regresar al depósito, una vez haya finalizado su recorrido. Se presenta generalmente, cuando las organizaciones no cuentan con una flota de vehículos propia parcial o totalmente.
VRPF	VRP difuso: Surge en respuesta a la dificultad para establecer demandas, tiempos de recorrido y ubicación de los clientes desconocidos.
PVRPTW	El problema de ruteo periódico del vehículo con ventanas de tiempo: requiere atender a un conjunto de clientes en un horizonte de planificación de varios períodos y satisfacer las restricciones de ventana de tiempo para las visitas en cada período. Cada cliente se caracteriza por un conjunto de combinaciones viables de períodos de visita (llamados "patrones") y una ventana de tiempo para las visitas.
DVRPTW	Ruteo dinámico del vehículo con ventanas de tiempo: se enfoca en la demanda cambiante que se presenta al interior de un sistema de transporte y en donde los requerimientos, deben satisfacerse dentro de una ventana de tiempo definida. Es decir, el cliente solo se encuentra disponible para atender el requerimiento en un intervalo de tiempo definido; de lo contrario se verá penalizado económicamente.

LRP	Problema de localización y ruteo: Puede ser representado por una red, donde cada cliente y depósito son simbolizados por un vértice y deben ser visitados una vez para la entrega de un bien o recolección de un objeto; y, los arcos son las rutas de acceso. Para este problema no se toma en cuenta el inventario de cada instalación o del depósito.
CTP	El problema del recorrido de cobertura: El CTP consiste en determinar un ciclo hamiltoniano de longitud mínima, de modo que cada vértice esté dentro de una distancia preespecificada del ciclo.

Tabla 1.1: Taxonomía de problemas de ruteo de vehículos

CAPÍTULO II

OBTENCIÓN DE CASOS DE ESTUDIOS.

En este capítulo se presenta una recopilación de casos de estudios más importantes relacionados con metaheurísticas para el proceso de entrega óptima de productos mediante ruteo de vehículos aplicados en zonas de emergencia.

2.1. Obtención de casos de estudio

2.1.1. Obtención de casos de estudios local.

No se han encontrado estudios en el ámbito local sobre la distribución de productos mediante ruteo de vehículos aplicados en zonas de emergencia.

2.1.2. Obtención de casos de estudios Nacional.

En el ámbito nacional se han encontrado algunos trabajos de tesis.

1. Benavente Sotelo (2016) ha realizado un trabajo de investigación para hallar una solución factible para un modelo de ruteo de vehículos frente a un terremoto de magnitud 8,0 Mw con epicentro frente a Lima y hace énfasis en la distribución de ayuda humanitaria no alimentaria. Propone un modelo con fundamento científico que busca optimizar las rutas de distribución en menos de 72 horas, plazo máximo definido por el Instituto Nacional de Defensa Civil (INDECI).

   Trabaja bajo los supuestos de contar con 22 almacenes los cuales abastecerán a 42 subestaciones. Cada subestación distribuirá los bienes a 50 nodos ubicados en parques aledaños. La población damnificada deberá acercarse al parque seleccionado para recibir tres kits de bienes de ayuda.

Propone un modelo VRPTW, haciendo la aclaración que este modelo presenta limitaciones del software ya que el modelo es calificado como NP-hard debido a su complejidad computacional. Para superar este inconveniente emplea la heurística método en dos fases: asignando primero los nodos en grupos, usando el algoritmo de barrido, y luego determinar la ruta con ayuda de un modelo TSP. Luego de realizar la heurística a cada una de las 42 subestaciones, concluye que sí se llega a cumplir con la norma de 72 horas realizando un recorrido total de 70 800 kilómetros.

2. García Avellaneda (2019) desarrolla un diseño y la simulación de un modelo de un plan de evacuación en caso de emergencia por tsunami en el distrito La Punta (Perú).

   Señala que, en el Perú, se encuentra uno de los litorales más vulnerables de Sudamérica, expuesta a inundaciones por causa de tsunami, pudiendo ser la causa de un gran número de pérdidas de vidas humanas en el país. Contempla como uno de estos escenarios, un terremoto de magnitud 8.5 Mw con epicentro en el mar, frente al Callao, coincidiendo con el terremoto más probable que podría ocurrir en Lima, donde se sabe que las primeras olas llegarían a las costas de La Punta en 15 minutos.

   El objetivo principal de su trabajo es, optimizar el número de recursos y número de personas evacuadas a una zona segura fuera del distrito La Punta. Propone el cumplimiento del objetivo a través de optimizar las rutas de escape a utilizar por los vehículos de la zona. Confirma la viabilidad de estas rutas a través de un modelo de simulación por dinámica de transportes de camino dirigido.

   Concluye que dos variables juegan un importante rol en la cantidad de vehículos evacuados en el límite de tiempo que llegan las olas a las costas de La Punta, el tráfico inicial y el tiempo de reacción de las personas para poder salir del distrito.

3. Pareja Villegas and Rodriguez Leiva (2016) Diseñan un modelo multivariado para identificar los factores determinantes que explican el número de damnificados por causa de un terremoto en la región de Lima y Callao; asimismo, formula un modelo de programación lineal entera para ruteo de vehículos con ventanas de tiempo para determinar el plan de distribución de los bienes de ayuda humanitaria, tomando en cuenta las restricciones de la situación. Con dicho modelo, determina la cantidad de vehículos necesarios y las rutas de despacho para atender a los afectados luego de ocurrido un terremoto. Discute y evalúa las propuestas de mejora para la distribución de ayuda humanitaria en la región del Callao desde el almacén nacional del INDECI, a partir de los resultados del modelo de ruteo de vehículos. Por último, contrasta los resultados de los escenarios

analizados en términos de costos y medidas de respuesta en el ámbito de la logística humanitaria.

4. (Alva Cabrera, 2014) Desarrolla un trabajo aplicativo para problemas de ruteo de vehículos estrechamente relacionado a la problemática surgida en la logística humanitaria. Este problema radica en cómo realizar la entrega de la ayuda desde los centros de abastecimiento hasta los puntos de acopio en los distritos de Lima Metropolitana y el Callao, al ser afectados por un terremoto de elevada intensidad, teniendo en cuenta además la necesidad de realizar dicha entrega de una manera ágil y eficiente y recorrer las menores distancias obteniendo de esta forma un mejor tiempo de respuesta satisfaciendo toda la demanda, representada en este caso por los damnificados a asistir.

   Para lograr las mejores características planteadas en la entrega, primero realizó el mapeo de dichos puntos de acopio tomando en cuenta los almacenes de insumos médicos para luego resolver el problema con la aplicación de diversos métodos heurísticos como el de Cercanía de Puntos, Sweep o Barrido, Gran Ruta aplicando posteriormente el Método del Ahorro y Programación Lineal y el de Clasificación Ascendente Jerárquica. En segundo lugar, una vez concluidas todas las propuestas toma una decisión acerca de cuál sería el mejor modelo a seguir para la resolución de este VRP (Vehicle Routing Problem) basado en las distancias recorridas por el transporte hacia cada uno de los puntos que conforman las clusters establecidas. También consideró la cantidad de damnificados a asistir y determinó la cantidad de recursos empleados por cada viaje de la flota vehicular terrestre.

2.1.3. Obtención de casos de estudios Internacional.

En el ámbito internacional se han encontrado algunos trabajos de investigación.

1. (Jeanmonod et al., 2018) presenta el problema de ruteo dinámico de vehículos para la logística de socorro en desastres naturales como terremotos o tifones. Considera que la entrega / recogida coordinada y ordenada de los recursos disponibles ayuda a mitigar los daños a la propiedad y salvar vidas.

   Para solucionar este problema propone un método (DVRPRL) problema del ruteo dinámico del vehículo para la logística de socorro en desastres naturales donde las demandas de bienes de socorro, incluida la entrega o la recogida, pueden variar en tiempo real y probablemente sean impredecibles. En cada sitio de demanda, se requieren varios tipos

de artículos de socorro con diferentes niveles de emergencia y plazos, y es posible que se necesiten operaciones de entrega y recogida simultáneamente. Reutiliza exceso de bienes de socorro o equipo de rescate recogido en un sitio en cualquiera de los siguientes sitios del mismo recorrido. Para una respuesta rápida a las demandas futuras, un vehículo despachado puede no tener que regresar al depósito a menos que sea necesario rellenarlo con artículos de ayuda, este puede esperar en su última parada hasta que reciban el próximo pedido del centro de coordinación de logística.

2. (Allen, 2017) elabora un trabajo de investigación, producto del terremoto en Nepal de abril del 2015, modela la entrega de suministros como un problema de enrutamiento de vehículos utilizando el método de dos etapas de Fisher y Jaikumar, que asigna ubicaciones a los vehículos a través de un programa entero y luego utiliza la heurística para el ruteo de los vehículos. En la etapa de asignación, utiliza la formulación del problema de asignación para asignar ubicaciones a los vehículos. En la etapa de ruteo, implementa sub-tour y la heurística de recocido simulado por el bien de la comparación.

3. (Al Theeb and Murray, 2017) expresa que después de un desastre, los suministros deben distribuirse de manera eficiente y equitativa a las personas necesitadas, los heridas deben ser evacuados a centros de triaje, y los socorristas deben ser transportados a las áreas afectadas.

   Para abordar este problema, presenta un modelo detallado de programación matemática. Resalta que, debido a la complejidad de la formulación, solo las instancias de problemas a pequeña escala se pueden resolver de manera óptima a través de un software de solución comercial. Propone un nuevo enfoque heurístico para resolver problemas de tamaño práctico dentro de un tiempo aceptable.

4. (Korkou et al., 2016) Presenta un modelo de varios períodos con el objetivo de minimizar la escasez de diferentes productos de socorro en varias áreas afectadas. Genera un conjunto de pruebas de problemas de referencia con diversas características a partir del modelo propuesto y resuelve de manera óptima con CPLEX. Utiliza el conjunto de pruebas para comparar una serie de metaheurísticas. Las modificaciones necesarias las introducen en los algoritmos, para ajustarse a los requisitos especiales del tipo de problema específico. El rendimiento de los algoritmos los evalúa en términos de precisión de la solución con respecto a las soluciones óptimas. Las comparaciones entre las metaheurísticas empleadas ofrecen información valiosa sobre su capacidad para abordar los problemas de logística humanitaria.

5. (Mguis et al., 2014) emplea un algoritmo genético para la planificación de la ayuda humanitaria en casos de desastres naturales. Clasifica su

estudio como un problema de ruteo dinámico del vehículo con ventanas de tiempo (DVRPTW), donde los clientes deben ser atendidos durante un intervalo de tiempo específico. Expresa que en caso de un desastre, la planificación de emergencias debe ser rápida y consistente. Por esas razones, opta por un algoritmo genético mejorado al agregar una variedad de guías para acelerar la convergencia del algoritmo. Por lo tanto, el algoritmo genético proporciona una población de soluciones que pueden abordar el aspecto dinámico del problema. Su objetivo es proporcionar un plan para satisfacer todas las demandas minimizando la distancia total recorrida. Prueba el enfoque propuesto con datos teóricos y encuentra una alta eficiencia, lo que infiere la posibilidad de aplicar la gestión de llamadas de emergencia en caso de un desastre mayor.

6. (Nolz et al., 2010) estudia un problema de objetivos múltiples que surge en una situación posterior a un desastre natural. Supone que la infraestructura en la región afectada ha sido parcialmente destruida por un terremoto, inundación o tsunami. El problema refiere a la ayuda de supervivencia, lo que significa el suministro de alimentos, refugio y medicamentos a la población que vive en el área afectada, compensando las instalaciones destruidas. Desarrolla un método híbrido basado en algoritmos genéticos, búsqueda de vecindad variable y vinculación de ruta para resolver el problema motivado del mundo real. El algoritmo ha sido probado con datos reales de la provincia de Manabí en Ecuador. También presenta resultados para instancias del mundo real de mediano y gran tamaño.

7. (Penna et al., 2018) examina un problema de ruteo de vehículos (VRP) aplicado a la fase de respuesta después de un desastre natural. Propone un modelo VRP que involucra una flota heterogénea de vehículos, múltiples viajes, múltiples depósitos y dependencias del sitio del vehículo. El método es una heurística híbrida genérica que utiliza una formulación de particionamiento de conjunto para agregar memoria a un marco de búsqueda local iterada de inicio múltiple. La heurística calcula rápidamente rutas eficientes mientras determina la cantidad de vehículos requeridos y el subconjunto de depósitos que se utilizarán.

8. (Han et al., 2018) sugiere un procedimiento de solución que consta de dos partes. Uno es generar un plan de ruta de vehículos de emergencia para vehículos individuales y el otro es coordinar esos planes para la sede. El algoritmo para la calificación genética de un plan de enrutamiento se compone de una técnica de optimización que utiliza la programación de enteros y una metaheurística que utiliza el algoritmo genético. Y para coordinar, propone una agrupación heurística y de k-medias basada en prioridades. El rendimiento del algoritmo está empíricamente demostrado que es eficiente tanto en la calidad de la solución como en el tiempo de búsqueda en varios tamaños de problemas.

9. (Wang et al., 2014) construye un modelo de ruteo de ubicación abierta de número entero no lineal para el problema de distribución de socorro considerando el tiempo de viaje, el costo total y la confiabilidad con entrega dividida. Propone el algoritmo genético de clasificación no dominada y el algoritmo de evolución diferencial de clasificación no dominada para resolver el modelo propuesto.

10. (Yi and Kumar, 2007) presenta una metaheurística de optimización de colonias de hormigas (ACO) para resolver el problema logístico que surge en las actividades de socorro en casos de desastre. La planificación logística implica el envío de productos a los centros de distribución en las áreas afectadas y la evacuación de los heridos a los centros médicos. El rendimiento del algoritmo es probado en varias redes generadas aleatoriamente y los resultados indican que este algoritmo funciona bien en términos de calidad de la solución y tiempo de ejecución.

11. (Alinaghian et al., 2019) presenta un nuevo modelo matemático para la ubicación de centros de ayuda temporales y el enrutamiento dinámico de vehículos de rescate aéreos que distribuyen suministros básicos en operaciones de ayuda. La función objetivo del modelo propuesto minimiza el tiempo de llegada al último centro de ayuda temporal designado. El modelo busca ubicar los centros de ayuda temporales de manera que todas las áreas afectadas estén cubiertas por al menos un centro de ayuda temporal. Presenta un algoritmo metaheurístico híbrido mejorado basado en la búsqueda de dispersión combinada con la búsqueda de vecindad variable. Para evaluar el rendimiento del algoritmo, prueba y compara con un método exacto, búsqueda de dispersión básica y algoritmo genético. Los resultados muestran el buen desempeño del algoritmo.

12. (Li and Chung, 2019) aborda el problema de enrutamiento de vehículos capacitados (CVRP) y el problema de enrutamiento de vehículos de entrega dividida (SDVRP) con tiempos de viaje inciertos y demandas al planificar rutas de vehículos para entregar suministros críticos a la población afectada que lo necesita después de un desastre. Utiliza un enfoque de optimización robusto para CVRP y SDVRP considerando las cinco funciones objetivas: minimización del número total de vehículos desplegados, el tiempo total de viaje / costo de viaje, la suma de los tiempos de llegada, la suma de los tiempos de llegada ponderados por demanda y el último tiempo de llegada. Propone un nuevo método heurístico de dos etapas que combina el algoritmo de inserción extendido y la búsqueda tabú para resolver los modelos VRP para problemas a gran escala.

13. (Xiong et al., 2019) presenta un modelo de enrutamiento de ubicación de varios niveles (LPR) para situaciones posteriores a un terremoto.

Desarrolla un modelo multiobjetivo para el (LRP). El objetivo de este modelo es minimizar el tiempo total en la entrega de suministros de emergencia y minimizar el tiempo máximo de espera para que los suministros de emergencia lleguen a los puntos de demanda. Emplea un algoritmo heurístico híbrido para resolver el modelo.

2.2. Ruteo de vehículos aplicado a la distribución de productos en zonas de emergencia

En la mayoría de problemas de ruteo de vehículos (VRP) las demandas de los clientes debe procesarse a un costo mínimo. Sin embargo, hay aplicaciones de (VRP) donde el servicio no tiene beneficio monetario. Estas aplicaciones aparecen en el contexto de ayuda humanitaria, donde el principio clave es prevenir o aliviar el sufrimiento humano.

Definimos un desastre como un evento extraordinario que puede ocurrir con o sin previo aviso limitado y tiene efectos devastadores en la población. Los lugares declarados como zonas de emergencia suelen ser lugares afectados por desastres naturales como: terremotos, huracanes, tsunami, o desastres provocados por el hombre como: graves circunstancias políticas o civiles que afectan e impiden la vida normal de una comunidad.

La gravedad del impacto en las víctimas depende de la magnitud del evento y de dos factores adicionales: la vulnerabilidad de la población afectada, expresado como la incapacidad para resistir el peligro, y las capacidades disponibles para hacer frente a los efectos del evento. Como las capacidades locales se ven abrumadas después de un desastre, las agencias de ayuda nacionales e internacionales juegan un papel importante en proporcionar el apoyo que se requiere para aliviar el sufrimiento. Aunque las autoridades locales son responsables de las operaciones de socorro, las agencias de ayuda están muy involucradas en el proceso de gestión de desastres. Sus funciones incluyen el despliegue de equipos de rescate y el abastecimiento de suministros esenciales como agua, alimentos y medicamentos a las regiones afectadas.

La investigación en el campo del ruteo de vehículos en zonas de emergencia para la entrega de productos puede hacer contribuciones importantes a apoyar a las agencias de ayuda en sus actividades operativas y eventualmente ayudar a salvar vidas (Toth and Vigo, 2002).

2.2.1. Logística en la gestión de zonas de emergencia

Una de las herramientas más importantes para aprovechar al máximo los recursos existentes y permitir un alivio rápido, incluso en circunstancias agravantes, es la logística. Una vez que se han evaluado los daños y se han determinado los requisitos, se envían suministros de todo el mundo a la zona del desastre. En esta etapa, se necesitan estrategias logísticas efectivas para permitir un traslado fluido de productos desde el punto de origen a las víctimas. De hecho, cada desastre tiene diferentes efectos, sin embargo, la forma básica en que se establece una red es similar en todas las operaciones de socorro. Por lo general, los suministros internacionales ingresan al área afectada a través de un gran puerto o aeropuerto. A continuación, estos suministros se transfieren a un almacén primario cercano. Los productos (suministros) luego fluyen a ubicaciones de almacenamiento final y finalmente se entregan a las víctimas. La primera etapa del proceso de transporte generalmente se lleva a cabo en camiones o trenes de larga distancia, vehículos o camionetas más pequeñas cubren los últimos kilómetros hasta las víctimas remotas (Toth and Vigo, 2002).

En esta tesis, nos enfocamos en el ruteo de vehículos que involucran asistencia directa a las víctimas (entrega de suministros esenciales).

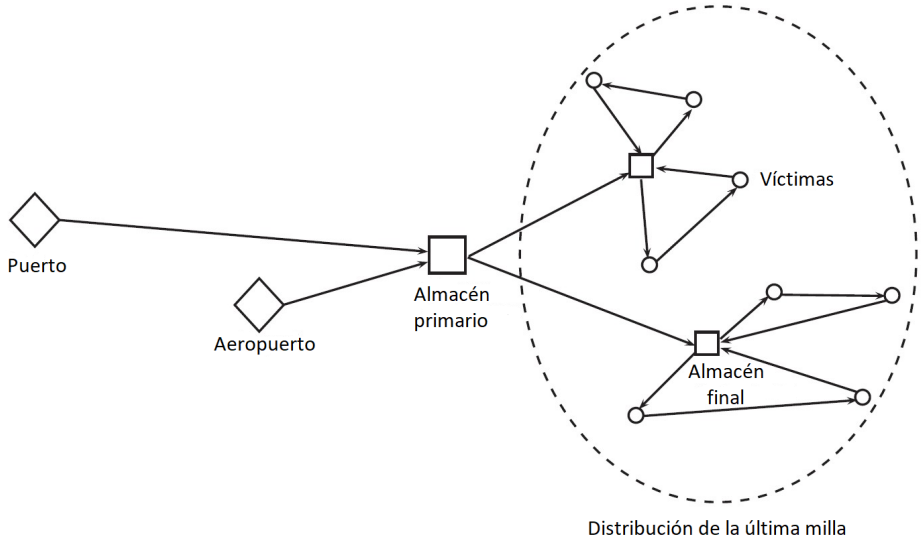


Figura 2.1: Cadena de suministro de socorro en casos de desastre

El objetivo humanitario de utilizar todos los medios posibles para aliviar el sufrimiento humano se desvía claramente del objetivo clásico de VRP para minimizar el costo. Además, el entorno peligroso en el que operan los vehículos después de un desastre plantea nuevos desafíos para modelar y resolver estos problemas.

La red de carreteras en la que viajan los vehículos puede estar en malas condiciones (por ejemplo, las carreteras con puentes dañados pueden no ser transitables, lo que hace que toda la planificación sea inútil). Puede ser difícil reunir medios de transporte suficientes para manejar el suministro entrante debido a la escasez de vehículos, la competencia entre las agencias de ayuda por vehículos o la negativa de los transportistas locales a operar en áreas expuestas al riesgo de saqueo. Incluso si la capacidad de la flota es suficiente a pesar de estas dificultades, la escasez de combustible podría obligar a algunos vehículos a permanecer inactivos. Además de las circunstancias extremas que deben capturarse en modelos significativos, las decisiones deben tomarse rápidamente.

Por lo general, los equipos internacionales de rescate pueden llegar al sitio del desastre en 24 horas. No pueden esperar un día o más para obtener los resultados de un algoritmo de optimización. En tal caso, la eficiencia computacional no es una cuestión de costo o conveniencia del usuario, sino más bien una cuestión de salvar vidas. Aunque está claro que el objetivo final es reducir el sufrimiento, de acuerdo con la literatura, definimos tres objetivos específicos que se originan directamente de la naturaleza de un desastre.

Primero, la situación inmediatamente después de un desastre requiere una respuesta rápida. Las personas pueden resultar heridas, enterradas bajo los escombros, traumatizadas y quedar sin refugio y comida. Cada segundo cuenta en esta etapa, y el tiempo se convierte en el recurso más importante para evitar una mayor pérdida de vidas. Por lo tanto, acelerar la respuesta es un aspecto importante en el socorro en casos de desastre. En segundo lugar, el número de personas afectadas es muy alto después de un desastre. Como resultado, puede haber suministros insuficientes y medios de transporte limitados. Si el transporte es el cuello de botella, los modelos de enrutamiento de vehículos pueden usarse para maximizar la cantidad entregada. El tercer objetivo es dedicar una atención imparcial y justa a todos. La importancia de este objetivo aumenta con la escasez de recursos. Finalmente, aunque no es un objetivo humanitario, generalmente debemos tener en cuenta los costos. De hecho, los desastres de inicio rápido a menudo están sobrefinanciados por donaciones. Sin embargo, en las fases posteriores de una operación de socorro en casos de desastre, es probable que el presupuesto se convierta en una preocupación importante.

2.2.2. Fases en la gestión de desastres

Las actividades de logística de desastres en las operaciones de ayuda humanitaria se realizan en cuatro fases. Estas fases son: mitigación, preparación, respuesta y recuperación (Toth and Vigo, 2002).

2.2.2.1. Fase de mitigación

La fase de mitigación se refiere a actividades que disminuyen el impacto potencial de un desastre. Las actividades típicas incluyen el refuerzo de edificios para soportar desastres o la prevención de poblar áreas peligrosas.

2.2.2.2. Fase de preparación

En la fase de preparación, se hacen arreglos que permiten una respuesta rápida y apropiada después de un desastre. Cuando ocurre un desastre, hay una enorme demanda de suministros esenciales en un corto período de tiempo. Es crucial tener suministros de emergencia como carpas, botiquines de primeros auxilios y alimentos a la mano para estar preparados para el peor de los casos. Cuanto mejor sea la preparación, más rápido será el tiempo de respuesta después de un desastre y más vidas se pueden salvar. Sin embargo, esta fase se caracteriza principalmente por la falta de información sobre la ubicación y el alcance de un desastre que dificulta la planificación previa.

2.2.2.3. Fase de respuesta

La fase de respuesta se centra en operaciones que satisfacen las necesidades urgentes de una población inmediatamente después de un desastre y puede continuar durante varias semanas. En esta fase, todos los recursos disponibles se movilizan para minimizar la amenaza inmediata para el bienestar de la población. Una operación humanitaria exitosa es aquella que satisface las necesidades urgentes de la población en el menor tiempo posible. El costo asociado con las operaciones juega un papel menor. En cambio, el tiempo es el factor más crítico.

2.2.2.4. Fase de recuperación

La fase de recuperación incluye actividades a largo plazo que apoyan la autosostenibilidad de la población y el retorno a la normalidad, incluida la eliminación de escombros y el suministro de atención médica y alimentos. Típicamente, las cuatro fases se llevan a cabo de manera cíclica (es decir, las comunidades afectadas comienzan con la fase de mitigación justo después de haberse recuperado de un desastre). Sin embargo, casi siempre es difícil determinar un punto específico en el tiempo para la transición de la fase de respuesta a la fase de recuperación.

2.2.3. Procesos de la gestión del riesgo de desastres del estado Peruano

El estado peruano también ha promulgado en su artículo 38 del decreto supremo que aprueba el reglamento de la ley $N°29664$, que crea el Sistema Nacional de Gestión de Riesgo de Desastres (**SINAGERD**) la *Estructura del Plan Nacional de Gestión del Riesgo de Desastres*, la sustenta bajo el siguiente proceso:

- **Estimación del Riesgo**: El proceso de estimación del Riesgo comprende las acciones y procedimientos que se realizan para generar el conocimiento de los peligros o amenazas, analizar la vulnerabilidad y establecer los niveles de riesgo que permitan la toma de decisiones en la Gestión del Riesgo de Desastres.
- **Prevención del Riesgo**: El proceso de Prevención del Riesgo comprende las acciones que se orientan a evitar la generación de nuevos riesgos en la sociedad en el contexto de la gestión del desarrollo sostenible.
- **Reducción del Riesgo**: El proceso de Reducción del Riesgo comprende las acciones que se realizan para reducir las vulnerabilidades y riesgos existentes en el contexto de la gestión del desarrollo sostenible.
- **Preparación**: La Preparación está constituida por el conjunto de acciones de planeamiento, de desarrollo de capacidades, organización de la sociedad, operación eficiente de las instituciones regionales y locales encargadas de la atención y socorro, establecimiento y operación de la red nacional de alerta temprana y de gestión de recursos, entre otros, para anticiparse y responder en forme eficiente y eficaz. en caso de desastre o situación de peligro inminente, a fin de procurar una óptima respuesta en todos los niveles de gobierno y de la sociedad.
- **Respuesta**: La Respuesta, como parte integrante de la Gestión del Riesgo de Desastres, está constituida por el conjunto de acciones y actividades, que se ejecutan ante una emergencia o desastre, inmediatamente de ocurrido este, así como ante la inminencia del mismo.
- **Rehabilitación**: El proceso de Rehabilitación es el conjunto de acciones conducentes al restablecimiento de los servicios públicos básicos indispensables e inicio de la reparación del daño físico, ambiental, social y económico en la zona afectada por una emergencia o desastre. Se constituye en el puente entre el proceso de respuesta y el proceso de reconstrucción.
- **Reconstrucción**: El Proceso de Reconstrucción comprende las acciones que se realizan para establecer condiciones sostenibles de desarrollo en

las áreas afectadas, reduciendo el riesgo anterior al desastre y asegurando la recuperación física y social, así como la reactivación económica de las comunidades afectadas.

2.2.4. Métricas de rendimiento en operaciones de desastres

Los aspectos financieros juegan un papel importante, pero la pregunta crucial es cómo el servicio alivia el sufrimiento humano. En consecuencia, los problemas de dinero se desvanecen en el fondo y se tratan como restricciones, en lugar de un objetivo.

Existen diferentes medidas de desempeño para cuantificar la eficiencia y efectividad de las operaciones humanitarias. Las medidas más adecuadas para los problemas de enrutamiento de socorro en casos de desastre son el tiempo de respuesta (el tiempo que transcurre hasta que se satisfacen las necesidades urgentes), la equidad del servicio (las diferencias en la asistencia brindada a las personas en varios lugares), la satisfacción de la demanda (el volumen de bienes distribuidos a la población) y el costo de transporte. La importancia de cada medida depende de la fase de gestión de desastres con la que se asocia el problema específico. Por ejemplo, el tiempo de respuesta es la principal preocupación en la fase de respuesta, mientras que el costo es más importante en la fase de recuperación (Toth and Vigo, 2002).

2.2.5. Tiempo de respuesta

El tiempo de respuesta es el intervalo desde la ocurrencia de una necesidad hasta que se satisface. Debido al alto grado de urgencia en las operaciones de socorro en casos de desastre, una respuesta rápida es crucial. Cualquier retraso en la asistencia podría provocar un mayor sufrimiento de la población afectada. En consecuencia, el tiempo es el recurso más valioso inmediatamente después de un desastre.

2.2.6. Satisfacción de la demanda

En las operaciones de socorro en casos de desastre, las agencias de ayuda se enfrentan a una gran demanda que debe gestionarse con escasos recursos. Necesitan encontrar rutas para cada vehículo disponible de manera que se maximice el suministro distribuido, es decir, se incremente el número de vidas salvadas.

2.2.7. Equidad de servicio

Cada persona tiene el mismo derecho a recibir asistencia, por lo que las operaciones deben planificarse para evitar el trato injusto de cualquier grupo. Además, la distribución injusta de la oferta puede causar tensiones entre las comunidades cercanas y aumentar el riesgo de saqueo. Por lo tanto, se deben considerar métricas de equidad apropiadas durante la planificación de operaciones para equilibrar los niveles de servicio entre diferentes ubicaciones.

2.2.8. Costos de transporte

Minimizar los costos no es el objetivo principal de las organizaciones de ayuda, a pesar de que el alivio de desastres puede requerir mucho dinero. Inmediatamente después de un desastre, se satisfacen las necesidades urgentes independientemente de los gastos. Sin embargo, una vez que la situación se ha estabilizado, las agencias de ayuda deben operar económicamente para poder brindar asistencia hasta que se recupere la autosostenibilidad de la población.

Los modelos de rutas comerciales y de socorro son muy similares en términos de costo como una métrica de rendimiento. Los modelos de enrutamiento de socorro consideran los costos de viaje con menor prioridad o en conjunto con las limitaciones humanitarias.

2.2.9. Solución de problemas de alivio de desastres

Los diferentes requisitos en el enrutamiento de vehículos comerciales y el enrutamiento de socorro en casos de desastre plantean la cuestión de en qué medida las soluciones correspondientes son diferentes entre sí. Examinar las diferencias en las métricas de rendimiento es importante ya que los modelos de enrutamiento, los algoritmos y los productos de software se han desarrollado principalmente para aplicaciones comerciales. Para justificar la adaptación de estas herramientas de enrutamiento, el servicio proporcionado a las víctimas del desastre mediante enfoques humanitarios debe ser significativamente mejor después de aplicar las herramientas al problema.

Las diferencias significativas en las soluciones óptimas entre problemas comerciales y de enrutamiento de socorro requieren atención especial en el diseño de algoritmos de solución. Cambiar la función objetivo de un algoritmo que funciona bien en problemas de enrutamiento comercial puede no ser una opción viable para resolver problemas de enrutamiento humanitario.

El tiempo de computación es una preocupación importante en el socorro en casos de desastre. Cuando cada segundo cuenta, los tomadores de decisiones no pueden esperar mucho para generar soluciones razonables. Además, el entorno en constante cambio requiere revisiones y modificaciones frecuentes de las soluciones generadas. Los investigadores necesitan desarrollar algoritmos de baja complejidad que brinden buenas soluciones rápidamente para una variedad de parámetros de entrada.

CAPÍTULO III

METAHEURÍSTICAS

En Inteligencia Artificial (IA) se emplea el calificativo heurístico, en un sentido muy genérico, para aplicarlo a todos aquellos aspectos que tienen que ver con el empleo de conocimiento en la realización dinámica de tareas. Se habla de heurística para referirse a una técnica, método o procedimiento inteligente de realizar una tarea que no es producto de un riguroso análisis formal, sino de conocimiento experto sobre la tarea. En especial, se usa el término heurístico para referirse a un procedimiento que trata de aportar soluciones a un problema con un buen rendimiento, en lo referente a la calidad de las soluciones y a los recursos empleados.

La idea más genérica del término heurístico está relacionada con la tarea de resolver inteligentemente problemas reales usando el conocimiento disponible. El término heurística proviene de una palabra griega con un significado relacionado con el concepto de encontrar y se vincula a la supuesta exclamación eureka de Arquímedes al descubrir su famoso principio[1]. Las técnicas **heurísticas** son algoritmos que encuentran soluciones de buena calidad para problemas combinatorios complejos explotando el conocimiento del dominio de aplicación. Los algoritmos heurísticos son fáciles de implementar y encuentran buenas soluciones con esfuerzos computacionales relativamente pequeños; sin embargo, renuncian (desde el punto de vista teórico) a encontrar la solución óptima global de un problema. En problemas de gran tamaño rara vez un algoritmo heurístico encuentra la solución óptima global.

El término metaheurística se obtiene de anteponer a heurística el sufijo meta que significa "más allá" o "a un nivel superior". Los conceptos actuales de lo que es una metaheurística están basados en las diferentes interpretaciones de lo que es una forma inteligente de resolver un problema. Las metaheurísticas son estrategias inteligentes para diseñar o mejorar procedimientos heurísticos,

---

[1]El principio de Arquímedes es el principio físico que afirma: ≪Un cuerpo total o parcialmente sumergido en un fluido en reposo experimenta un empuje vertical en sentido contrario igual al peso del fluido desalojado≫

generales con un alto rendimiento. Son algoritmos de propósito general, que no dependen del problema, y que ofrecen buenos resultados pero que normalmente no acaban ofreciendo la solución óptima sino soluciones subóptimas. Se acostumbran a utilizar para aquellos problemas en que no existe un algoritmo o heurística específicos que los resuelva, o bien cuando no es práctico implementar dichos métodos.

En este capítulo se presentan dos metaheurísticas que se utilizaran en esta tesis, para optimizar problemas de ruteo de vehículos; la metaheurística inspirada en colonias de hormigas y algoritmos genéticos.

3.1. Metaheurística inspirada en colonias de hormigas

En ciencias de la computación y en investigación operativa, el algoritmo inspirado en colonias de hormigas, algoritmo hormiga u optimización por colonia de hormigas (Ant Colony Optimization, ACO) es una técnica para solucionar problemas computacionales que pueden reducirse a buscar los mejores caminos o rutas en grafos.

3.1.1. Descripción de los algoritmos ACO

En nuestro mundo natural, las hormigas (inicialmente) vagan de manera aleatoria, al azar, y una vez encontrada la comida regresan a su colonia dejando un rastro de feromonas. Si otras hormigas encuentran dicho rastro, es probable que estas no sigan caminando aleatoriamente, puede que estas sigan el rastro de feromonas, regresando y reforzándolo si estas encuentran comida finalmente.

Sin embargo, al paso del tiempo el rastro de feromonas comienza a evaporarse, reduciéndose así su fuerza de atracción. Cuanto más tiempo le tome a una hormiga viajar por el camino y regresar de vuelta otra vez, más tiempo tienen las feromonas para evaporarse. Un camino corto, en comparación, es recorrido con más frecuencia, y por lo tanto la densidad de feromonas se hace más grande en caminos cortos que en los largos. La evaporación de feromonas también tiene la ventaja de evitar convergencias a óptimos locales. Si no hubiese evaporación en absoluto, los caminos elegidos por la primera hormiga tenderían a ser excesivamente atractivos para las siguientes hormigas. En este caso, el espacio de búsqueda de soluciones sería limitado.

El concepto de colonias de hormigas llevado al mundo de la algoritmia es usado tanto para encontrar caminos de coste mínimo como para resolver

otro tipo de problemas donde se usan grafos para su representación, en este trabajo de investigación minimizarán el tiempo de ruteo.

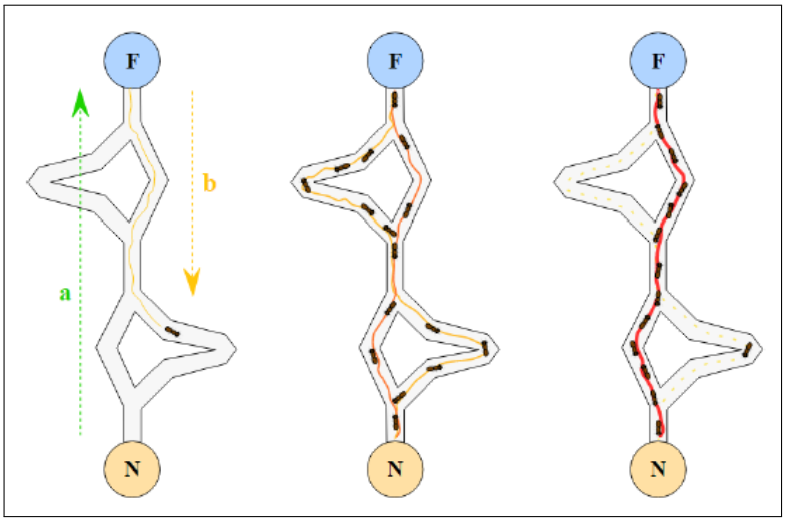


Figura 3.1: Comportamiento de las hormigas en búsqueda de su fuente de alimento.

Fuente:
`https://es.wikipedia.org/wiki/Algoritmo_de_la_colonia_de_hormigas`

La optimización de colonias de hormigas (ACO) aplicada a problemas de ruteo de vehículos involucra hormigas simuladas moviéndose a través del gráfico G visitando cada ciudad una vez y depositando feromona a medida que avanzan. El nivel de feromona depositada se define por la calidad del recorrido que encuentra la hormiga dada. Las hormigas deciden probabilísticamente qué ciudad visitar a continuación usando este nivel de feromona en los bordes del gráfico G y la información heurística basada en la distancia entre la ciudad actual de una hormiga y las ciudades no visitadas. Se utiliza un efecto de evaporación para evitar que los niveles de feromona alcancen un estado de optima local. Por lo tanto, ACO consta de dos etapas, la construcción del primer recorrido y la actualización de la feromona de la segunda etapa. La etapa de construcción del recorrido involucra a los actores que construyen recorridos completos. Las hormigas comienzan en una ciudad aleatoria y de forma iterativa toman decisiones probabilísticas sobre qué ciudad visitar a continuación utilizando la regla proporcional aleatoria por la cual la probabilidad de que la hormiga k en la ciudad i visite la ciudad $j \in N^k$ se define como:

$$p_{ij}^k = \frac{\tau_{ij}^\alpha \times \eta_{ij}^\beta}{\sum_{l \in N^k} \tau_{ij}^\alpha \times \eta_{ij}^\beta} \tag{3.1}$$

donde

- τ_{ij} es el nivel de feromona depositado en la arista que conduce de la ciudad i a la ciudad j.

- η_{ij} representa la visibilidad de la hormiga, es la información heurística que consiste en la distancia entre la ciudad i y la ciudad j establecida en $\dfrac{1}{dij}$.

- α y β son dos parámetros que se utilizan para ajustar la influencia relativa de feromonas y visibilidad.

- N_i^k es el conjunto de ciudades que aún no ha visitado la hormiga k cuando la hormiga se encuentra en el nodo i.

Una vez que todas las hormigas han completado la etapa de construcción del recorrido, se actualizan los niveles de feromona en los bordes del grafo G. Primero, se produce la evaporación de los niveles de feromona en cada borde del grafo G, por lo que el nivel se reduce en un valor relativo a la feromona en ese borde:

$$\tau_{ij} \leftarrow (1-\rho)\tau_{ij}, \rho \in (0,1] \tag{3.2}$$

donde p es la velocidad de evaporación. Una vez que se completa esta evaporación, cada hormiga k depositará feromona en los bordes que ha atravesado en función de la calidad del recorrido que encontró:

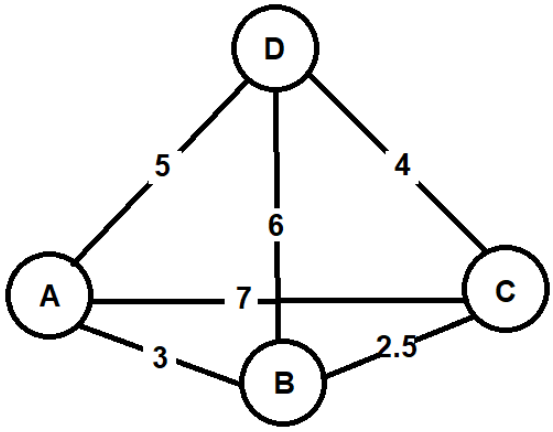
$$\tau_{ij} \leftarrow \tau_{ij} + \sum_{k=1}^{m} \Delta\tau_{ij}^k \tag{3.3}$$

donde la cantidad de feromona depositada por hormiga k, $\Delta\tau_{ij}^k$ se define por:

$$\Delta\tau_{ij}^k = \begin{cases} \dfrac{1}{C^k}, & \text{si el borde } (i,j) \text{ pertenece a } T^k \\ 0, & \text{en otro caso} \end{cases}$$

donde $1/C^k$ es la duración del viaje de la k-ésima hormiga en T^k.

Ejemplo 3.1.1.



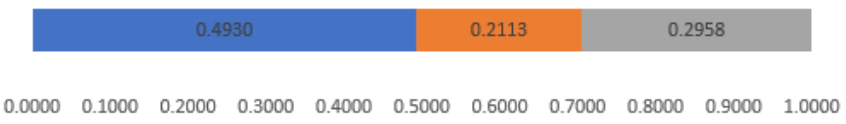
Para este ejemplo asignamos $\alpha = 1$, $\beta = 1$, $\tau = 0.1$, $\rho = 0.01$ *y* 2 *hormigas. Las hormigas se ubican inicialmente en el vértice A y deben volver al punto de inicio.*

- *Primera iteración:*

Caminos	**Longitudes**	η **(Visibilidad)**		τ **(inicializamos)**
A - B	3	1/3=	0.3333	0.1
A - C	7	1/7=	0.1429	0.1
A - D	5	1/5=	0.2000	0.1
B - C	2.5	1/2.5=	0.4000	0.1
B - D	6	1/6=	0.1667	0.1
C - D	4	1/4=	0.2500	0.1

Proceso para la hormiga 1:

Caminos	$\tau \times \eta$	**Probabilidad**
A - B	0.0333	0.4930
A - C	0.0143	0.2113
A - D	0.0200	0.2958
Sumatoria	**0.0676**	



Generamos un número aleatorio = 0.5500
Este número se encuentra en AC, entonces la hormiga se mueve al punto C.

Punto C:

Caminos	Longitudes	η (Visibilidad)		τ (inicializamos)
C - B	2.5	1/2.5=	0.400	0.1
C - D	4	1/4=	0.250	0.1

Hormiga 1	$\tau \times \eta$	Probabilidad
C - B	0.0400	0.6154
C - D	0.0250	0.3846
Sumatoria	**0.0650**	1

Generamos número aleatorio = 0.4500

Este número esta cae en CB entonces la hormiga se mueve al punto B

Punto B:

Caminos	Longitudes	η (Visibilidad)		τ (inicializamos)
B - D	6	1/6=	0.167	0.1

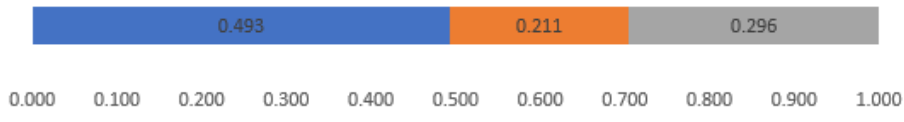
Hormiga 1	τ	Probabilidad
B - D	0.0167	1.0000
Sumatoria	**0.0167**	1

Entonces la hormiga se mueve al punto D

Ruta (Costo) = ACBDA = 25

Proceso para la hormiga 2:

Hormiga 2	$\tau \times \eta$	Probabilidad
A - B	0.0333	0.493
A - C	0.0143	0.211
A - D	0.0200	0.296
Sumatoria	**0.0676**	1



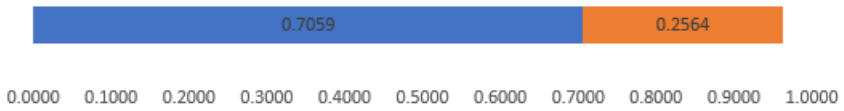
Generamos número aleatorio =0.1500

Este número esta cae en AB entonces la hormiga se mueve al punto B

Punto B:

Caminos	Longitudes	η (Visibilidad)		τ (inicializamos)
B - C	2.5	1/2.5=	0.400	0.1
B - D	6	1/6=	0.167	0.1

Hormiga 1	$\tau \times \eta$	Probabilidad
B - C	0.0400	0.7059
B - D	0.0167	0.2564
Sumatoria	**0.0567**	1



Generamos número aleatorio =0.6600

Este número esta cae en BC entonces la hormiga se mueve al punto C

Punto C:

Caminos	Longitudes	η (Visibilidad)		τ (inicializamos)
C - D	4	1/4=	0.250	0.1

Hormiga 1	$\tau \times \eta$	Probabilidad
C - D	0.0250	1.0000
Sumatoria	**0.0250**	

Entonces la hormiga se mueve al punto D

Ruta (Costo de la hormiga 2) =ABCDA = 14.5

Evaporación y actualización de Feromona

Caminos	Long.	η (Visibilidad)		τ	(1-ρ)*τ	H 1	H 2	nvo τ
A - B	3	1/3=	0.3333	0.1	0.0990	0.0000	0.0690	0.1680
A - C	7	1/7=	0.1429	0.1	0.0990	0.0400	0.0000	0.1390
A - D	5	1/5=	0.2000	0.1	0.0990	0.0400	0.0690	0.2080
B - C	2.5	1/2.5=	0.4000	0.1	0.0990	0.0400	0.0690	0.2080
B - D	6	1/6=	0.1667	0.1	0.0990	0.0400	0.0000	0.1390
C - D	4	1/4=	0.2500	0.1	0.0990	0.0000	0.0690	0.1680

3.2. Metaheurística inspirada en Algoritmos genéticos

Los algoritmos evolutivos (**EA**) son las metaheurísticas más influyentes para la optimización. El algoritmo genético (**GA**) es la forma más popular de los EA y fueron propuestos por John Henry Holland en 1975. Esta técnica está inspirada en la teoría de la evolución darwiniana[2].

Son básicamente una estrategia usada para problemas de búsqueda del óptimo basado en una heurística aleatoria. La idea consiste en simular la selección natural. La población inicial irá evolucionando a través de variaciones emergentes de cruces de los más aptos y de mutaciones.

3.2.1. Descripción de algoritmos genéticos

En la naturaleza los individuos de una población compiten entre sí en la búsqueda de recursos tales como comida, agua y refugio. Incluso los miembros de una misma especie compiten a menudo en la búsqueda de un compañero. Aquellos individuos que tienen más éxito en sobrevivir y en atraer compañeros tienen mayor probabilidad de generar un gran número de descendientes. Por el contrario individuos poco dotados producirán un menor número de descendientes. Esto significa que los genes de los individuos mejor adaptados se propagarán en sucesivas generaciones hacia un número de individuos creciente. La combinación de buenas características provenientes de diferentes ancestros, puede a veces producir descendientes, cuya adaptación es mucho mayor que la de cualquiera de sus ancestros. De esta manera, las especies evolucionan logrando unas características cada vez mejor adaptadas al entorno en el que viven.

Los Algoritmos Genéticos usan una analogía directa con el comportamiento natural. Trabajan con una población de individuos, cada uno de los cuales representa una solución factible a un problema dado. A cada individuo se le asigna un valor o puntuación, relacionado con la bondad de dicha solución. En la naturaleza esto equivaldría al grado de efectividad de un organismo para competir por unos determinados recursos. Cuanto mayor sea la adaptación de un individuo al problema, mayor será la probabilidad de que el mismo sea seleccionado para reproducirse, cruzando su material genético con otro individuo seleccionado de igual forma. Este cruce producirá nuevos individuos los cuales comparten algunas de las características de sus padres.

---

[2]Charles Robert Darwin (1809-1882) fue un naturalista inglés, reconocido por ser el científico más influyente de los que plantearon la idea de la evolución biológica a través de la selección natural. Postuló que todas las especies de seres vivos han evolucionado con el tiempo a partir de un antepasado común mediante un proceso denominado selección natural.

Cuanto menor sea la adaptación de un individuo, menor será la probabilidad de que dicho individuo sea seleccionado para la reproducción, y por tanto de que su material genético se propague en sucesivas generaciones.

De esta manera se produce una nueva población de posibles soluciones, la cual reemplaza a la anterior y verifica la interesante propiedad de que contiene una mayor proporción de buenas características en comparación con la población anterior. Así a lo largo de las generaciones las buenas características se propagan a través de la población. Favoreciendo el cruce de los individuos mejor adaptados, van siendo exploradas las áreas más prometedoras del espacio de búsqueda. Si el Algoritmo Genético ha sido bien diseñado, la población convergerá hacia una solución óptima del problema.

3.2.2. Desarrollo del algoritmo genético

Algunas terminologías que se utilizan en la literatura de computación evolutiva se enumeran a continuación. Estas terminologías son una analogía con sus contrapartes biológicas (Du and Swamy, 2016).

Definición 3.2.1. ***Población***. *Un conjunto de individuos en una generación se llama población,* $P(t) = x_1, x_2, \ldots, x_{NP}$ *, donde* x_i *es el i-ésimo individuo y* NP *es el tamaño de la población.*

Definición 3.2.2. ***Cromosoma***. *Cada individuo* x_i *en una población es un solo cromosoma. Un cromosoma, a veces llamado genoma, es un conjunto de parámetros que definen una solución al problema.*

Biológicamente, un cromosoma es un fragmento de ADN largo y continuo que contiene muchos genes, elementos reguladores y otras secuencias de nucleótidos que intervienen. Los miembros normales de una especie en particular tienen el mismo número de cromosomas.

Definición 3.2.3. ***Gen***. *En EA, cada cromosoma* x *se compone de una cadena de elementos* x_i*, llamados genes, es decir,* $x = (x_1, x_2, \ldots, x_n)$*, donde* n *es el número de genes en el cromosoma. Cada gen codifica un parámetro del problema en el cromosoma. Un gen generalmente se codifica como una cadena binaria o un número real.*

En biología, los genes son entidades que los padres transmiten a la descendencia durante la reproducción. Estas entidades codifican información esencial para la construcción y regulación de proteínas y otras moléculas que determinan el crecimiento y el funcionamiento del organismo.

Definición 3.2.4. ***Alelo***. *La definición biológica de un alelo es cualquiera de una serie de formas alternativas del mismo gen que ocupa una posición dada*

llamada locus en un cromosoma. La posición del gen en el cromosoma se llama locus.

Los alelos son las unidades de información más pequeñas en un cromosoma. En la naturaleza, los alelos existen en parejas, mientras que en los EA un alelo está representado por un solo símbolo e indica el valor de un gen.

Definición 3.2.5. ***Genotipo***. *Un genotipo se refiere biológicamente a la codificación genética subyacente de un organismo vivo, generalmente en forma de ADN. En EA, un genotipo representa una solución codificada, es decir, el cromosoma de un individuo.*

Definición 3.2.6. ***Fenotipo***. *Biológicamente, el fenotipo de un organismo es su apariencia física total y constitución o una manifestación específica de un rasgo. Hay un fenotipo asociado con cada individuo. El fenotipo de un individuo es el conjunto de todos sus rasgos (incluida su aptitud y su genotipo).*

El fenotipo está determinado por el genotipo o múltiples genes e influenciado por factores ambientales.

Definición 3.2.7. ***Fitness***. *La aptitud física en biología se refiere a la capacidad de reproducción de un individuo de cierto genotipo. El conjunto de todos los genotipos posibles y sus respectivos valores de estado físico se denomina paisaje de estado físico.*

La función de condición física es un tipo particular de función objetivo que cuantifica la optimización de una solución, es decir, un cromosoma, en un EA. Se utiliza para mapear el cromosoma de un individuo en un número positivo. La aptitud es el valor de la función objetivo para un cromosoma x_i, es decir, $f(x_i)$. La función fitness se utiliza para convertir los valores de los parámetros del fenotipo en fitness.

3.2.3. Codificación de variables

Para poder trabajar con los algoritmos genéticos es necesario codificar el cromosoma, el cual contiene varios genes, los cuales corresponden a los parámetros del problema. Existen tres formas principales de codificación de variables:

- En cadenas binarias, donde la posición de los ceros y unos representa el valor de algún aspecto de la solución.

- En cadenas de enteros o números, que al igual que en las cadenas binarias la posición representa alguna característica particular. Este método tiene mayor precisión y por lo tanto también complejidad que el sistema binario.
- En cadenas de letras, donde cada letra representa algún aspecto de la solución.

Si bien el alfabeto utilizado para representar los individuos no debe necesariamente estar constituido por el $\{0, 1\}$, buena parte de la teoría en la que se fundamentan los Algoritmos Genéticos utiliza dicho alfabeto.

La codificación para ruteo de vehículos consiste en una secuencia de nodos a visitar, siendo el primer y último gen el depósito.

D	N3	N1	N4	N7	N5	N6	N2	D

3.2.4. Proceso de los Algoritmos Genéticos

Para la búsqueda de soluciones del problema, el algoritmo genético sigue una serie de pasos:

1. **Inicialización:** la población inicial de candidatos es generada aleatoriamente en el espacio de búsqueda.
2. **Evaluación:** una vez iniciada la población o una población descendiente, los valores de aptitud de las soluciones candidatas son evaluadas.
3. **Condición de término:** El AG se deberá detener cuando se alcance la solución óptima, pero esta generalmente se desconoce, por lo que se deben utilizar otros criterios de detención. Normalmente se usan dos criterios: correr el AG un número máximo de iteraciones (generaciones) o detenerlo cuando no haya cambios en la población. Mientras no se cumpla la condición de término se hace lo siguiente:
   a) **Selección:** la idea general de este paso es la preferencia de las mejores soluciones de las peores, haciendo uso de procedimientos de selección, como la selección por ruleta, selección elitista, selección escalada, selección por torneo, por mencionar algunos.
   b) **Recombinación:** el procedimiento combina partes de dos o más soluciones aparentes para crear nuevas soluciones, posiblemente mejores.

c) **Mutación:** mientras la recombinación opera con 2 o más cromosomas parentales, la mutación localmente, pero aleatoriamente altera una solución.

4. Se repiten los pasos del 1 al 3, hasta encontrar la condición final deseada.

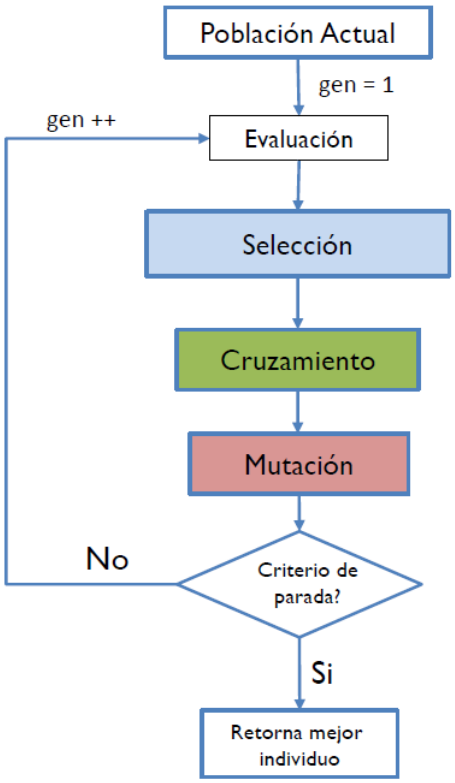


Figura 3.2: Funcionamiento de un algoritmo genético básico.

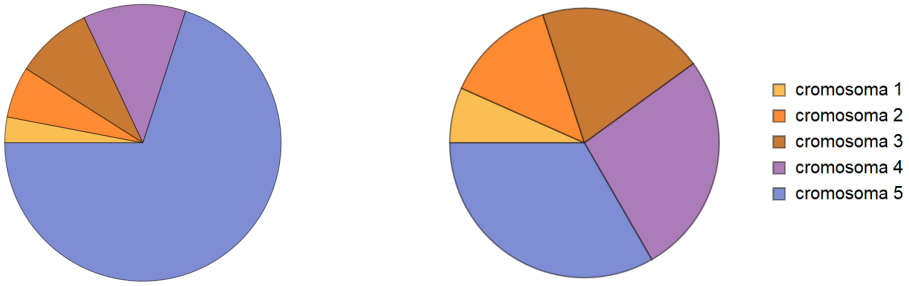
3.2.5. Operadores de un algoritmo genético

- **Selección** La evaluación se encarga de decodificar los genes del cromosoma, para convertirlos en los parámetros del problema, para después poder encontrar una solución a partir de ellos. Dentro de la evaluación se califica la solución en función de lo cercano que esté la solución a las condiciones requeridas, a esto se le llama "fitness" (en inglés). El "fitness" determina siempre los cromosomas que se van a reproducir, y aquellos que se van a eliminar.

  Existen varias técnicas de selección que se pueden utilizar en los algoritmos genéticos, para poder pasar a la generación siguiente, enseguida se presentan algunos:

  - **Selección por Ranking:** Los individuos son ordenados de acuerdo a su ranking de fitness. De esta manera si tenemos n cromosomas el individuo con peor fitness se le asignará un 1 y el que tenga el mejor fitness se le asignará 1 a n.

    Véase en las dos siguientes figuras cómo cambia la situación antes y después del ranking.



(a) Situación antes del Ranking (b) Situación después del Ranking

Figura 3.3: Comparación de situaciones antes y después del ranking.

    Ahora todos los cromosomas tienen la oportunidad de ser seleccionados. Sin embargo este método puede hacer que el genético converja lentamente a la solución.

  - **Selección por torneo:** se eligen aleatoriamente subgrupos de individuos de la población, y los miembros compiten entre ellos. Solo se elige a un individuo de cada subgrupo.

- **Operador de cruce**

Los operadores de cruce tratan de crear una generación de individuos nuevos pidiendo información a sus ancestros. Aunque estos operadores parecen corresponderse con la representación basada en precedencia.

- **Cruce en orden:** Elegir aleatoriamente 2 puntos de corte en los padres

Padre 1

D	N3	N6	N1	N5	N4	N7	N2	D

Padre 2

D	N1	N5	N2	N4	N6	N3	N7	D

Copiar el segmento generado entre los puntos de corte del primer padre (P1) al primer hijo (H1), desconsiderar en el segundo padre los valores que ya existen en H1, a partir de la derecha del 2 do punto de corte, copiar del P2 al H1 los genes que no fueron desconsiderados. Respetar el orden en que aparecen en el P2.

Hijo 1

D	N4	N6	N1	N5	N3	N7	N2	D

Repetir el procedimiento para encontrar el segundo hijo (H2)

Hijo 2

D	N1	N5	N2	N4	N7	N3	N6	D

Figura 3.4: Cruce en orden.

- **Operador de mutación**

  - **Inversión:** altera el orden y conserva la información de adyacencia. Elegir 2 genes al azar y luego invertir la subcadena entre ellos

D	N3	N1	N5	N7	N4	N6	N2	D

D	N3	N1	N6	N4	N7	N5	N2	D

  - **Swan:** no conserva el orden, pero si la información de adyacencia

D	N3	N1	N5	N7	N4	N6	N2	D

D	N3	N1	N4	N7	N5	N6	N2	D

- **Inserción:** conserva el orden y la información de adyacencia Elegir 2 genes aleatoriamente, mover el segundo alelo al lado del primero, luego desplazar el resto de alelos

D	N3	N1	N5	N7	N4	N6	N2	D

D	N3	N1	N6	N5	N7	N4	N2	D

- **Perturbación:** altera el orden y la información de adyacencia, elegir una subcadena de genes al azar y luego reorganizarla aleatoriamente.

D	N3	N1	N5	N7	N4	N6	N2	D

D	N3	N1	N4	N6	N7	N5	N2	D

3.2.6. Algoritmos genéticos aplicado a problemas de ruteo de vehículos

La correspondencia de los elementos definidos en este capítulo con el problema de estudio, es el siguiente: los individuos son representados como una secuencia de nodos, siendo el primer y último nodo, el depósito, y una secuencia de puntos afectados que visitará un vehículo, cada nodo representa un gen. Se encoge un conjunto de rutas (individuos de la población), la cantidad obtenida al evaluar este individuo en la función objetivo será el fitness o también llamada función de adaptación, que representa el grado de adaptación con el medio, en este caso existe una relación inversamente proporcional entre el fitness y sobrevivir, dado que es una función que minimiza los tiempos de ruteo, así que a menor fitness será considerado una solución tentativa a ser el óptimo, estos individuos se cruzan generando más individuos, si la descendencia muestra mejor fitness, tiene mayor probabilidad de sobrevivir, los que presentan menor fitness, tienen mayor probabilidad de desaparecer, pero en menor grado, un descendiente que no tiene un buen fitness sobrevive, esto sirve para no quedar estancados en óptimos locales. Suele aplicarse también la mutación, que consiste en intercambiar la posición de dos o más nodos que no sean los extremos de la codificación de un individuo.

Ejemplo 3.2.1. *Supóngase que una empresa está planteándose la ubicación en su planta de producción de 10 secciones diferentes:* S_1, S_2,..., S_{10}. *Para ello ha dividido su planta en 10 áreas diferentes, tal como muestra la figura (3.5).*

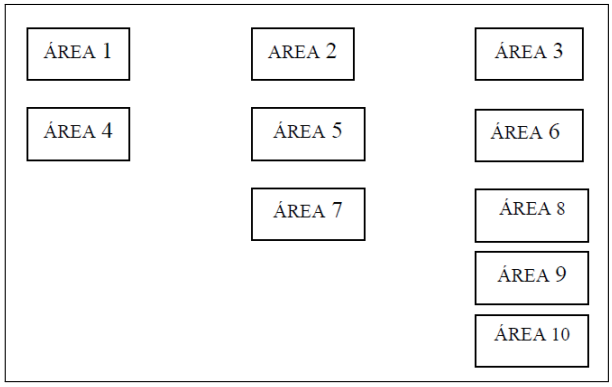


Figura 3.5: Disposición de las diferentes áreas en planta de producción.

Determinadas secciones deberían estar situadas próximas debido al flujo de materiales entre ambas, por ejemplo supóngase que estos flujos son los que aparecen en la Tabla (3.1).

	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10
S1	-	10	15	12	-	-	-	21	-	5
S2	-	-	13	8	-	10	11	-	-	-
S3	20	-	-	5	10	-	-	15	8	6
S4	-	8	-	-	-	-	10	12	-	8
S5	10	12	-	-	-	8	6	4	-	-
S6	-	-	-	-	-	-	10	12	-	-
S7	-	-	-	-	-	12	-	8	-	-
S8	-	-	-	-	-	6	-	-	-	-
S9	12	6	4	-	-	-	-	-	-	-
S10	4	8	10	-	-	-	-	-	-	-

Tabla 3.1: Flujos entre secciones

Además de los flujos de mercancías, han de tenerse en cuenta las distancias entre las áreas de la planta, ya que el coste del transporte de la mercancía será proporcional a la distancia que debe recorrer. Las distancias entre las áreas se muestra en la Tabla (3.2). Tal como queda planteado en el problema, el número de posibles colocaciones de las secciones en la planta es:

$$10! = 3628800$$

Una cifra considerable como para poder analizar todas esas posibilidades. El objetivo se plantea como búsqueda de una distribución óptima en el sentido

de minimizar el coste total del transporte de las mercancías entre las distintas secciones. Dicho coste total de transporte (CTT) viene dado por:

$$\sum_{i=1}^{10}\sum_{j=1}^{10} t_{ij}d_{ij}c_{ij}$$

Donde t_{ij} es el número de unidades a mover entre las secciones S_i y S_j, d_{ij} es la distancia existente desde el área i hasta el j y C_{ij} es el coste por unidad de distancia y carga desde i hacia j. En el ejemplo considerado se supondrá que ese coste es igual en todos los desplazamientos, es decir, C_{ij} constante.

Dado el número extraordinario elevado de posibles combinaciones existentes, que puede llegar a impedir obtener soluciones óptimas, para la resolución de este tipo de problemas se ha recurrido a diversas opciones, desde la utilización de algoritmos heurísticos, que al menos proporcionan soluciones factibles y satisfactorias, a técnicas informatizadas como CRAFT, SPACECRAFT, ALDEP o incluso al desarrollo de sistemas expertos, como FADES (Fisher,1986). En este caso, se utilizara un algoritmo genético para mostrar su aplicabilidad a problemas concretos.

	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10
A1	0	1	2	1	2	3	3	4	5	6
A2		0	1	2	1	2	2	3	4	5
A3			0	3	2	1	3	2	3	4
A4				0	1	2	2	3	4	5
A5					0	1	1	2	3	4
A6						0	2	1	2	3
A7							0	1	2	3
A8								0	1	2
A9									0	1
A10										0

Tabla 3.2: Distancias entre las áreas. Se asume que la matriz de distancias es simétrica.

Las peculiaridades del problema a tratar hacen necesarios una serie de ajuste en la representación genética de las posibles soluciones y en los operadores genéticos a utilizar.

Representación de soluciones potenciales *Toda posible solución del problema de distribución en planta debe ser codificada mediante una cadena de longitud finita en un alfabeto también finito. En este caso el alfabeto binario no resulta apropiado ya que los operadores de cruce y mutación podrían producir soluciones no factibles.*

La representación más adecuada es utilizar un vector de números enteros, de manera que toda posible solución (individuo) puede representarse a través de una permutación $P = (S_1, S_2, \ldots, S_n)$ *de* $(1, 2, \ldots, n)$*, donde* $P_i = S_j$ *indica que la selección* S_j *está localizada en el área* A_i*. Por ejemplo, el individuo* $(4, 5, 6, 1, 2, 8, 9, 10, 3, 7)$ *corresponde a una distribución en la que las secciones cuarta, quinta y sexta están situadas en las tres primeras áreas, las secciones primeras y segunda en las áreas cuarta y quinta, etc.*

Función objetivo (Función de ajuste) *En el problema considerado el objetivo se establece en términos de la minimización de la función de coste total de trasporte, CTT, sin embargo, todo algoritmo genético tiene por objetivo la maximización de una función de ajuste positiva. En este caso, se podría definir la función de ajuste de la siguiente manera:*

$$F(P) = Cmax - CTT(P)$$

Donde C_{max} *puede definirse como el mayor valor encontrado hasta el momento en todas las evaluaciones de la función CTT, es decir,*

$$Cmax = maxCTT(P)$$

Población inicial *Como población inicial del problema se ha optado por generar un conjunto de* $n = 30$ *soluciones potenciales de forma totalmente aleatoria. La Tabla (3.3) muestra las posibles distribuciones obtenidas junto son sus respectivos costes totales de transporte.*

DISTRIBUCIÓN POTENCIAL	COSTE	DISTRIBUCIÓN POTENCIAL	COSTE
$P_1 = (1, 10, 2, 6, 8, 4, 3, 7, 9, 5)$	840	$P_{16} = (8, 6, 10, 5, 4, 1, 3, 7, 9, 2)$	867
$P_2 = (5, 6, 10, 7, 1, 9, 4, 8, 3, 2)$	922	$P_{17} = (3, 10, 9, 1, 2, 5, 4, 8, 7, 6)$	672
$P_3 = (8, 2, 7, 5, 9, 1, 4, 10, 3, 6)$	1002	$P_{18} = (9, 10, 2, 6, 4, 5, 3, 7, 1, 8)$	898
$P_4 = (9, 4, 5, 3, 10, 2, 8, 1, 6, 7)$	749	$P_{19} = (1, 7, 3, 8, 9, 6, 4, 5, 10, 2)$	903
$P_5 = (5, 10, 7, 2, 4, 1, 9, 3, 8, 6)$	768	$P_{20} = (10, 1, 6, 5, 3, 8, 2, 7, 4, 9)$	710
$P_6 = (5, 10, 8, 3, 4, 7, 9, 2, 1, 6)$	957	$P_{21} = (8, 2, 6, 7, 10, 5, 1, 3, 9, 4)$	854
$P_7 = (5, 6, 10, 3, 2, 4, 7, 1, 8, 9)$	812	$P_{22} = (5, 6, 3, 1, 4, 9, 2, 8, 10, 7)$	920
$P_8 = (4, 7, 6, 10, 1, 3, 9, 5, 8, 2)$	880	$P_{23} = (6, 7, 1, 5, 2, 10, 9, 4, 8, 3)$	902
$P_9 = (4, 10, 6, 2, 7, 1, 8, 3, 9, 5)$	784	$P_{24} = (8, 1, 7, 10, 2, 9, 3, 6, 4, 5)$	905
$P_{10} = (8, 4, 1, 9, 7, 5, 10, 2, 6, 3)$	978	$P_{25} = (7, 5, 2, 1, 4, 10, 6, 9, 8, 3)$	964
$P_{11} = (5, 9, 7, 2, 1, 8, 4, 10, 6, 3)$	919	$P_{26} = (2, 1, 4, 3, 8, 6, 7, 10, 5, 9)$	844
$P_{12} = (3, 7, 6, 10, 8, 5, 4, 1, 9, 2)$	875	$P_{27} = (10, 2, 8, 9, 5, 6, 4, 3, 7, 1)$	926
$P_{13} = (6, 10, 4, 3, 7, 1, 8, 2, 9, 5)$	852	$P_{28} = (8, 9, 4, 7, 6, 2, 3, 5, 1, 10)$	784
$P_{14} = (1, 4, 5, 7, 3, 10, 2, 8, 6, 9)$	853	$P_{29} = (9, 4, 3, 8, 7, 6, 2, 10, 1, 5)$	858
$P_{15} = (3, 2, 5, 9, 8, 10, 1, 4, 6, 7)$	802	$P_{30} = (3, 6, 2, 8, 7, 1, 4, 5, 10, 9)$	829

Tabla 3.3: Población inicial generada de forma aleatoria.

Como puede observarse, la población inicial tiene un rango de costes asociados en el intervalo [cmin, cmax]= [672,1002] , siendo además el coste medio para la población de 860.98 y una desviación típica de 77.67

Operadores de cruce. *El operador de cruce simple analizado anteriormente es claramente ineficaz en este problema, por cuanto conduce a individuos sin ningún sentido. Obsérvese el ejemplo ilustrado en la Figura (3.6), ninguno de los descendientes obtenidos es admisible ya que en ambos casos existen secciones que quedarían sin ubicación física y por el contrario existen secciones con dos áreas asignadas.*

Figura 3.6: Ejemplo de inconsistencia del operador de cruce tradicional

Se hace necesario, por tanto, algún otro tipo de operador de cruce. Ejemplos de operadores de cruce utilizados en problemas con una estructura similar son:

- *El operador de Emparejamiento Parcial (Partial Matched Crossover, PMX).*
- *El operador de cruzamiento Ordenado (Order Crossover, OX).*
- *El operador de Cruzamiento Ciclico (Cycle Crossover, CX).*

El operador PMX (propuesto por Goldberg y Lingle), construye dos descendientes eligiendo al azar dos posiciones de cruce sobre las representaciones genéticas de los progenitores. Estos dos puntos definen una sección de emparejamiento que es usada para efectuar el cruce mediante operaciones de intercambio de elementos sobre los vectores progenitores. En la figura (3.7) puede verse un ejemplo de aplicación del operador de cruce PMX.

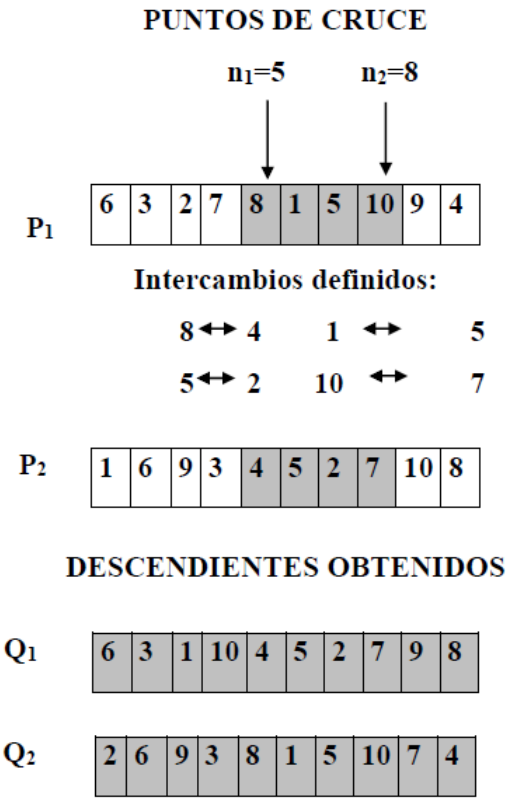


Figura 3.7: Ejemplo de cruce utilizado el operador PMX.

Como puede verse en la Figura (3.7), para obtener los dos descendientes, primero se intercambian los segmentos situados entre las posiciones de cruce obtenidas aleatoriamente. Este intercambio define Además una serie de cambios puntuales ($8 \leftrightarrow 4$, $1 \leftrightarrow 5$, $5 \leftrightarrow 2$, *y* $10 \leftrightarrow 7$*) que deben ser aplicados sobre* P_1 *y* P_2*. Como resultado de este proceso se obtienen dos nuevos vectores (descendientes):* $Q_1 = (6, 3, 5, 10, 4, 5, 2, 7, 9, 8)$ *y* $Q_2 = (2, 6, 9, 3, 8, 1, 5, 10, 7, 4)$*; El operador OX (propuesto por Davis) comienza de una manera similar al PMX, es decir, eligiendo de forma aleatoria dos posiciones de cruce sobre los vectores y las correspondientes relaciones ente los elementos. A continuación el operador OX realiza un movimiento de desplazamiento para cubrir los huecos dejados por los elementos del vector a intercambiar; este movimiento de desplazamiento comienza a partir de la segunda posición de cruce. Posteriormente se producen intercambios de elementos definidos por la sección de emparejamiento. Por ejemplo, considerando las permutaciones* P_1 *y* P_2 *anteriores y los puntos de cruce* $n_1 = 5$ *y* $n_2 = 8$*, las nuevas permutaciones que se obtienen con el operador OX son:* $Q_1 = (3, 8, 1, 10, 4, 5, 2, 7, 9, 6)$ *y* $Q_2 = (3, 4, 2, 7, 8, 1, 5, 10, 6, 9)$*. El proceso de obtención de estos descendientes es ilustrado en la figura (3.8).*

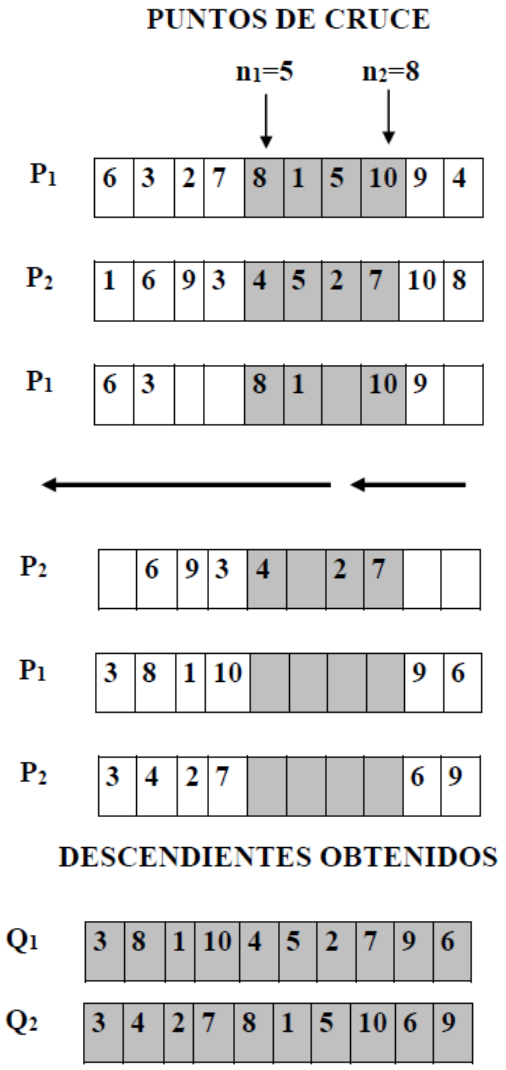


Figura 3.8: Ejemplo de cruce con el operador OX.

La principal diferencia entre los operadores PMX OX es el hecho de que, mientras PMX tiende a respetar las posiciones absolutas de los elementos dentro de los vectores, OX tiende a respetar las posiciones relativas.

Finalmente, el operador CX (propuesto por Oliver) construye los descendiente de tal forma que cada elemento de los nuevos vectores (tanto su valor como su occisión dentro del vector) viene de uno de sus vectores progenitores. Por ejemplo, considerando de nuevo P_1 *y* P_2 *se toma el primer elemento del primer vector; como todo elemento debe ser tomado de uno de sus progenitores, la elección del elemento 6 de* P_1 *obliga a coger el elemento 1 de* P_1 *. Esta regla se sigue aplicando hasta que se complete un ciclo, momento en el cual los restantes elementos se completan del otro vector progenitor. De esa forma se obtendrían como descendiente* $Q_1 = (6, 3, 2, 7, 4, 1, 5, 10, 9, 8)$ *y* $Q_2 = (1, 6, 9, 3, 8, 5, 2, 7)$ *la*

Figura (3.9) ilustra de nuevo el proceso de obtención de los descendientes.

Los tres operadores analizados (PMX, CX y OX), para el problema de distribución en planta considerado, obtienen soluciones admisibles (posibles distribuciones de las secciones en la áreas) a partir de dos soluciones admisibles.

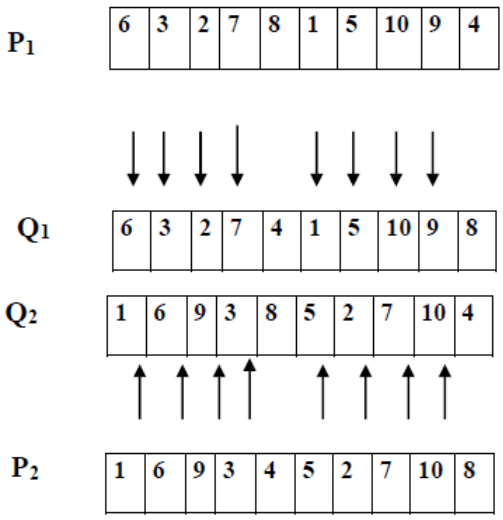


Figura 3.9: Ejemplo de cruce con el operador OX.

Operador de mutación *El operador de mutación para el problema considerado puede consistir en intercambiar las posiciones de dos elementos del vector, posiciones elegidas de manera aleatoria. Por ejemplo, dada la distribución* $P = (1, 4, 7, 2, 8, 9, 5, 3, 10, 6)$ *y las posiciones de mutación* $n_1 = 2$ *y* $n_2 = 7$*, elegidas aleatoriamente, el resultado de la mutación sería una distribución ligeramente diferentes:* $P' = (1, 5, 7, 2, 8, 9, 4, 3, 10, 6)$.

Resultados Experimentales. *Para probar la efectividad del algoritmo, se implementó un algoritmo genético con las siguientes características:*
Tamaño de población: 30
Número de Generaciones 30
Operador de Cruce: PMX
Probabilidad de cruce: 0.7
Probabilidad de Mutación: 0.1

De igual forma, la Tabla (3.4) presenta un informa estadístico sobre el número de cruces y mutaciones producidas, y los cortes mínimo, máximo y medio obtenidos en cada una de las 30 generaciones del experimento.

GENERACIÓN	N° CRUCES	N° MUT.	MEDIDA	DESV. TIPICA	MIN	MAX
1	11	0	897.1	72.9456	759	1040
2	11	2	852.3	63.3753	759	982
3	12	5	842.133	61.8506	745	1040
4	15	6	829.267	58.0909	711	946
5	10	3	818.733	82.6872	708	1002
6	11	2	788.733	51.0341	690	873
7	15	4	778.633	57.3913	675	881
8	9	4	765.1	63.553	633	908
9	12	2	771	77.0298	609	929
10	9	3	762.633	83.4363	653	1079
11	12	6	741.8	69.7065	651	883
12	10	3	737.633	69.9199	633	883
13	13	1	710.6	62.3055	636	859
14	8	1	683.9	43.2812	626	787
15	11	1	681.033	57.6287	626	867
16	13	6	684.767	46.3217	626	797
17	13	3	672.2	42.9686	627	794
18	10	4	667.367	46.537	626	859
19	11	3	653.533	26.6959	621	749
20	11	3	661.467	52.5618	621	815
21	12	2	641.7	20.3455	609	695
22	8	4	644.767	39.8434	609	814
23	14	5	642.467	31.581	598	748
24	8	3	642.467	34.0838	609	750
25	11	8	650.5	52.3218	598	824
26	8	5	649.367	48.84	598	791
27	10	1	639.867	39.1282	598	764
28	8	2	624.567	21.0708	598	693
29	8	3	619.8	19.6704	598	695
30	11	3	619.967	33.1178	598	781

Tabla 3.4: Informe de evolución.

Finalmente en la Tabla (3.5) puedes verse la última generación de soluciones potenciales obtenida. La mejor distribución después de las 30 generaciones es: $P = (9, 3, 5, 10, 1, 2, 4, 8, 6, 7)$ *con un coste asociado de* 598. *Observar también que se ha obtenido un conjunto de buenas distribuciones, además pueden descubrirse importantes similitudes (esquemas) entre los individuos de esta última generación. Por ejemplo: todas las potenciales distribuciones sitúan la sección* 9 *en el primer área, y las sección* 6, 7 *y* 8 *siempre quedan colocadas en las* 3 *últimas áreas.*

DISTRIBUCIÓN POTENCIAL	COSTE	DISTRIBUCIÓN POTENCIAL	COSTE
$P_1 = (9,3,4,10,1,2,5,8,7,6)$	624	$P_{16} = (9,3,5,10,1,2,4,8,7,6)$	609
$P_2 = (9,1,5,10,3,2,4,8,6,7)$	601	$P_{17} = (9,10,5,3,1,2,4,8,6,7)$	618
$P_3 = (9,3,2,10,1,5,4,8,6,7)$	616	$P_{18} = (9,3,5,10,1,2,4,8,7,6)$	609
$P_4 = (9,3,2,10,1,5,4,8,7,6)$	627	$P_{19} = (9,3,5,10,1,2,4,8,6,7)$	598
$P_5 = (9,1,5,10,3,2,4,8,7,6)$	612	$P_{20} = (9,10,5,3,1,2,4,8,6,7)$	618
$P_6 = (9,3,4,10,1,2,5,8,7,6)$	624	$P_{21} = (9,10,5,3,1,2,4,8,7,6)$	629
$P_7 = (9,3,5,10,1,2,4,8,6,7)$	598	$P_{22} = (9,1,5,10,3,2,4,8,6,7)$	601
$P_8 = (9,3,5,10,1,2,4,8,7,6)$	609	$P_{23} = (9,3,4,10,1,2,5,8,7,6)$	624
$P_9 = (9,3,2,10,1,5,4,8,7,6)$	627	$P_{24} = (9,3,5,10,1,2,4,8,7,6)$	609
$P_{10} = (9,8,5,10,3,2,4,1,7,6)$	781	$P_{25} = (9,3,5,10,1,2,4,8,6,7)$	598
$P_{11} = (9,3,5,10,1,2,4,8,6,7)$	598	$P_{26} = (9,3,4,1,10,2,5,8,6,7)$	657
$P_{12} = (9,3,5,10,1,2,4,8,7,6)$	609	$P_{27} = (9,3,5,10,1,2,4,8,7,6)$	609
$P_{13} = (9,3,4,10,1,2,5,8,7,6)$	624	$P_{28} = (9,3,5,10,1,2,4,8,6,7)$	598
$P_{14} = (9,3,5,10,1,2,4,8,7,6)$	609	$P_{29} = (9,3,1,10,2,5,4,8,6,7)$	610
$P_{15} = (9,10,5,3,1,2,4,8,7,6)$	629	$P_{30} = (9,3,4,10,1,2,5,8,7,6)$	624

Tabla 3.5: Población después de 30 generaciones

Evidentemente, el algoritmo descrito puede ser refinado, incluyendo aspectos como penalizaciones a determinados individuos, eliminación de duplicados, poblaciones de tamaño visible, etc. Sin embargo, para los propósitos de esta lección es claramente ilustrativo del modo de funcionamiento de los algoritmos genéticos.

CAPÍTULO IV

ELABORACIÓN Y SOLUCIÓN DEL MODELO.

En este capítulo se presenta el modelo de optimización basado en los modelos clásico del Problema de Ruteo de Vehículos, además la aplicación de una metaheurística basado en algoritmos genéticos. Finalmente, para evaluar el modelo propuesto, se muestra un supuesto, trabajado con datos recogidos de la ciudad de Piura frente a un eventual fenómeno del niño costero, como el ocurrido en el año 2017.

(a) Radio Cutivalú

(b) WordPress.com

(c) La Hora

(d) Walac Noticias

Figura 4.1: Daños causados por el fenómeno del niño costero de 2017

(a) El Comercio

(b) Radio Cutivalú

(c) El Comercio

(d) RPP

(e) Peru21

(f) Andina

Figura 4.2: Daños causados por el desborde del rio Piura

4.1. Modelo de optimización propuesto.

4.1.1. Supuestos

De acuerdo a la documentación obtenida, se puede dar los siguientes supuestos.

- El número de vehículos de socorro es limitado y se aplican diferentes tipos de ellos para atender áreas afectadas. En consecuencia, la capacidad de los vehículos es heterogéneo.
- En el caso de un incidente a gran escala, se requerirá el uso de todos los vehículos.
- La ubicación de la distribución de productos de socorro y los depósitos es la misma.
- Las operaciones de distribución tienen como propósito entregar un kit que contiene el botiquín de primeros auxilios, productos enlatados, agua mineral, mantas y tiendas de campaña.
- Se conoce la ubicación de cualquier área afectada, así como su distancia de los depósitos.
- La cantidad de demanda en cada punto afectado es conocido.
- El plazo de entrega de los bienes de socorro no debe superar las 72 horas.
- La velocidad promedio de los vehículos distribuidores será de 50 km/h.
- Los nodos con carretera dañada solo pueden ser abastecidos por helicópteros.
- Los nodos con carretera en buen estado solo pueden ser abastecidos por camiones.

Una vez considerado estos supuestos y basados en (Gharib et al., 2018), procedemos a realizar la notación que usa el modelo propuesto en la presente investigación:

4.1.2. Variables

- C Conjunto de vehículos terrestres-conjunto de camiones.
- H Conjunto de vehículos aéreos-conjunto de helicópteros.

- P_t Conjunto de puntos afectados con camino transitable.
- P_d Conjunto de puntos afectados con carretera dañada.
- D_c Depósito de vehículos terrestres.
- D_h Depósito de vehículos aéreos.
- n_t Número de puntos con camino transitable.
- n_d Número de puntos con carretera dañada.
- $T = P_t \cup D_c$ conjunto de vehículos.
- $A = P_d \cup D_h$ conjunto de vehículos.

4.1.3. Parámetros

- Cap_v Capacidad del vehículo tipo v.
- Cap_d Capacidad del depósito d.
- Dem_i Demanda del nodo i (punto afectado).
- $t_{vij} = d_{ij}/v$ Tiempo de viaje del vehículo v desde el nodo i al nodo j.
- d_{ij} distancia entre los puntos i, j.
- $v_c = 50km/h$ velocidad promedio de los camiones.
- $v_h = 260km/h$ velocidad promedio de los helicópteros.
- U_{vi} Variable auxiliar y secuencial que muestra el número de nodos que visita el camión v en restricciones de eliminación de subtour.
- U'_{vi} Variable auxiliar y secuencial que muestra el número de nodos que visita el helicóptero v en restricciones de eliminación de subtour.

4.1.4. Variable de decisión

$$x_{vij} = \begin{cases} 1, & \text{si el camión } v \text{ viaja desde el nodo } i \text{ hacia el nodo } j \\ 0, & \text{en otro caso} \end{cases}$$

$$y_{vij} = \begin{cases} 1, & \text{si el helicóptero } v \text{ viaja desde el nodo } i \text{ hacia el nodo } j \\ 0, & \text{en otro caso} \end{cases}$$

4.1.5. Modelo Matemático

Ahora, el modelo propuesto puede formularse matemáticamente mediante:

$$y_{vij} \in \{0,1\} \quad \forall v,i,j \tag{4.1}$$

$$\text{mín}\left(\sum_{i\in T}\sum_{j\in T}\sum_{v\in C} t_{vij}x_{vij} + \sum_{i\in A}\sum_{j\in A}\sum_{v\in H} t_{vij}y_{vij}\right) \tag{4.2}$$

sujeto a

$$\sum_{i\in P_t\cup D_c} x_{vij} = \sum_{i\in P_t\cup D_c} x_{vji} \quad \forall j \in P_t \cup D_c, v \in C, i \neq j \tag{4.3}$$

$$\sum_{i\in P_d\cup D_h} y_{vij} = \sum_{i\in P_d\cup D_h} y_{vji} \quad \forall j \in P_d \cup D_h, v \in H, i \neq j \tag{4.4}$$

$$\sum_{v\in C}\sum_{j\in P_t\cup D_c} x_{vij} = 1 \quad \forall i \in P_t \tag{4.5}$$

$$\sum_{v\in H}\sum_{j\in P_d\cup D_h} y_{vij} = 1 \quad \forall i \in P_d \tag{4.6}$$

$$\sum_{v\in C}\sum_{i\in P_t\cup D_c} x_{vij} = 1 \quad \forall j \in P_t \tag{4.7}$$

$$\sum_{v\in H}\sum_{i\in P_d\cup D_h} y_{vij} = 1 \quad \forall j \in P_d \tag{4.8}$$

$$\sum_{i\in P_t\cup D_c}\sum_{j\in P_t} x_{vij}\,\text{dem}_j \leq \text{Cap}_v \quad \forall v \in C, i \neq j \tag{4.9}$$

$$\sum_{i\in P_d\cup D_h}\sum_{j\in P_d} y_{vij}\,\text{dem}_j \leq \text{Cap}_v \quad \forall v \in H, i \neq j \tag{4.10}$$

$$\sum_{v\in C}\sum_{i\in P_t\cup D_c}\sum_{j\in P_t} x_{vij}\,\text{dem}_j \leq \text{Cap}_d, \quad d \in D_c \quad , i \neq j \tag{4.11}$$

$$\sum_{v\in H}\sum_{i\in P_d\cup D_h}\sum_{j\in P_d} y_{vij}\,\text{dem}_j \leq \text{Cap}_d \quad , d \in D_h, \quad i \neq j \tag{4.12}$$

$$U_{vi} - U_{vj} + n_e \cdot x_{vij} \leq n_e - 1 \quad \forall v \in C, \forall i,j \in P_t, i \neq j \tag{4.13}$$

$$U'_{vi} - U'_{vj} + n_{ee} \cdot y_{vij} \leq n_{ee} - 1 \quad \forall v \in H, \forall i,j \in P_d, i \neq j \tag{4.14}$$

$$U_{vi} \leq n_e \quad \forall v \in C, \forall i \in P_t \tag{4.15}$$

$$U_{v0} = 0 \quad \forall v \in C, \forall i \in P_t \tag{4.16}$$

$$U'_{vi} \leq n_{ee} \quad \forall v \in H, \quad \forall i \in P_d \tag{4.17}$$

$$U'_{v0} = 0 \quad \forall v \in H, \forall i \in P_d \tag{4.18}$$

$$U_{vi} = \{0, 1, 2, \ldots\} \tag{4.19}$$

$$U'_{vi} = \{0, 1, 2, \ldots\} \tag{4.20}$$

$$x_{vij} \in \{0, 1\} \quad \forall v, i, j \tag{4.21}$$

La función objetivo (4.2) minimiza el tiempo de transporte del vehículo v entre el nodo i y el nodo j para flota terrestre y aérea, además de minimizar la de CO_2. La restricción (4.3) garantiza el equilibrio de flujo para los puntos afectados con un camino saludable asistidos por vehículos terrestres. Es decir, cada camión después de ingresar al nodo y servir el punto sale del nodo. La restricción (4.4) garantiza el equilibrio de flujo para puntos no saludables asistidos por helicóptero. En otras palabras, los helicópteros abandonan el nodo después de la entrada. La restricción (4.5) indica que un camión solo puede ingresar una vez a un nodo con carretera transitable, mientras que la restricción (4.6) indica que un helicóptero puede ingresar solo una vez a un punto con carretera dañada. Las restricciones (4.7) y (4.8) garantizan que cualquier vehículo (es decir, camión o helicóptero) después de prestar servicio a los nodos deben salir del mismo y continuar su ruta. Las restricciones (4.9) y (4.10) son las limitaciones de capacidad de camiones y helicópteros, mientras que las restricciones (4.11) y (4.12) son las limitaciones de capacidad de los depósitos de camiones y depósito de helicópteros respectivamente. La parte considerada como restricción de sub-tour está representada en las restricciones (4.13) a (4.20), entre las cuales las restricciones (4.13) y (4.14) son las restricciones de eliminación de sub-tour para camiones y helicópteros, restricciones (4.15) y (4.16) son las restricciones de eliminación de sub-tour para las variables axilares, U_{vi} y U_{v0}, para camiones, y las restricciones (4.17) y (4.18) son las restricciones de eliminación de sub-tour para las variables axilares U'_{vi} y $U'_{v0)}$ para helicópteros Finalmente, las restricciones (4.19 y 4.20) se refiere a la secuencia de U_{iv} y las restricciones (4.21 y 4.1) se refieren a las variables binarias x_{vij} y y_{vij}.

4.1.6. Información computacional

El problema se ha modelado en el lenguaje **GNU MathProg**, que es un lenguaje de modelado diseñado para describir modelos lineales de programación matemática y se resuelve con el solver **GLPK**, que se ejecuta en el software **glpsol.exe** y que utiliza a la vez la interface **GUSEK** *Versión 0.2*.

El modelo ha sido evaluado en un computador con las siguientes características:

- Nombre del SO: Microsoft Windows 10 Home Single Language
- Fabricante del SO:Microsoft Corporation.
- Modelo del sistema:Modelo del sistema HP Laptop 15-dw1xxx.
- Tipo de sistema:x64-based PC.
- Procesador: Intel(R) Core(TM) i5-10210U CPU @ 1.60GHz, 2112 Mhz, 4 procesadores principales, 8 procesadores lógicos.
- Memoria física instalada (RAM) 8.00 GB.

4.1.6.1. Aplicación del modelo para el ruteo terrestre

```
/*DEPÓSITOS */
set Dc;        # Conjunto de depósitos de camiones.

#PUNTOS
set Pt;        # Conjunto de  puntos afectados con
                 carretera  en buen estado

/*VEHÍCULOS*/
set C;         # Conjunto de camiones.


/*NODOS*/
set T:= Pt union Dc; # Conjunto de nodos que forman la red entre
                       nodos de puntos afectados con carretera en
                       buen estado y depósitos de camiones.

/* DISTANCIAS */
param xcoord_T {i in T}; # x coordenada de puntos afectados con
```

```
                          carretera en buen estado.

param ycoord_T {j in T}; # y coordenada de puntos afectados con
                          carretera en buen estado.

param c{i in T, j in T:i!=j}:= sqrt((xcoord_T[i] - xcoord_T[j])^2
                              + (ycoord_T[i] - ycoord_T[j])^2);

/* PARÁMETROS */
param capc:= 150;  # Capacidad del depósito
param qc{v in C};  # Capacidad del camión
param velc:= 50;   # velocidad de los camiones
param ddc{i in Pt};# Entrega al cliente de acuerdo con la demanda

/* VARIABLES DE DECISIÓN */
var X{v in C, i in T, j in T: i != j}, binary;
#  Variable de ruteo par camiones

/* AUXILIARES */
var U1{i in T, j in C}, integer, >= 0;
# Entrega de carga antes de atender al cliente para carretera
transitable

/*FUNCIÓN OBJETIVO*/
minimize Tiempo:
sum{i in T, j in T, v in C: i!=j}  (c[i,j]/velc)*X[v,i,j] ;

s.t. R2 {j  in T, v in C}:
sum{i  in T: i != j} X[v, i, j]=sum{i in T: i != j} X[v,j,i];

s.t. R4{i  in  Pt}:
sum{j  in T, v  in C: i !=j}  X[v, i, j]=1 ;

s.t. R6{j  in  Pt}:
sum{i  in T, v  in C: i !=j}  X[v, i, j]=1  ;

s.t. R8{v  in C}:
sum{i  in T, j in Pt: i != j} X[v,i,j]*ddc[j] <=  qc[v];

s.t. R10:
sum{v in C, i  in T, j in Pt: i != j} X[v,i,j]*ddc[j] <=  capc;

s.t. R12{i in Pt, j in Pt, v in C: i!=j}:
U1[i,v]-U1[j,v]+ card(Pt)* X[v, i, j] <=  card(Pt)-1;
```

```
s.t. R14{v  in  C, i  in T}: U1[i,v] <= card(Pt);

s.t. R16{v in C, i in Dc}: U1[i,v] = 0 ;

solve;

printf "\n\n\n";
printf"   Tiempo de ruteo:
%s\n", sum{i in T, j in T, v in C: i!=j}  (c[i,j]/velc)*X[v,i,j];

printf "\n" ;
printf"   Transporte de i para j por medio de camiones: %s\n";
printf "\n" ;
printf {v in C, i in T, j in T: i != j and X[v,i,j] } "
%10s  %10s   %10s \n", i, j,v;
printf "\n\n\n" ;

data;

set Pt :='c1' 'c3' 'c5' 'c7' 'c9' 'c11' 'c13' 'c15' 'c17' 'c19'
'c21' /* 'c23' 'c25'  'c27' 'c29'  'c31' 'c33' 'c35' 'c37' 'c39'
'c41'*/; #;   # Conjunto de clientes

set Dc:= 'Dca' ;     # Conjunto de depósitos de camiones
set C:= 'k1' 'k2' ;  # Conjunto de vehículos

param:   xcoord_T   ycoord_T :=
# Coordenadas del depósito de camiones y puntos
afectados con carretera en buen estado

'Dca' 542698 9426330
'c1'  541731 9423472
'c3'  541726 9423805
'c5'  541315 9425980
'c7'  540289 9421817
'c9'  536579 9420045
'c11' 535712 9417720
'c13' 542440 9426962
'c15' 541163 9426549
'c17' 537035 9411431
'c19' 541802 9425051
'c21' 541823 9425629
'c23' 537273 9420509
```

```
'c25' 532591 9413999
'c27' 537089 9418374
'c29' 532756 9417178
'c31' 542736 9432030
'c33' 540865 9440855
'c35' 540929 9425045
'c37' 536241 9417094
'c39' 538414 9419951
'c41' 531623 9416210;

param  ddc :=  # Demanda de los puntos afectados con
carretera en buen estado
'c1'   9
'c3'   4
'c5'   6
'c7'   7
'c9'   2
'c11'  4
'c13'  6
'c15'  2
'c17'  3
'c19'  2
'c21'  6
'c23'  6
'c25'  4
'c27'  7
'c29'  8
'c31'  9
'c33'  5
'c35'  2
'c37'  6
'c39'  4
'c41'  2;

param qc :=   # Capacidad de los camiones
   'k1'  100
   'k2'  125 ;

end;
```

4.1.6.2. Aplicación del modelo para el ruteo Aéreo

La existencia de puntos afectados que no pueden ser atendidos por carretera, ha incentivado a realizar también un modelo de ruteo para ser atendidos por vía aérea; El modelo es básica mente el mismo, solo se ha modificado la terminología.

```
/*DEPÓSITOS */
set Dh;  # Conjunto de depósitos de helicópteros.

#PUNTOS
set Pd;  # Conjunto de  puntos con carretera dañada.

/*VEHÍCULOS*/
set H;   # Conjunto de helicópteros.

/*NODOS*/
set A:= Pd union Dh;  # Conjunto de nodos que forman la red entre
                        nodos de puntos afectados con carretera
                        dañada y depósitos de helicópteros

/* DISTANCIAS */
param xcoord_A {i in A}; # x coordinada de puntos afectados con
                           carretera dañada
param ycoord_A {j in A}; # y coordinada de puntos afectados con
                           carretera dañada
param cc{i in A, j in A:i!=j}:= sqrt((xcoord_A[i] - xcoord_A[j])^2
                                + (ycoord_A[i] - ycoord_A[j])^2);

/* PARÁMETROS */
param caph:= 100;    # Capacidad del depósito
param qh{v in H};  # Capacidad del helicóptero
param velh:= 260;    # velocidad promedio de los helicópteros
param ddh{i in Pd};  # Entrega al cliente de acuerdo con la demanda


/* VARIABLES DE DECISIÓN */
var Y{v in H, i in A, j in A: i != j}, binary;
#  Variable de ruteo para helicópteros

/* AUXILIARES */
var U2{i in A, j in H}, integer, >= 0;
# Entrega de carga antes de atender al cliente para carretera dañada
```

```
/* OBJETIVO*/
minimize Tiempo:
sum{i in A, j in A, v in H: i!=j}  (cc[i,j]/velh)*Y[v,i,j];

s.t. R3{j in A, v  in H}:
sum{i in A: i != j} Y[v, i, j]=sum{i in A: i != j} Y[v, j, i];

s.t. R5{i  in  Pd}:
sum{j  in A, v  in H: i !=j}  Y[v, i, j]=1  ;

s.t. R7{j  in  Pd}:
sum{i  in A, v  in H: i !=j}  Y[v, i, j]=1  ;

s.t. R9{v  in H}:
sum{i  in A, j in Pd: i != j} Y[v,i,j]*ddh[j] <=  qh[v];

s.t. R11:
sum{v in H, i  in A, j in Pd: i != j} Y[v,i,j]*ddh[j] <=  caph;

s.t. R13{i in  Pd, j in Pd, v in H: i!=j}:
U2[i,v]-U2[j,v]+ card(Pd)* Y[v, i, j] <=  card(Pd)-1;

s.t. R16{v  in  H, i  in A}:
U2[i,v] <= card(Pd);

s.t. R17{v in H, i in Dh}:
U2[i,v] = 0 ;

solve;

printf "\n\n\n";
printf"   Tiempo de ruteo: %s\n", sum{i in A, j in A, v in H: i!=j}
(cc[i,j]/velh)*Y[v,i,j];

printf "\n" ;
printf "\n" ;
printf" Transporte de i para j por medio de helicópteros: %s\n";
printf "\n" ;
printf {v in H, i in A, j in A: i != j and Y[v,i,j] } "
%10s  %10s   %10s \n", i, j,v;
printf "\n\n\n" ;
printf "\n" ;
data;
```

```
set Pd :='c2' 'c4'/* 'c6' 'c8'  'c10'  'c12' 'c14' 'c16' 'c18'
'c20' 'c22' 'c24' 'c26' 'c28' 'c30'  'c32' 'c34' 'c36'  'c38'
'c40'; # Conjunto de clientes

set Dh:= 'Dhe' ; # Conjunto localidades de las facilidades

set H:= 'k3' 'k4';  # Conjunto de vehículos urbanos

param:  xcoord_A    ycoord_A := # Coordenadas de clientes
'Dhe' 542550    9425499
'c2'  542073 9424493
'c4'  542034  9424732
'c6'  541921 9424621
'c8'  540701 9424727
'c10' 535893 9418014
'c12' 537461 9418575
'c14' 541953 9424368
'c16' 537208 9419504
'c18' 534048 9414120
'c20' 541354 9425470
'c22' 536870 9419244
'c24' 533508 9414217
'c26' 534715 9413304
'c28' 537019 9406974
'c30' 541196 9427992
'c32' 542422 9430392
'c34' 543819 9433859
'c36' 535171 9416937
'c38' 538831 9420329
'c40' 531620 9416290;

param   ddh :=   # Demanda de los puntos afectados con
carretera dañada
'c2'   3
'c4'   5
'c6'   8
'c8'   7
'c10'  6
'c12'  4
'c14'  9
'c16'  5
'c18'  9
'c20'  1
'c22'  5
```

```
'c24'  3
'c26'  5
'c28'  2
'c30'  5
'c32'  6
'c34'  7
'c36'  5
'c38'  4*/;

param  qh :=     # Capacidad de los helicóptero
 'k3'    55
 'k4'    55 ;

end;
```

4.1.7. Caso de estudio

Las coordenadas de depósitos, puntos afectados con carretera dañada y puntos con carretera transitables han sido obtenidas mediante una trasformación a partir de coordenadas geográficas obtenidas en *Google Earth*.

Los datos considerados constan de: 1 depósito de camiones, 1 depósito de helicópteros, 41 puntos afectados.

Se muestran a continuación las tablas (4.1), (4.2), (4.3) de las coordenadas cartesianas de la ubicación de los depósitos y puntos afectados.

Depósitos	X	Y
Dh	542519	9425345
Dc	542698	9426330

Tabla 4.1: Coordenadas cartesianas de los depósitos.

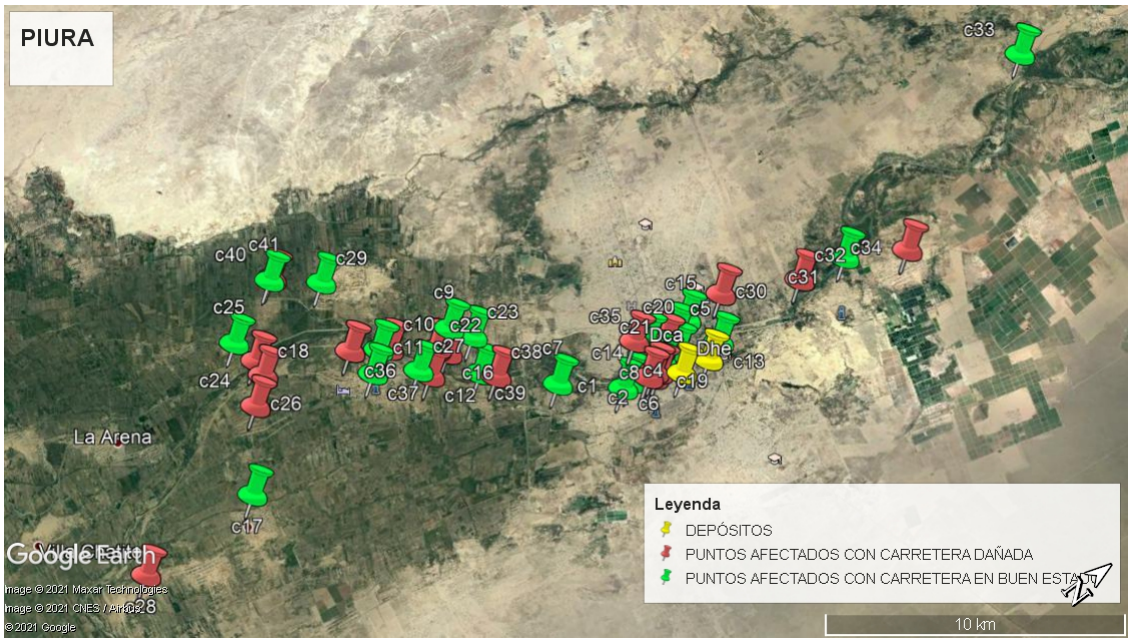


Figura 4.3: Ubicación de puntos afectados y depósitos en la ciudad de Piura

Clientes	X	Y	Clientes	X	Y
c1	541731	9423472	c3	541726	9423805
c5	541315	9425980	c7	540289	9421817
c9	536579	9420045	c11	535712	9417720
c13	542440	9426962	c15	541163	9426549
c17	537035	9411431	c19	541802	9425051
c21	541823	9425629	c23	537273	9420509
c25	532591	9413999	c27	537089	9418374
c29	532756	9417178	c31	542736	9432030
c33	540865	9440855	c35	540929	9425045
c37	536241	9417094	c39	538414	9419951
c41	531623	9416210			

Tabla 4.2: Coordenadas cartesianas de los puntos afectados con carretera transitable.

Clientes	X	Y	Clientes	X	Y
c2	542073	9424493	c4	542034	9424732
c6	541921	9424621	c8	540701	9424727
c10	535893	9418014	c12	537461	9418575
c14	541953	9424368	c16	537208	9419504
c18	534048	9414120	c20	541354	9425470
c22	536870	9419244	c24	533508	9414217
c26	534715	9413304	c28	537019	9406974
c30	541196	9427992	c32	542422	9430392
c34	543819	9433859	c36	535171	9416937
c38	538831	9420329	c40	531620	9416290

Tabla 4.3: Coordenadas cartesianas de los puntos afectados con carretera dañada.

4.2. Análisis de resultados

Se muestra a continuación una tabla comparativa de los resultados obtenidos al evaluar diferentes situaciones en el modelo propuesto.

Depósitos	Clientes	Optimo	Función objetivo	Tiempo
1	2	si	121.116454571151	0.0 secs
1	4	si	218.958300972506	0.0 secs
1	6	si	457.551720751719	0.2 secs
1	8	si	481.293911306633	0.6 secs
1	10	si	706.732959290292	36.6 secs
1	12	no	710.371283081989	3600.1 secs
1	14	si	796.819293231555	1274.3 secs
1	16	no	1049.28072401929	3600.3 secs
1	18	no	1434.25090874539	3600.3 secs
1	20	no	1446.15977039676	3600.2 secs

Tabla 4.4: Comparación de resultados para ruteo terrestre.

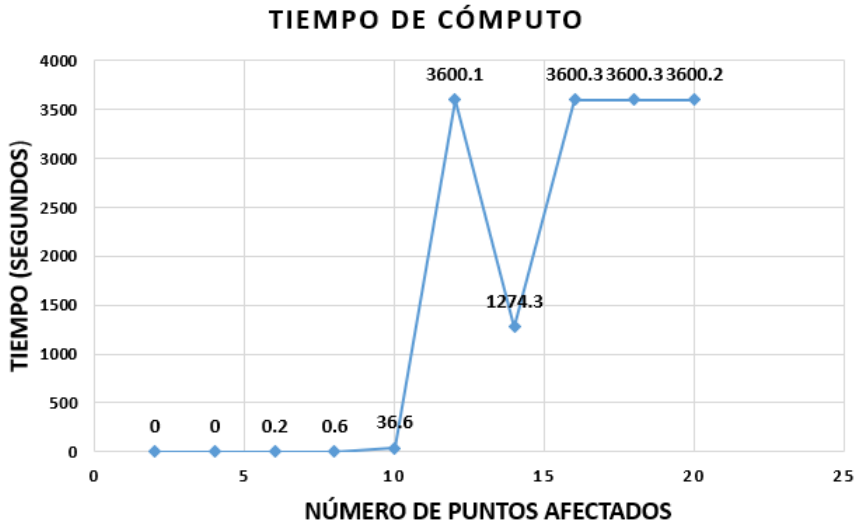


Figura 4.4: Tiempo de computo para modelo de ruteo terrestre

Depósitos	Clientes	Optimo	Función objetivo	Tiempo
1	2	si	8.76897949980401	0.0 secs
1	4	si	17.6676235137897	0.0 secs
1	6	si	79.6562925675801	0.1 secs
1	8	si	79.7117196021378	3.4 secs
1	10	si	112.163876425294	21.5 secs
1	12	no	116.756303170662	3600.2 secs
1	14	no	169.270040547867	3600.2 secs
1	16	no	213.658596199221	3600.3 secs
1	18	no	237.448569728781	3600.3 secs
1	20	no	261.18361562098	3600.2 secs

Tabla 4.5: Comparación de resultados para ruteo aéreo.

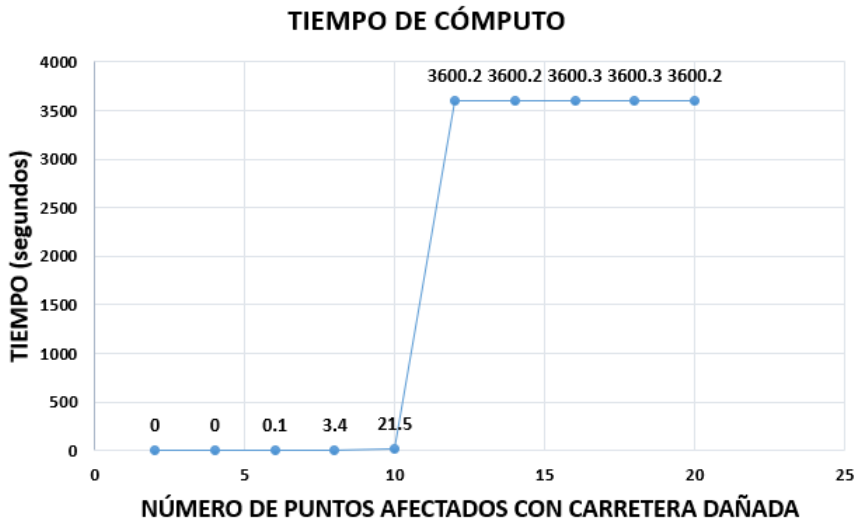


Figura 4.5: Tiempo de computo para modelo de ruteo aéreo

Las grafos de ruteo de vehículos se han elaborado en el software científico *Wolfram Mathematica 11.3* en base a los datos obtenidos del modelo de programación lineal elaborado en *GLPK*, se comparte a continuación el miniprograma

```
Dca = {542698, 9426330}; Dhe = {542519, 9425345};
c1 = {541731, 9423472};  c2 = {542073, 9424493};
c3 = {541726, 9423805};  c4 = {542034, 9424732};
c5 = {541315, 9425980};  c6 = {541921, 9424621};
c7 = {540289, 9421817};  c8 = {540701, 9424727};
c9 = {536579, 9420045};  c10 = {535893, 9418014};
c11 = {535712, 9417720}; c12 = {537461, 9418575};
c13 = {542440, 9426962}; c14 = {541953, 9424368};
c15 = {541163, 9426549}; c16 = {537208, 9419504};
c17 = {537035, 9411431}; c18 = {534048, 9414120};
c19 = {541802, 9425051}; c20 = {541354, 9425470};
c21 = {541823, 9425629}; c22 = {536870, 9419244};
c23 = {537273, 9420509}; c24 = {533508, 9414217};
c25 = {532591, 9413999}; c26 = {534715, 9413304};
c27 = {537089, 9418374}; c28 = {537019, 9406974};
c29 = {532756, 9417178}; c30 = {541196, 9427992};
c31 = {542736, 9432030}; c32 = {542422, 9430392};
c33 = {540865, 9440855}; c34 = {543819, 9433859};
c35 = {540929, 9425045}; c36 = {535171, 9416937};
c37 = {536241, 9417094}; c38 = {538831, 9420329};
c39 = {538414, 9419951}; c40 = {531620, 9416290};
c41 = {531623, 9416210};

Graphrv[l_, m_, x_, y_] :=
Graphics[{Hue[0.6], PointSize[0.025], Thick,
Table[{Circle[l[[i]], 100], Hue[y], Disk[l[[i]], 100]},
{i, 1, Length[l] - 1}], Hue[x], Thick,
Table[Arrow[{l[[i]], l[[i + 1]]}], {i, 1, Length[l] - 1}],
Hue[1/4, 1, 0],
Table[Text[Style[ m[[i]], 10], l[[i]] + {100, 100}],
{i, 1, Length[l]}]}]


l1 = {Dca, c13, c15, c5, c9, c11, c17, c7, c1, c3, c19, Dca};
m1 = {"Dca", "c13", "c15", "c5", "c9", "c11", "c17", "c7",
"c1", "c3", "c19", "Dca"};
g1 = Graphrv[l1, m1, 0.7, 0.5]
```

```
l2 = {Dhe, c4, c6, c2, c14, c12, c18, c10, c16, c8, c20, Dhe};
m2 = {"Dhe", "c4", "c6", "c2", "c14", "c12", "c18", "c10", "c16",
   "c8", "c20", "Dhe"};
g2 = Graphrv[l2, m2, 0, 0.5]

Show[g1, g2]
```

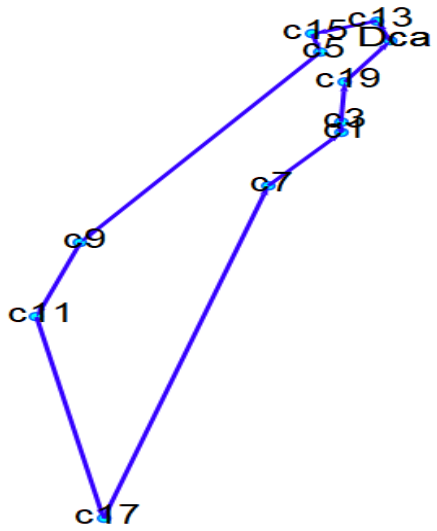


Figura 4.6: Ruteo de camiones para 10 puntos afectados

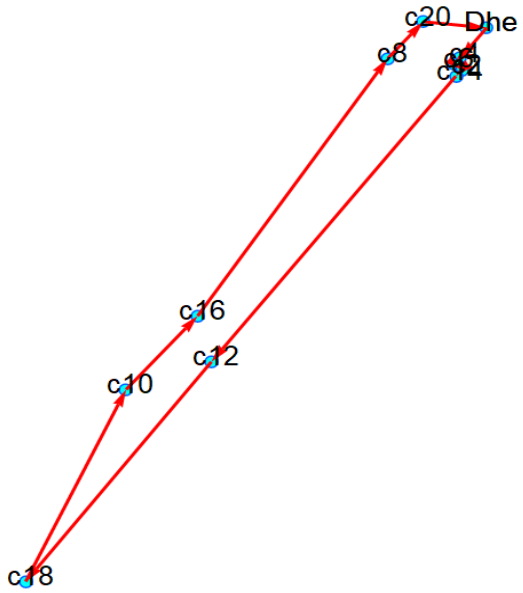


Figura 4.7: Ruteo de helicópteros para 10 puntos afectados

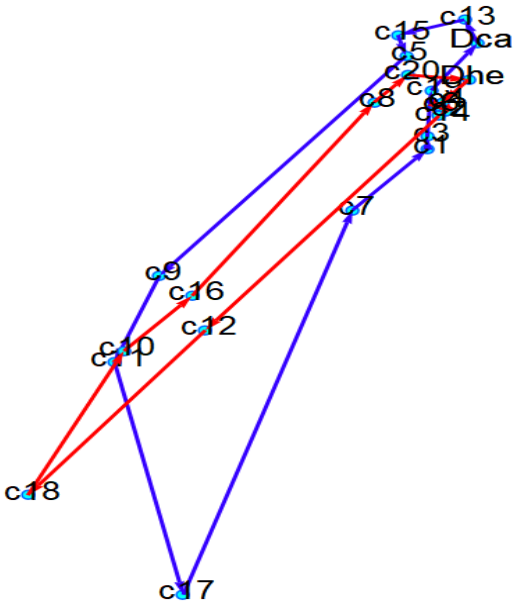


Figura 4.8: Ruteo de flotas mixtas para 10 puntos afectados con carretera en buen estado y 10 puntos afectados con carretera dañada

CAPÍTULO V

APLICACIÓN DE METAHEURÍSTICAS AL CASO DE ESTUDIO

Los resultados obtenidos al evaluar el modelo de ruteo de vehículos propuesto en el capítulo anterior, muestran un alto consumo del tiempo de ejecución y no consigue encontrar una solución óptima para más de 14 puntos afectados, dada la necesidad y la urgencia de hallar una solución en un tiempo breve, se siente la necesidad de implementar metaheurísticas para reducir el tiempo de cómputo y obtener una buena solución. Se analizan dos metaheurística: Algoritmos Inspirados en Colonias de Hormiga (**ACO**) y Algoritmos Genéticos (**AG**).

El propósito de esta investigación es determinar cuál de las metaheurísticas devuelve mejores resultados en un tiempo y número de iteraciones limitadas. La metaheurísticas han sido implementadas en el lenguaje de programación *Python*, tanto los **AG** como **ACO** han sido limitados para 100 iteraciones y 100 individuos.

5.1. Aplicación de la metaheurística basados en colonias de hormigas al caso de estudio.

5.1.1. Descripción de la metaheurística ACO

Generar ruta

Para generar una ruta con el algoritmo ACO, se definen dos variables de influencia: alfa (coeficiente de influencia sobre la evaporación de feromonas) y beta (coeficiente de influencia sobre la visibilidad). Los valores que se van a asignar son 2 y 5 respectivamente porque se ha comprobado que con ellos existe más posibilidad de obtener buenas soluciones.

El proceso comienza con la entrada al bucle permitido por la variable "parada" con valor true (verdadero). La primera instrucción dentro del bucle determina el grupo de nodos destinos posibles desde un nodo origen o nodo actual. En esta instrucción se invoca a una función denominada "obtener Nodos Destinos" y los parámetros enviados son la variable de ruteo, el conjunto de clientes, el nodo actual indicado por "$nodo_{actual}$" y una variable de atención de clientes.

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, sino se asigna el valor de false (falso) a la variable "parada" cancelando el funcionamiento del bucle y terminando con el proceso.

Partiendo del caso de que la condición mencionada anteriormente resulte ser verdadera, la siguiente instrucción permite crear 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 y, se calcula la visibilidad entre los nodos mencionados mediante el inverso multiplicativo de la distancia entre estos dos.

$$temp_k = f_{ij}^{\alpha} \times v_{ij}^{\beta}$$

$$\forall k \in [0, long(nodosdestinos)] \wedge i = nodoactual \wedge j = nodosdestinos$$

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.

$$probabilidad_k = \frac{temp_k}{\sum_{i=0}^{k-1} temp_i}$$

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.

$$temp_k \leftarrow probabilidad_k$$

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

$$\text{Región}_k = [suma, suma + temp_i]$$

Partiendo de suma=0, la primera región es:

$$suma = 0 \rightarrow \text{Región}_0 = [0, temp_0]$$

Luego, las siguientes regiones se obtienen de la siguiente forma:

$$suma = suma_{anterior} + temp_0 \rightarrow \text{Región}_1 = [suma, suma + temp_1]$$

$$suma = suma_{anterior} + temp_1 \rightarrow \text{Región}_2 = [suma, suma + temp_2]$$

$$\vdots \quad = \quad \vdots \quad + \quad \vdots \quad \rightarrow \quad \vdots \quad = \quad \vdots$$

$$suma = suma_{anterior} + temp_{(i-1)} \rightarrow \text{Región}_k = [suma, suma + temp_i]$$

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

$$\text{número aleatorio} \leq suma + temp_i$$

Si la condición es falsa, se sigue 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 construcción de una tabla de entregas. La tabla estará conformada por tramos los cuales son caminos de un nodo a otro y por medio del cual el vehículo recorre con una cierta cantidad de carga o pedidos denominado entrega. 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 las entregas no superan la capacidad del vehículo con el que se está evaluando y también si la entrega total no supera la capacidad total o restante de la facilidad asignada para esta ruta. Si el resultado es verdadero, el nodo destino es agregado a la ruta, el camino recorrido por la hormiga (vehículo) entre el nodo actual y el nodo destino es indicado por el valor 1 en la variable de ruteo, el nodo siguiente se convierte en nodo actual, su atención es indicada con el valor 1 en la variable de atención de clientes. Luego, 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, se eliminan los tramos de la tabla de entregas que se conectan a este nodo y se verifica si la tabla tiene por lo menos un tramo, si este fuera el caso, se añade el tramo de retorno. Asimismo, el valor de falso es asignado a la variable "parada" para cancelar el bucle principal.

Finalmente, se agrega a la ruta el nodo inicial el cual será la facilidad y se verifica la cantidad de nodos de la ruta para devolver un resultado final. Si la ruta tiene una cantidad de nodos menor que 3, entonces la ruta es inválida dando como resultado un vacío; o sino, se retorna una lista que adjunta la ruta y la cantidad total de pedidos.

Generar ruteo

La variable importante del algoritmo es mejor_ruteo-_global pues devolverá una buena solución calculada por medio de ACO mejorada. Se declaran e inicializan las variables "feromonas" e "iterador". A partir de ahora, se inicia el bucle principal y la decisión de seguir o cancelar depende de la variable "iterador" comenzando con el valor 1 hasta llegar al número máximo de iteraciones.

Dentro del bucle o por cada iteración, se lanza un cierto número de hormigas el cual es un parámetro de entrada ya definido por el usuario. Por cada lanzamiento de una hormiga, se inicializan las variables "Ca" (atención de clientes), "X" (variable de ruteo) y "W" (variable de cargamento que se almacena en una facilidad). Además, se declara en inicializan las variables "facilidad" (facilidad elegida), "facilidades_descartadas", "band",y "vehículos_disponibles" (copia de la lista de vehículos). La primera parte comprende la generación de ruteos aplicando ACO.

El bucle interno "Mientras" es la primera parte del algoritmo y es en ella en donde se generará la solución inicial del ruteo empleando ACO. 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 haya sido descartada. 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.

Partiendo de que existe la facilidad elegida, se asigna el primer vehículo disponible y se genera la ruta empleando el algoritmo **generarRuta**. Luego, se verifica la inexistencia de resultados y la disponibilidad de un vehículo. Si la condición es verdadera, se genera otra ruta con otro vehículo hasta que haya resultados o bien hasta que se termine de evaluar con todos los vehículos disponibles.

Posteriormente, se verifica la existencia de resultados. Si estos existen, los pedidos se añaden a la lista "W" y se verifica la capacidad restante de la facilidad; si esta se llena, se anula para evitar utilizarla en la generación de otra ruta; caso contrario, se seguirá utilizando esta facilidad hasta que no

tenga espacio para almacenar otra carga. La ruta se agrega al ruteo con ACO, se limpia la lista de facilidades descartadas, se reinicia la variable de ruta y el vehículo utilizado deja de estar disponible. Después, se verifica la lista de clientes atendidos. Si todos ya fueron atendidos, se termina la búsqueda de otra ruta; sino, se procede a calcular otra aplicando los procedimientos anteriormente explicados.

En el párrafo anterior se explicó el procedimiento que sigue cuando se obtienen resultados. Para el caso de la inexistencia de estos, la facilidad se descarta y se calcula otra nueva ruta. Posteriormente, si existe una solución o ruteo, se ejecutan las instrucciones para encontrar el mejor ruteo local (el mejor ruteo de entre todos aquellos generados por las hormigas). Cuando no existe un mejor ruteo local al inicio, se asigna el ruteo de la primera hormiga, de otro modo se comparan el tiempo objetivo de la ruta de la hormiga y el mejor ruteo local anteriormente asignado. El que tiene un tiempo menor, es considerado el nuevo mejor ruteo local o, en otro caso, seguir siendo el mejor ruteo local.

La segunda parte es el mejoramiento de la solución inicial obtenido por ACO. Si existe el ruteo inicial, se inicializan las variables necesarias para esta parte como "X_{aux}" que es la variable de ruteo alternativa a "X" y será utilizada para la solución con búsqueda local de 2-opt. Teniendo un ruteo inicial (ruteo con ACO) y un tiempo inicial objetivo (tiempo del ruteo con ACO) se inicia el ciclo del bucle Mientras. Para un ciclo de este bucle, se obtiene la ruta mejorada, un tiempo final y se evalúa la diferencia entre el tiempo inicial y final. Si la diferencia es mayor que 0, el ruteo mejorado se convierte en inicial, el tiempo final se convierte en tiempo inicial y se ejecuta el siguiente ciclo hasta que la diferencia de tiempos objetivos sea igual a 0 y termine la parte de mejoramiento de la solución.

Después, se comparan los tiempos del ruteo mejorado y del ruteo inicial (ACO). Si el primero es menor, se convierte en el mejor ruteo global (el mejor de entre todos los ruteos generado en cada iteración) y se actualiza la matriz de feromonas teniendo en cuenta su variable de ruteo. De otro modo, se realiza el mismo procedimiento, pero con el ruteo inicial y se actualiza la matriz de feromonas de la misma forma que el anterior. Después, se continúa con la siguiente iteración. Finalmente, se retorna el mejor ruteo global después de haber terminado la ejecución del bloque de iteraciones.

Búsqueda local 2-opt

El algoritmo de búsqueda local de dos opciones o 2-opt permite mejorar una ruta eliminando los cruces sobre sí misma. Los parámetros que recibe son: el ruteo generado con algoritmo inspirado en colonias de hormigas, la matriz global de distancias, la lista de vehículos y la variable de ruteo. El ruteo con búsqueda local se inicializa con la copia del ruteo con ACO.

El mejoramiento del ruteo se procede ruta por ruta. Por cada ciclo del bucle principal, se va a emplear las variables de "min_{cambio}" (cambio mínimo), "min_i" y "min_j". Sin embargo, antes de proceder a reordenar la ruta, se localiza el vehículo asignado para esta ruta por medio del índice de la ruta. El vehículo se localizará en la variable de ruteo.

El primer paso para mejorar una ruta es tomar dos nodos $(ruta_i, ruta_j)$ no consecutivos y calcular los costos de la siguiente manera:

$$costo_{actual} = distancias_{(ruta_i, ruta_{(}i+1))} + distancias_{(ruta_j, ruta_{(}j+1))}$$

$$costo_{nuevo} = distancias_{(ruta_i, ruta_j)} + distancias_{(ruta_{(}i+1), ruta_{(}j+1))}$$

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

$$cambio = costo_{nuevo} - costo_{actual}$$

Después se compara si el cambio es menor que el cambio mínimo. Si el resultado de la condición es verdadero, 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, 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 valor de 1 en los caminos que forman parte de la ruta reordenada teniendo en cuenta el vehículo que recorre la ruta. Luego, se produce el cambio del índice 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 el ruteo mejorado denominado "$ruteo_{bl}$".

5.1.2. Resultados

Los resultados obtenidos para el modelo de ruteo de vehículos terrestre y aéreo aplicando algoritmos inspirados en colonias de hormigas son:

Depósitos	**Clientes**	**Optimo**	**Función objetivo**	**Tiempo**
1	4	si	218.9583	1.3328 secs
1	6	si	457.5517	2.9417 secs
1	8	si	481.2939	3.9348 secs
1	10	si	706.733	5.3423 secs
1	12	si	710.3713	6.4947 secs
1	14	-	797.4248	8.48 secs
1	16	-	1051.4637	18.8295 secs
1	18	-	1403.0042	13.4699 secs
1	20	-	1419.5279	15.8466 secs

Tabla 5.1: Obtención de resultados en **ACO** para ruteo terrestre.

Depósitos	Clientes	Optimo	Función objetivo	Tiempo
1	4	si	17.6676	1.26 secs
1	6	si	79.6563	2.3148 secs
1	8	si	79.7117	3.81 secs
1	10	si	112.1639	5.1369 secs
1	12	si	114.9851	6.7401 secs
1	14	-	168.5043	8.8048 secs
1	16	-	208.1165	11.1003 secs
1	18	-	236.1925	13.5916 secs
1	20	-	260.6064	15.2493 secs

Tabla 5.2: Obtención de resultados en **ACO** para ruteo aéreo.

El modelo ha sido evaluado inicialmente para cuatro puntos afectados, estas van incrementando de dos en dos. A comparación de los resultados obtenidos en la evaluación del modelo en **GLPK**, los **ACO** muestra una considerable reducción del tiempo de cómputo, para el ruteo terrestre encuentra el óptimo hasta 12 puntos afectados, mientras que para el ruteo aéreo encuentra el óptimo hasta 10 puntos afectados.

Los resultados por **ACO** muestran hasta un 2.2 % y 2.6 % de variación de los resultados de la función objetivo obtenidos en **GLPK**.

5.2. Aplicación de la metaheurística basados en algoritmos genéticos al caso de estudio.

5.2.1. Descripción del la metaheurística AG

Generar ruteo Este procedimiento ha sido desarrollado basándose en la lógica general para obtener un ruteo como una buena solución. Es la función principal del programa.

El primer paso es obtener una población utilizando el algoritmo de generarPoblación el cual permitirá retornar la población, la lista fitness y la lista general de vehículos utilizados. Posteriormente, se da inicio a la ejecución de un bloque iterativo cuyo número de iteraciones es un parámetro de entrada y será definido por el usuario, así como el número de individuos de todas las poblaciones que se irán generando durante la ejecución de esta función.

En cada iteración, se calcula la descendencia hasta que el número de descendientes sea igual al número de individuos definido al inicio. Para generar cada par de descendientes (caso ideal), se emplean los algoritmos de selección, cruzamiento, mutación y evaluación. El par de descendientes generados por los tres primeros algoritmo pasa a evaluación y puede ocurrir lo siguiente: un descendiente puede ser aceptado y pasar a la población de descendencia; ambos son aceptados y pasan a la población de descendencia; o que ninguno sea aceptado. Cuando el número de descendientes sea igual o quizá mayor pero cercano al número de individuos iniciales, se detiene el procedimiento de encontrar descendientes y lo siguiente es generar la nueva población aplicando el método de ranking explicada en su apartado respectivo. Con la nueva población o nueva generación obtenida, se ejecuta la siguiente iteración.

La solución inicial será aquel individuo con mejor fitness (el menor tiempo) de la población que resulte tras terminar de ejecutarse la última iteración. Esta solución pasa al siguiente procedimiento que es el mejoramiento aplicando el algoritmo de búsqueda local 2-opt. El mejoramiento es un proceso iterativo y finalizará cuando los tiempos final e inicial sean iguales.

Luego, se comparan los tiempos inicial (el de la solución inicial obtenida con AG) y el final (el último calculado en el mejoramiento de la solución). Si el tiempo final es el menor, el ruteo mejorado se convierte en la buena solución esperada, caso contrario, será la solución inicial.

Finalmente, se retornará el ruteo óptimo (en la mayoría de caso no será óptimo) el cual adjunta a la solución final, los vehículos utilizados y el tiempo objetivo.

Generar población El número de individuos de la población es un parámetro de entrada el cual será asignado por el usuario, pero si no recibe valor alguno, entonces se determina mediante la siguiente expresión matemática:

$$nroInd = aleatorio(nro_{clientes} + nro_{facilidades}, 2 * (nro_{clientes} + nro_{facilidades}))$$

El número de individuos se calcula aleatoriamente teniendo en cuenta un intervalo desde $tam_{clientes} + tam_{facilidades}$ hasta $2 * (tam_{clientes} + tam_{facilidades})$.

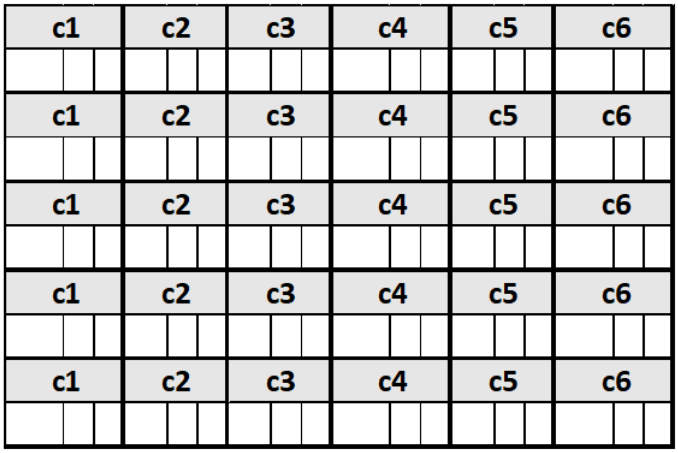


Figura 5.1: Población conformada por cinco individuos.

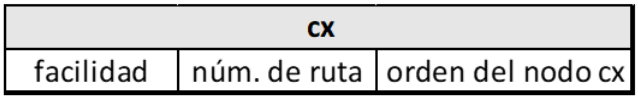


Figura 5.2: Estructura de un gen.

En la figura (5.1) se observa una población con cinco individuos y en la figura (5.2) se observa la estructura de un gen "cx" de cada individuo.

El siguiente paso es almacenar datos por cada gen o cliente. Primero, para construir un individuo, se seleccionan todos los clientes, los cuales pasan a ser los genes de este. Luego, se procede a calcular las rutas. Para generar una ruta, primero se elige una facilidad aleatoriamente y también una cantidad aleatoria de clientes. Posteriormente, se elige aleatoriamente a cada cliente y se ubica en el individuo con el objetivo de ingresar los datos de la facilidad elegida, el número de la ruta que se está calculando y el número de orden de elección del cliente. Por ejemplo, si se ha elegido una facilidad "$f2$", se está calculando la ruta 1 y se acaba de elegir segundo el cliente "$c5$", entonces la información del gen "$c5$" en el individuo que se está construyendo debe

tener la estructura $c5 = (f2, 1, 2)$, la misma estructura tal como se observa en la figura (5.2) solo que en este caso "cx" es "$c5$". El proceso de hallar una ruta se realizará hasta culminar con la cantidad de clientes calculada aleatoriamente y luego se volverá a calcular este número para hallar otra ruta. Este procedimiento termina cuando todos los clientes ya han sido atendidos o cuando todos los genes ya tienen datos asignados. Cuando el individuo ya está construido, se evalúa para poder verificar si cumple con las restricciones. Si el caso es afirmativo, el individuo pasa definitivamente a la población, su fitness pasa a la lista de fitness, la lista de vehículos utilizados es agregada a la lista general de vehículos utilizados y se procede a construir otro individuo hasta completar la población con la cantidad de individuos asignada. Si el caso es negativo, el individuo es desechado y se construye un reemplazo aplicando los procedimientos explicados anteriormente y someterlo a evaluación. Finalmente, se retornará una lista que adjunta a la población, la lista fitness y la lista general de vehículos utilizados.

Evaluación La evaluación de un individuo es importante para verificar si este cumple con las restricciones planteadas en el modelo matemático las cuales han sido adaptadas para este ruteo aplicando algoritmo genético.

El primer paso es decodificar el individuo, es decir, transformarlo en un ruteo. El siguiente paso es evaluar cada ruta de la posible solución. Se verifica la disponibilidad de vehículos y se procede a construir una tabla de entregas así como la acumulación de distancia acumulada o recorrida y la entrega total. Posteriormente, se presenta la condición que determinará la supervivencia del individuo. Esta consiste en verificar si las entregas no superan la capacidad del vehículo con el que se está evaluando y también si la entrega total no supera la capacidad total o restante de la facilidad asignada para esta ruta. Si la respuesta es afirmativa, se activan con 1 los caminos que conforman la ruta en la variable de ruteo "X", se agrega el vehículo seleccionado a la lista de vehículos utilizados en la ruta, deja de estar disponible y se da por aceptada la ruta mas no el ruteo. Si la respuesta es negativa, se evalúa con otro vehículo siempre y cuando haya al menos uno disponible; y si después de evaluar con todos los vehículos disponibles la respuesta es negativa, la ruta no es aceptada y en consecuente, el ruteo tampoco. En el caso de no contar con vehículos disponibles para evaluar una ruta también convierte al ruteo en inválido. Si todas las ruta son aceptadas, el ruteo también se considerará aceptado.

Para un ruteo aceptado, se calcula el fitness que no es mas que el opuesto o inverso aditivo del valor obtenido de la función objetivo, es decir, el fitness será un número negativo. Se utilizarán fitness negativos con el objetivo de encontrar la solución con un mejor fitness el cuál será un valor cercano a cero de entre todos los fitness calculados, es decir, el mayor. Luego, al obtener el valor absoluto será el mínimo fitness encontrado.

Finalmente, la función, para el caso del ruteo aceptado, retornará el fitness y la lista de vehículos utilizados, caso contrario, no retornará nada dando a entender que el individuo no ha sido aceptado.

Selección: La selección de individuos padres se realiza por medio del método de torneo. Consiste en asignar un valor para la variable k el cual indica la cantidad de competidores, siendo 4 el número elegido para este algoritmo. De los 4 individuos elegidos aleatoriamente, el que tiene el mayor fitness será elegido como padre. El valor de k considerado es 4 por lo cual se permite como mínimo 5 individuos ya que el segundo padre debe ser elegido de entre 4 individuos restantes. El primer padre seleccionado es asignado a la lista de padres y deja de ser parte de la población inicial para evitar otra posible participación en la selección del segundo padre. La cantidad de padres está indicada por el valor asignado a la variable p el cual es 1, generando a los (padre 0 y padre 1). Finalmente, la función de selección retorna los dos padres seleccionados para luego aplicarse el cruzamiento de ambos.

Cruzamiento El cruzamiento de padres se realiza por medio del método de cruzamiento parcialmente mapeado. El primer paso para poder aplicar este método es codificar la información de los genes de los padres en números. La estructura de un gen es la siguiente:

$$\text{Cliente} = (\text{facilidad,número de ruta,orden de visita })$$

Donde el nombre del gen es el cliente. Por ejemplo, un gen puede tener la siguiente información: $c1 = (\text{“}f2\text{”}, 3, 2)$ lo que significa que un vehículo de la ruta 3, que ha salido de la facilidad $f2$, es el segundo nodo de la ruta, el cual es el gen “$c1$”. Posteriormente, se utilizan 2 variables ind (ver pseudocódigo) las cuales representaran a los dos padres pero con genes numéricos. Por tanto, los valores del gen “$c1$” se transforman en un dato numérico de la forma $c1 = 232$.

c1			c2			c3			c4			c5			c6		
f1	2	3	f2	1	1	f2	1	3	f1	2	2	f1	2	1	f2	1	2

Figura 5.3: Individuo padre original.

c1			c2			c3			c4			c5			c6		
1	2	3	2	1	1	2	1	3	1	2	2	1	2	1	2	1	2

Figura 5.4: Individuo con genes numéricos del padre original.

La figura (5.3) representa a un individuo padre con los valores originales en cada uno de sus genes y la figura (5.4) representa a un individuo cuyos genes tienen valores numéricos ya transformados tomando los valores de los

genes del padre original. Esto se realiza con el objetivo de facilitar el proceso del cruzamiento de los padres. Una vez que ya se tienen a los dos individuos generados, se procede a obtener segmentos por cada uno de ellos. Para obtener los segmentos, se necesita saber cuales son los puntos de corte o genes extremos de cada segmento. De forma aleatoria, se obtienen dos genes extremos de un individuo de los cuales luego se obtiene sus indices de ubicación y se los ordena ascendentemente. Esto es útil para ubicar los segmentos y tratarlos de manera independiente.

0	1	2	3	4	5
c1	**c2**	**c3**	**c4**	**c5**	**c6**
123	211	213	122	121	212

Figura 5.5: Individuo padre 1 con un segmento en color rojo.

0	1	2	3	4	5
c1	**c2**	**c3**	**c4**	**c5**	**c6**
212	214	211	321	213	322

Figura 5.6: Individuo padre 2 con un segmento en color rojo.

Para obtener los segmentos de ambos individuos, se calculan los genes extremos. Tomando cualquiera de los individuos, aleatoriamente se tienen los extremos y en efecto, se generan los segmentos. En la figura (5.5) y figura (5.6), se observan los segmentos en color rojo los cuales fueron obtenidos a partir del la elección aleatoria de los extremos, en este caso los genes 3 y 5 con sus índices de ubicación 2 y 4 respectivamente. Los índices son necesarios para que el programa pueda localizar los segmentos en los individuos.

El siguiente paso es colocar los valores de los genes únicos en un segmento, es decir, aparecen en un segmento pero no en el otro. Los conjuntos que almacenan estos valores se llamarán i y j. En el conjunto i se almacenarán los valores de los genes del segmento 1 (individuo padre 1) que no aparecen en el segmento 2 (individuo padre 2).

$$i = \{122, 121\}$$

Y para el caso del conjunto j, se almacenan los valores de los genes del segmento 2 que no aparecen en el segmento 1.

$$j = \{211, 321\}$$

Los valores que se han utilizado como ejemplo han sido tomados de la figura (5.6) y figura (5.7). El valor que se repite en ambos segmentos es 213. Después,

se procede a encontrar a los descendientes o hijos los cuales, inicialmente, serán copias de cada padre, es decir, hijo 1 será la copia del padre 2 e hijo 2 será la copia del padre 1.

Para el hijo 1, copiar los valores del segmento 1 teniendo en cuenta los índices extremos.

c1	c2	c3	c4	c5	c6
212	214	213	122	121	322

Figura 5.7: El hijo 1 con los cambios aplicados.

En la figura (5.7), el hijo 1 tiene los valores del padre 2 y del padre 1 (en color azul en referencia al segmento 1). Luego, se copian los valores del conjunto j teniendo en cuenta el recorrido por el conjunto i.

$$i = \{122, 121\} \rightarrow indices_{(individuo2)} = \emptyset$$

Los valores del conjunto i no se encuentran en el individuo 2, por tanto, la figura (5.7) representa el primer descendiente obtenido.

Para el hijo 2, copiar los valores del segmento 2 teniendo en cuenta los índices extremos.

c1	c2	c3	c4	c5	c6
123	211	211	321	213	212

Figura 5.8: El hijo 2 con los primeros cambios aplicados.

En la figura (5.8), el hijo 2 tiene los valores del padre 1 y del padre 2 (en color azul en referencia al segmento 2). Luego, se copian los valores del conjunto i teniendo en cuenta el recorrido por el conjunto j.

$$j = \{211, 321\} \rightarrow indices_{(individuo1)} = \{1\}$$

Se ha recorrido el conjunto j buscando cada uno de sus valores en el individuo 1. En ese recorrido, solo se ha encontrado el valor 211 en el índice 1 del individuo 1 y utilizando dicho indice ubicamos el gen en el hijo 2 el cual es $c2$ y se reemplaza su valor por el del conjunto i ubicado en el mismo orden de posición que el 211 en el conjunto i, es decir, el 122.

En la figura (5.9), se observa el resultado final después de realizar los cambios, por tanto, se acaba de obtener el hijo 2. Todos estos cambios se

c1	c2	c3	c4	c5	c6
123	122	211	321	213	212

Figura 5.9: El hijo 2 con los últimos cambios realizados.

realizan para evitar valores repetidos en un individuo descendiente como el caso del valor 211 en el hijo 2 en la figura (5.8). Los resultados que la función de cruzamiento son los descendientes obtenidos los cuales se muestran en la figura (5.10) y (5.11) con los valores decodificados.

c1			c2			c3			c4			c5			c6		
f2	1	2	f2	1	4	f2	1	3	f1	2	2	f1	2	1	f3	2	2

Figura 5.10: Individuo descendiente 1.

c1			c2			c3			c4			c5			c6		
f1	2	3	f1	2	2	f2	1	1	f3	2	1	f2	1	3	f2	1	2

Figura 5.11: Individuo descendiente 2.

Mutación Para el proceso de mutación, se utiliza el método swap el cual es simplemente un intercambio de información entre 2 genes seleccionados aleatoriamente.

c1			c2			c3			c4			c5			c6		
f1	2	3	f1	2	2	f2	1	1	f3	2	1	f2	1	3	f2	1	2

Figura 5.12: Un individuo hijo antes de ser mutado.

c1			c2			c3			c4			c5			c6		
f1	2	3	f3	2	1	f2	1	1	f1	2	2	f2	1	3	f2	1	2

Figura 5.13: El individuo hijo mutado.

En la figura (5.12) se observa el descendiente obtenido después de efectuar el cruzamiento y se realiza la mutación de este dando como resultado lo que se observa en la figura (5.13).

Generar nueva población

Para generar la nueva población, se aplicó el método de selección por ranking. Consiste en seleccionar un individuo que tenga el mejor fitness de la población antigua o predecesora y otro individuo de la descendencia de la misma forma. Luego, se realiza una comparación entre ambos fitness: Si el fitness del individuo predecesor es mayor o igual al fitness del descendiente, entonces el primero pasa a la lista de la nueva población o nueva generación eliminando al otro individuo de la población en donde se encuentra. De otro modo, el descendiente pasa a la nueva generación eliminando al predecesor de la misma forma. Esto se realiza iterativamente hasta que el número de los individuos sobrevivientes sea igual al número de individuos determinado al inicio del programa. La función retornará como resultado la nueva población encontrada.

Búsqueda local 2-opt

El algoritmo de búsqueda local de dos opciones o 2-opt permite mejorar una ruta eliminando los cruces sobre sí misma. Los parámetros que recibe son el ruteo generado con algoritmo génetico (AG), la matriz global de distancias, la lista de vehículos y la variable de ruteo. El ruteo con búsqueda local se inicializa con la copia del ruteo con AG.

El mejoramiento del ruteo se procede ruta por ruta. Por cada ciclo del bucle principal, se va a emplear las variables de "min_{cambio}" (cambio mínimo), "min_i" y "min_j". Sin embargo, antes de proceder a reordenar la ruta, se localiza el vehículo asignado para esta ruta por medio del índice de la ruta. El vehículo se localizará en la variable de ruteo.

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:

$$costoactual = distancias_{(ruta_i, ruta_(i+1))} + distancias_{(ruta_j, ruta_(j+1))}$$

$$costonuevo = distancias_{(ruta_i, ruta_j)} + distancias_{(ruta_{(i+1)}, ruta_{(j+1)})}$$

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

$$cambio = costonuevo - costoactual$$

Después se compara si el cambio es menor que el cambio mínimo. Si el resultado de la condición es verdadero, 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, 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 valor de 1 en los caminos que forman parte de la ruta reordenada teniendo en cuenta el vehículo que recorre la ruta. Luego, se produce el cambio del índice 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 el ruteo mejorado denominado "$ruteo_{bl}$".

5.2.2. Resultados

Los resultados obtenidos para el modelo de ruteo de vehículos terrestre y aéreo aplicando algoritmos genéticos son los siguientes:

Depósitos	**Clientes**	**Optimo**	**Función objetivo**	**Tiempo**
1	4	si	218.9583	1.1223 secs
1	6	si	457.5517	1.8901 secs
1	8	si	481.2939	2.7406 secs
1	10	-	718.457	3.66 secs
1	12	-	759.5164	4.6499 secs
1	14	-	815.2458	5.6548 secs
1	16	-	1065.5941	7.2649 secs
1	18	-	1497.6811	9.3501 secs
1	20	-	1460.9928	11.1599 secs

Tabla 5.3: Obtención de resultados en **AG** para ruteo terrestre.

Los algoritmos genéticos retornan los resultados en menor tiempo a comparación de los **ACO** o que **GLPK**, pero el margen de variación con respecto a la función objetivo el mayor, de 7.5 % y 17.2 % por encima de la función objetivo para ruteo terrestre y aéreo respectivamente.

Depósitos	Clientes	Optimo	Función objetivo	Tiempo
1	4	si	17.6676	1.0522 secs
1	6	si	79.6563	1.8292 secs
1	8	si	79.7117	2.5798 secs
1	10	si	112.1639	3.3475 secs
1	12	-	114.9851	4.7052 secs
1	14	-	176.5485	6.2448 secs
1	16	-	211.677	7.4354 secs
1	18	-	241.7453	9.5467 secs
1	20	-	306.0682	12.5452 secs

Tabla 5.4: Obtención de resultados en **AG** para ruteo aéreo.

5.3. Comparación de metaheurísticas

Para una mejor comprensión de los resultados, se muestran estos ordenados en tablas, tanto para ruteo terrestre como aéreo.

- Comparación de metaheurísticas para ruteo terrestre

TIEMPO DE CÓMPUTO RUTEO TERRESTRE						
	GLPK		ACO		AG	
N	F. OBJ	TIEMPO	F. OBJ	TIEMPO	F. OBJ	TIEMPO
4	218.9583	0	218.9583	1.3328	218.9583	1.1223
6	457.5517	0.2	457.5517	2.9417	457.5517	1.8901
8	481.2939	0.6	481.2939	3.9348	481.2939	2.7406
10	706.733	36.6	706.733	5.3423	718.457	3.66
12	710.3713	3600.1	710.3713	6.4947	759.5164	4.6499
14	796.8193	1274.3	797.4248	8.48	815.2458	5.6548
16	1049.281	3600.3	1051.4637	18.8295	1065.594	7.2649
18	1434.251	3600.3	1403.0042	13.4699	1497.681	9.3501
20	1446.16	3600.2	1419.5279	15.8466	1460.993	11.1599

Tabla 5.5: Resultados del ruteo terrestre para **GLPK**, **ACO** Y **AG**

Finalmente, para concluir cual de la metaheurísticas genera mejores resultados, se aplica como métrica, el promedio aritmético de las funciones objetivo y tiempo de cómputo, obteniendo los siguientes resultados para ruteo terrestre:

$$\overline{X}_{ACO-F.OBJ} = 805.148, \qquad \overline{X}_{AG-F.OBJ} = 830.588 \tag{5.1}$$

$$\overline{X}_{ACO-TIEMPO} = 8.51914, \qquad \overline{X}_{AG-TIEMPO} = 5.27696 \tag{5.2}$$

La ecuación 5.1, muestra que la metaheurística **ACO** presenta mejores resultados en la función objetivo, obteniendo una mejora del 3.01 % respecto de los **AG**, y la ecuación 5.2 muestra que los **AG** consumen menos tiempo computacional para devolver una respuesta, obteniendo una mejora de 38.07 %.

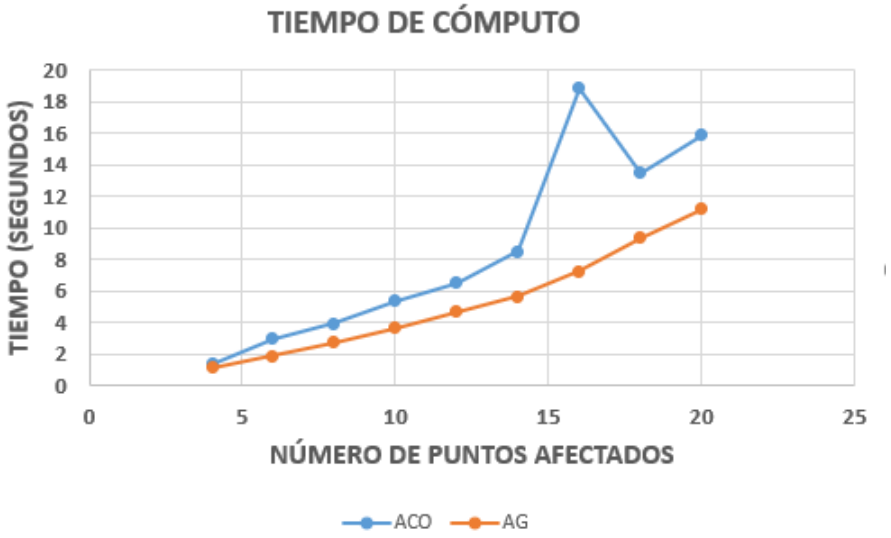


Figura 5.14: Tiempo de cómputo de ruteo terrestre para **ACO** Y **AG**

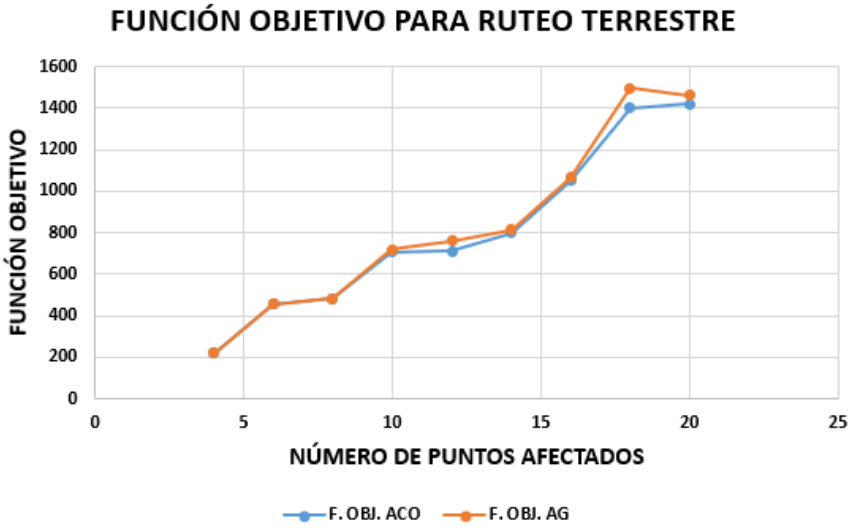


Figura 5.15: Comparación de las funciones objetivo de ruteo terrestre para **ACO** Y **AG**

- Comparación de metaheurísticas para ruteo aéreo

TIEMPO DE CÓMPUTO RUTEO AÉREO						
N°	**GLPK**		**ACO**		**AG**	
	F. OBJ	TIEMPO	F. OBJ	TIEMPO	F. OBJ	TIEMPO
4	17.66762	0	17.6676	1.26	17.6676	1.0522
6	79.65629	0.1	79.6563	2.3148	79.6563	1.8292
8	79.71172	3.4	79.7117	3.81	79.7117	2.5798
10	112.1639	21.5	112.1639	5.1369	112.1639	3.3475
12	116.7563	3600.2	114.9851	6.7401	114.9851	4.7052
14	169.27	3600.2	168.5043	8.8048	176.5485	6.2448
16	213.6586	3600.3	208.1165	11.1003	211.677	7.4354
18	237.4486	3600.3	236.1925	13.5916	241.7453	9.5467
20	261.1836	3600.2	260.6064	15.2493	306.0682	12.5452

Tabla 5.6: Resultados del ruteo aéreo para **GLPK**, **ACO** Y **AG**

Se aplica el mismo análisis para el ruteo aéreo, los resultados fueron los siguientes:

$$\overline{X}_{ACO-F.OBJ} = 141.956, \qquad \overline{X}_{AG-F.OBJ} = 148.914 \tag{5.3}$$

$$\overline{X}_{ACO-TIEMPO} = 7.55642, \qquad \overline{X}_{AG-TIEMPO} = 5.47622 \tag{5.4}$$

La ecuación 5.3, muestra que la metaheurística **ACO** presenta mejores resultados en la función objetivo, obteniendo una mejora del 4.67 % respecto de los **AG**, y la ecuación 5.4 muestra que los **AG** consumen menos tiempo computacional para devolver una respuesta, obteniendo una mejora de 27.54 %.

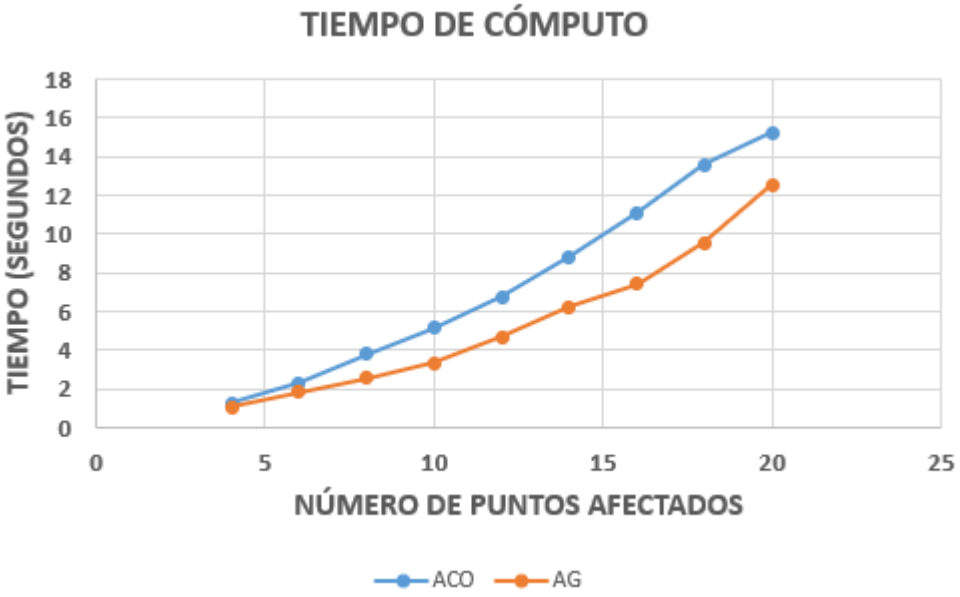


Figura 5.16: Tiempo de cómputo de ruteo aéreo para **ACO** Y **AG**

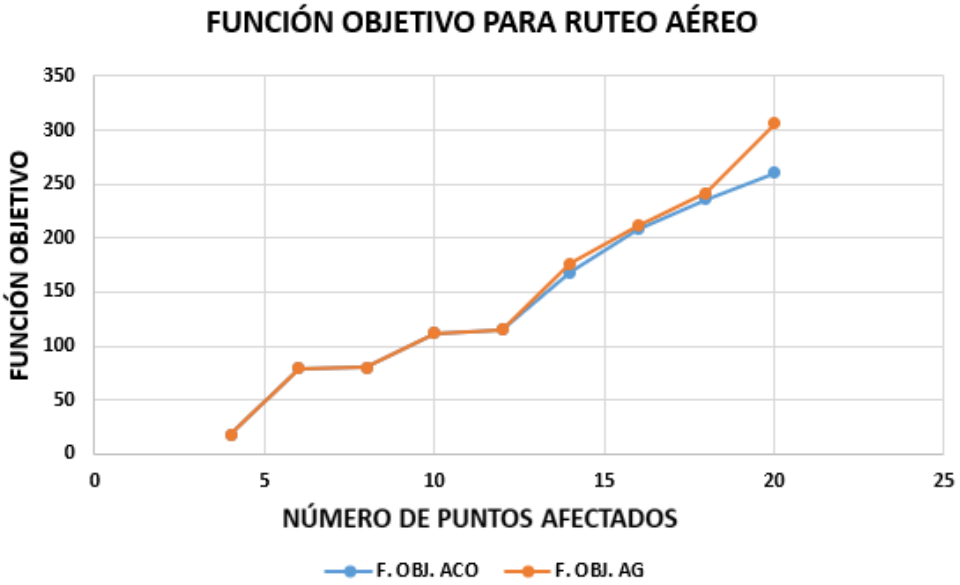


Figura 5.17: Comparación de las funciones objetivo de ruteo aéreo para **ACO** Y **AG**

CONCLUSIONES

- La ejecución de los modelos desde la terminal del sistema utiliza menos recursos computaciones que la ejecución en el editor de **GLPK** denominado **Gusek**.
- El modelo de programación lineal implementado en **GLPK**, devuelve soluciones óptimas para no más de 14 puntos afectados evaluados en un tiempo de cómputo de 1 hora.
- La metaheurística **ACO** genera mejores valores en la función objetivo que la metaheurística **AG** empleando el promedio aritmético como métrica de comparación.
- Los **AG** emplean menos tiempo de cómputo en relación a los algoritmos **ACO**.
- Se recomienda el uso de la metaheurística inspirada en colonias de hormigas para problemas de ruteo de vehículos aplicado en zonas de emergencia

BIBLIOGRAFÍA

Al Theeb, N. and Murray, C. (2017). Vehicle routing and resource distribution in postdisaster humanitarian relief operations. *International Transactions in Operational Research*, 24(6):1253–1284.

Alinaghian, M., Aghaie, M., and Sabbagh, M. S. (2019). A mathematical model for location of temporary relief centers and dynamic routing of aerial rescue vehicles. *Computers & Industrial Engineering*, 131:227–241.

Allen, S. (2017). A two stage vehicle routing algorithm applied to disaster relief logistics after the 2015 nepal earthquake. *arXiv preprint arXiv:1709.00162*.

Alva Cabrera, R. A. (2014). Plan de despacho para la distribución de ayuda humanitaria en caso de un terremoto de gran magnitud en lima metropolitana y callao.

Benavente Sotelo, R. A. (2016). Plan de ruteo para la distribución de ayuda humanitaria no alimentaria ante un terremoto de gran magnitud en lima metropolitana y callao.

Brassard, G., Bratley, P., and Giner, R. G.-B. (1997). *Fundamentos de algoritmia*, volume 2. Prentice Hall Madrid.

Caicedo, A., Wagner, G., and Mendez, R. (2010). Introducción a la teoría de grafos. *Ediciones Elizcom*, pages 1–33.

Du, K.-L. and Swamy, M. (2016). Search and optimization by metaheuristics. *Techniques and Algorithms Inspired by Nature; Birkhauser: Basel, Switzerland*.

García Avellaneda, A. J. A. (2019). Diseño de un plan de evacuación en caso de emergencia por tsunami en el distrito la punta usando métodos de optimización.

Gharib, Z., Bozorgi-Amiri, A., Tavakkoli-Moghaddam, R., and Najafi, E. (2018). A cluster-based emergency vehicle routing problem in disaster with reliability. *Scientia Iranica*, 25(4):2312–2330.

Gutin, G. and Punnen, A. P. (2006). *The traveling salesman problem and its variations*, volume 12. Springer Science & Business Media.

Guzman, F. G., Castro, V. N., Melquiades, J. R., Segura, E. L., and Segura, F. G. (2020). Estado del arte del ruteo de vehículos aplicado a desastres naturales en sudamérica. *Selecciones Matemáticas*, 7(02):340–353.

Han, S., Park, J., and Jeong, H. (2018). A solution procedure for emergency logistics problem in disaster scene. *Industrial Engineering & Management Systems*, 17(2):166–176.

Jeanmonod, D., Suzuki, K., et al. (2018). We are intechopen the worlds leading publisher of open access books built by scientists for scientists. *Intech open*.

Khouadjia, M. R., Sarasola, B., Alba, E., Talbi, E.-G., and Jourdan, L. (2013). *Metaheuristics for dynamic vehicle routing*. Springer.

Korkou, T., Souravlias, D., Parsopoulos, K., and Skouri, K. (2016). Metaheuristic optimization for logistics in natural disasters. In *International Conference on Dynamics of Disasters*, pages 113–134. Springer.

Li, Y. and Chung, S. H. (2019). Disaster relief routing under uncertainty: A robust optimization approach. *IISE Transactions*, 51(8):869–886.

Mandal, J. K., Mukhopadhyay, S., and Dutta, P. (2010). Multi objective optimization.

Merino, M. (2015). Técnicas clásicas de optimización. parte i: programación lineal y no lineal.

Mguis, F., Zidi, K., Ghedira, K., and Borne, P. (2014). Distributed and guided genetic algorithm for humanitarian relief planning in disaster case. In *Distributed Computing and Artificial Intelligence, 11th International Conference*, pages 149–156. Springer.

Montes Orozco, E. et al. (2017). *Metaheurísticas para el problema de ruteo de vehículos con ventanas de tiempo (VRP-TW)*.

Nolz, P. C., Doerner, K. F., Gutjahr, W. J., and Hartl, R. F. (2010). A bi-objective metaheuristic for disaster relief operation planning. In *Advances in multi-objective nature inspired computing*, pages 167–187. Springer.

Olivera, A. (2004). Heurísticas para problemas de ruteo de vehículos. *Reportes Técnicos 04-08*.

Pareja Villegas, C. A. and Rodriguez Leiva, X. M. (2016). Determinantes del número de damnificados por causa de un terremoto en lima metropolitana y callao y contraste de medidas de respuestas a través de modelos de programación lineal entera para la distribución de bienes para ayuda humanitaria.

Penna, P. H. V., Santos, A. C., and Prins, C. (2018). Vehicle routing problems for last mile distribution after major disaster. *Journal of the Operational Research Society*, 69(8):1254–1268.

Ramos, A., Sánchez, P., Ferrer, J. M., Barquín, J., and Linares, P. (2010). Modelos matemáticos de optimización. *Publicación Técnica*, 1.

Schaeffer, E. (2011). Complejidad computacional de problemas y el análisis y diseño de algoritmos. *Publicación en línea*, 34.

Toth, P. and Vigo, D. (2002). *The vehicle routing problem*. SIAM.

Ulmer, M. W. (2017). *Approximate dynamic programming for dynamic vehicle routing*, volume 61. Springer.

Wang, H., Du, L., and Ma, S. (2014). Multi-objective open location-routing model with split delivery for optimized relief distribution in post-earthquake. *Transportation Research Part E: Logistics and Transportation Review*, 69:160–179.

Xiong, X., Zhao, F., Wang, Y., and Wang, Y. (2019). Research on the model and algorithm for multimodal distribution of emergency supplies after earthquake in the perspective of fairness. *Mathematical Problems in Engineering*, 2019.

Yi, W. and Kumar, A. (2007). Ant colony optimization for disaster relief operations. *Transportation Research Part E: Logistics and Transportation Review*, 43(6):660–672.

Zirour, M. (2008). Vehicle routing problem: models and solutions. *Journal of Quality Measurement and Analysis JQMA*, 4(1):205–218.