# RUTEO DE VEHÍCULOS REFRIGERANTES Y DE TIPO GENERAL PARA LA ENTREGA DE MÚLTIPLES PRODUCTOS PERECEDEROS

**JESUS DAVID GALARCIO NOGUERA**

**UNIVERSIDAD DE CÓRDOBA**
**FACULTAD DE INGENIERÍA**
**INGENIERÍA INDUSTRIAL**
**MONTERÍA, CÓRDOBA**
**2017**

# RUTEO DE VEHÍCULOS REFRIGERANTES Y DE TIPO GENERAL PARA LA ENTREGA DE MÚLTIPLES PRODUCTOS PERECEDEROS

**JESUS DAVID GALARCIO NOGUERA**

**Trabajo de grado presentado, en la modalidad de Trabajo de Investigación y/o Extensión, como parte de los requisitos para optar al Título de Ingeniero Industrial.**

**Director (es):**
**JORGE MARIO LÓPEZ PEREIRA, M.Sc.**
**HELMAN ENRIQUE HERNÁNDEZ RIAÑO, Ph.D.**

**UNIVERSIDAD DE CÓRDOBA**
**FACULTAD DE INGENIERÍAS**
**INGENIERÍA INDUSTRIAL**
**MONTERÍA, CÓRDOBA**
**2017**

**La responsabilidad ética, legal y científica de las ideas, conceptos y resultados del proyecto, serán responsabilidad del autor.**

**Artículo 61, Acuerdo N◦ 093 del 26 de noviembre de 2002 del Consejo Superior.**

**Nota de aceptación**

_____

_____

_____

_____

_____

**Firma del jurado**

_____

**Firma del jurado**

*A Dios*
*A mis padres Elver y Ruth*

***Agradecimiento especial a:***

*Jorge Mario López Pereira y Helman Enrique Hernández Riaño*
*Directores de esta investigación*

*Por confiarme retos en el ámbito de la investigación que me han permitido descubrir un mundo de posibilidades*

***Agradecimientos:***

*A mis padres, por su apoyo incondicional y su comprensión durante toda la vida y, especialmente, durante el desarrollo de esta investigación*

*A mi novia y psicóloga, por su comprensión y sus palabras que me alentaron a no rendirme hasta cumplir mis objetivos*

*A mis compañeros, los "galácticos", que día a día me acompañaron y animaron durante el desarrollo de mi carrera profesional*

**TABLA DE CONTENIDO.**

**Pág.**

## LISTA DE ANEXOS

**Pag.**



## LISTA DE TABLA

**Pag.**

## LISTA DE FIGURAS

**Pag.**



## LISTA DE GRÁFICOS

**Pag.**

## RESUMEN

Para la presente investigación, la meta establecida se centraba en la formulación de un modelo matemático para el problema de ruteo con dos tipos de vehículos (refrigerado y no refrigerado) para representar la minimización de la pérdida de frescura en la entrega de múltiples productos perecederos, y el diseño de un algoritmo metaheurístico para su solución; sin embargo, como paso anterior a la formulación del modelo matemático para múltiples productos perecederos (Caso II), se hizo necesario formular un modelo mono producto que, posteriormente, pudiera ser extendido (Caso I). De esta forma, fue posible el desarrollo de dos (2) modelos matemáticos para los cuales se diseñaron (2) algoritmos independientes: N1 para el Caso I y N2 para el Caso II. Luego de esto, se realizaron 2 experimentos, comparando los algoritmos diseñados con otras metaheurísticas seleccionadas y realizando los análisis estadísticos respectivos.

A partir del análisis estadístico de cada experimento, se obtuvieron conclusiones relacionadas con el desempeño de los algoritmos comparados en términos de calidad de las soluciones generadas. En este orden de ideas, se encontró que, para ambos experimentos, existen diferencias estadísticamente significativas entre la calidad de los resultados obtenidos por los algoritmos comparados. Adicionalmente, la frecuencia en que cada algoritmo obtuvo las mejores soluciones y los mejores promedios en cada instancia fue considerada en el análisis.

Para el caso I, el algoritmo cromático tuvo el mejor desempeño entre los algoritmos comparados (genético, PSO, cromático y N1). Para el caso II, se compararon tres algoritmos (genético, cromático y N2), siendo N2 el algoritmo con mejor desempeño.



**Palabras Clave:** VRP; productos perecederos; modelo matemático; metaheurística;

## ABSTRACT

In this research, the main goal was focused on the formulation of a mathematical model for the vehicle routing problem with two vehicles (refrigerated and non-refrigerated) to represent the minimization of the loss of freshness in the delivery of multiple perishable products and the design of a metaheuristic algorithm to solve the problem. As a previous step to the mathematical model formulation for multiple perishable products (Case II) it was necessary to formulate a single-product model that could be extended (Case I). In this way, it was possible to develop two (2) mathematical models and two (2) algorithms to solve each model: N1 for case I and N2 for case II. Afterwards, two experiments were executed to compare the designed algorithm with another selected metaheuristics and a respective statistical analysis was performed.

From the statistical analysis of each experiment, conclusions related with the performance of the compared algorithms in terms of quality of solutions were obtained. It was found that, for both experiments, there were statistically significant differences between the qualities of the obtained results. Additionally, the frequencies at which each algorithm obtained the best results and the best average in each instance was considered in the analysis. For case I, the chromatic algorithm had a better performance among the compared algorithms (genetic, particle swarm optimization and N1). In case II, three algorithms (genetic, chromatic, N2) were compared, being N2 the algorithm with the best performance.



**Keywords**: VRP; perishable products; mathematical model; metaheuristic;

# CAPÍTULO 1: GENERALIDADES

En este capítulo, se presentan los antecedentes del problema de estudio y se describen aspectos específicos inherentes al mismo. De igual manera, se plantean los interrogantes, se formulan los objetivos, la justificación, el alcance y las limitaciones de la presente investigación.

## 1.1 DESCRIPCIÓN DEL PROBLEMA

En productos con una vida útil reducida, como frutas y hortalizas, o bien, en aquellos que requieren refrigeración, como las carnes, la frescura es un factor determinante para garantizar su calidad y disminuir su desperdicio en el proceso de distribución.

Actualmente, las organizaciones encargadas de la operación logística deben planificar la entrega de productos a grandes cantidades de clientes, contando con una flota de vehículos disponibles y una serie de restricciones asociadas; esta planificación es realizada, a menudo, de forma tradicional, lo que puede incidir en una pérdida de la buena imagen corporativa para la empresa distribuidora.

Cabe mencionar que disminuir la pérdida y el desperdicio de alimentos se ha convertido en un tema en torno al cual, existe un compromiso a nivel mundial enmarcado en los Objetivos de Desarrollo Sostenible que fueron aprobados en el año 2015. De hecho, sólo en Colombia, en el eslabón de distribución y *retail* se desperdician cerca de 2 millones de

toneladas de alimentos, de los cuales, 1.8 millones de toneladas son frutas, verduras y cereales (DNP 2016).

Dentro de los factores que pueden ocasionar problemas a la hora de planificar rutas en proceso de distribución, está la complejidad de la tarea por la cantidad numerosa de variables, la inexperiencia y desconocimiento de métodos o procedimientos específicos que dan solución al problema, la falta de información relevante del sistema para la toma de decisiones, como datos históricos de tiempos de viajes, tiempos de servicios y demandas, la carencia de expertos en el tema o herramientas sistémicas suficientes para solucionar el problema (software eficiente y eficaz), la falta de información acerca de la ubicación geográfica de cada uno de los clientes y las variaciones constantes del sistema real (Soto 2012).

Hoy por hoy, existe una amplia gama de herramientas logísticas de planificación de rutas, las cuales, por lo general, están enfocadas en la reducción de los costos de transporte en el proceso de distribución, sin embargo, muy pocas están orientadas a reducir la pérdida de frescura de los productos en este eslabón de la cadena productiva. Bajo este enfoque, se hace necesario desarrollar herramientas de planificación que tomen en cuenta este factor a la hora de diseñar las rutas.

El problema de ruteo para la entrega de productos perecederos con vehículos refrigerados y vehículos sin sistemas de refrigeración, consiste en generar rutas cercanas a la óptima para el transporte de productos con características de tipo perecedero, haciendo énfasis en la conservación de la frescura del producto durante el proceso de distribución. Este tipo de problema se caracteriza por su dificultad para hallar una solución óptima (NP-duro) (Golden y Assad 1989), siendo necesaria la ayuda de herramientas computacionales para

la generación de rutas que minimicen los tiempos de viaje y la cantidad de producto desperdiciado, permitiendo optimizar el proceso de distribución de productos perecederos. El presente proyecto aborda la modelación matemática para problema de ruteo de vehículos refrigerantes y de tipo general para la entrega de múltiples productos perecederos conociendo la demanda, la capacidad de los vehículos (flota heterogénea) y el número de vehículos disponibles de ambos tipos. De igual forma, comprende el diseño de una herramienta computacional para solucionar el modelo planteado.

Con los productos derivados de esta investigación, se busca contribuir a la reducción de la pérdida de frescura de los productos perecederos en el proceso de distribución, y así, incidir en la disminución del desperdicio de alimentos en este eslabón de la cadena productiva.

## 1.2 FORMULACIÓN DEL PROBLEMA

En la presente investigación, se brinda una propuesta algorítmica para el diseño de rutas que permitan minimizar la pérdida de frescura de los productos perecederos en el proceso de distribución. Se utilizarán diferentes metaheurísticas en la solución del problema, en aras de contrastar la calidad de las soluciones generadas, respondiendo a la siguiente pregunta:

¿Cómo diseñar una meta heurística que permita optimizar la frescura en el problema de ruteo de vehículos refrigerados y de tipo general para la entrega de múltiples productos perecederos?

## 1.3 SISTEMATIZACIÓN DEL PROBLEMA

Con el ánimo de posibilitar el alcance de los objetivos, es menester direccionar la investigación, para lo cual, se debe definir una secuencia lógica a partir del planteamiento de interrogantes que, a su vez, apoyaran el ordenamiento permanente de la información.

### 1.3.1 Estado del arte

Para abordar un problema, deben ser estudiados y valorados los trabajos existentes en torno al mismo, buscando conocer, aclarar y profundizar conceptualmente sobre los aspectos inherentes este y los métodos y procedimientos desarrollados por otros investigadores para su solución, lo que permitirá determinar qué se puede aportar para seguir ampliando la frontera del conocimiento. De esta forma, surge la siguiente pregunta: ¿Cuál es el estado del arte del problema de ruteo de vehículos para vehículos refrigerantes y de tipo general para la entrega de productos perecederos?

### 1.3.2 Modelación matemática

Para representar el comportamiento de un problema, se hace necesario estudiar los parámetros, las variables que puedan incidir en su solución e identificar las posibles restricciones a las que se encuentra sujeto, lo que nos permite llegar al siguiente interrogante:

¿Cuál será la formulación del modelo matemático que represente la maximización de la frescura en el producto entregado, en el problema de ruteo de vehículos para vehículos refrigerados y de tipo general?

### 1.3.3 Diseño de la metaheurística

Luego de tener conocimientos específicos sobre el problema, su representación matemática y las metaheurísticas comúnmente implementadas para abordar problemas de este tipo, es posible proponer una nueva metaheurística para solucionarlo. Por lo tanto, es razonable responder a la pregunta:

¿Cómo diseñar una nueva metaheurística que permita solucionar el modelo propuesto?

### 1.3.4 Comparación de metaheurísticas

Para conocer el desempeño y la calidad de las soluciones generadas por la metaheurística propuesta, es necesario compararla con otras metaheurísticas que se encuentren en la literatura, pudiendo establecer, del grupo seleccionado, la más adecuada para abordar este problema específico. Surge el siguiente interrogante:

¿Cómo establecer la metaheurística más adecuada para solucionar el problema?

## 1.4 OBJETIVOS

### 1.4.1 Objetivo general

Diseñar una meta heurística para optimizar la frescura en el problema de ruteo de vehículos refrigerados y de tipo general para la entrega de múltiples productos perecederos.

### 1.4.2 Objetivos específicos

- Realizar una descripción general del problema de ruteo de vehículos a través de una revisión bibliográfica

- Formular un modelo matemático que represente la maximización de la frescura en el producto entregado
- Diseñar un algoritmo meta heurístico eficiente que solucione el modelo matemático propuesto
- Comparar los resultados de la meta heurística desarrollada con otros algoritmos propuestos en la literatura y seleccionar el mejor algoritmo para resolver esta variante del problema de ruteo de vehículos

## 1.5 JUSTIFICACIÓN

A propósito de los Objetivos de Desarrollo Sostenible, una de las 169 metas que deben alcanzar los países suscritos a la Asamblea General de las Naciones Unidas para el año 2030, es "reducir a la mitad el desperdicio de alimentos per cápita mundial en la venta al por mayor y a nivel de consumidores y reducir las pérdidas de alimentos en las cadenas de producción y suministro, incluidas las pérdidas posteriores a la cosecha" (Naciones Unidas 2015)

Cabe señalar, que las **pérdidas** se refieren a la disminución de alimentos disponibles para el consumo humano en las etapas de producción agropecuaria, pos cosecha y almacenamiento, y procesamiento industrial, mientras que el **desperdicio** se define como la disminución de los mismos en las fases de distribución, *retail* y consumo. De acuerdo al informe de Pérdida y desperdicio de alimentos, en Colombia de pierden y desperdician un total de 9,76 millones de toneladas de alimentos al año, de los cuales, un 20.6% (2.01 millones de toneladas) se desperdicia en la distribución y *retail* (DNP 2016)**.**

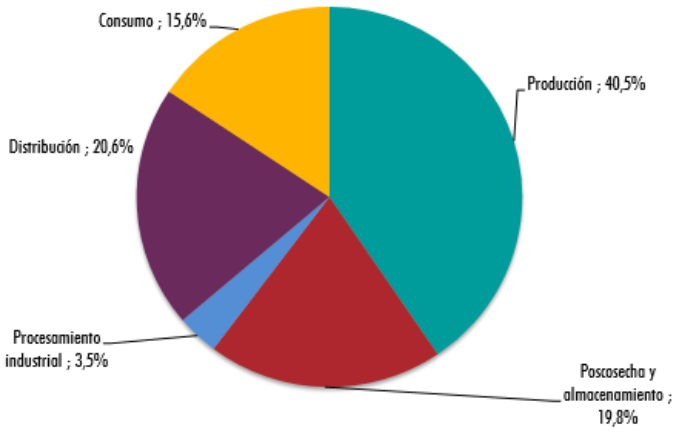



**Gráfico 1.** Distribución de pérdida y desperdicio por eslabón de la cadena alimentaria. Tomada de: Pérdida y desperdicio de alimentos en Colombia, 2016

Esto refleja que la disminución de la pérdida y el desperdicio de alimentos es un tema que no da espera, y deben ser adoptadas medidas como el desarrollo de herramientas metodológicas en el ámbito técnico y tecnológico que sirvan de base a las empresas logísticas y permitan mitigar el riesgo de desperdicio de productos perecederos en el proceso de distribución, en el cual, juega un papel importante el tiempo de transporte y el nivel de frescura de los productos al momento de la entrega.

Cuando se habla de productos perecederos en el proceso logístico, normalmente esto es asociado al diseño de una cadena de frío que permita mantener un producto bajo condiciones aceptables a lo largo de la misma. En países en vía de desarrollo, como Colombia, es posible evidenciar cierta carencia en términos de bodegas de frio disponibles y de vehículos refrigerados para el transporte de productos perecederos, teniendo en cuenta su alto costo y las limitantes económicas para crecientes economías de mercado, lo que provoca pérdidas constantes en la producción, una disminución en la calidad de los productos entregados y reduce la posibilidad de competir.

Con este proyecto, se espera optimizar el proceso de distribución de productos perecederos y reducir la cantidad de producto desperdiciado durante el mismo, generando un mecanismo para impulsar productos locales al mercado nacional e internacional.

En este orden de ideas, se busca desarrollar un modelo matemático para la entrega de múltiples productos perecederos. El problema consiste en generar rutas para la distribución de un producto desde un centro de distribución hacia una cantidad n de clientes repartidos geográficamente, para esto, debe conocerse inicialmente el tiempo de viaje entre el centro de distribución y cada cliente y entre un cliente y otro, además, debe conocerse la cantidad de producto requerido por cada cliente y la capacidad y el tipo de cada vehículo de la flota disponible para el transporte, teniendo en cuenta que la flota puede contar con vehículos con sistema de refrigeración y con vehículos comunes (o no refrigerados). Adicionalmente, se deben estudiar los factores que puedan incidir en la pérdida de frescura del producto durante la ruta, considerando agentes como la temperatura, el contacto con el aire y características inherentes al ambiente y al producto, tales como la humedad relativa y el "tiempo de vida" de cada producto.

La pertinencia de este proyecto no solo está sustentada en la baja eficiencia y eficacia de los métodos actuales de distribución, sino también en el impacto que tienen las pérdidas de producto en la economía de las regiones de Colombia y en la baja calidad de los productos cuando llegan a destino, ya sea por una mala planificación en el proceso de distribución o la inexistencia de esta, o por la omisión de factores críticos asociados al producto en el diseño de la ruta.

A nivel mundial, en las últimas décadas, el desarrollo de nuevos algoritmos de búsqueda aplicados a los problemas de optimización ha experimentado un rápido crecimiento, a

pesar de ello, dichos métodos siguen aún rezagados con respecto a ciertos factores como la calidad de sus respuestas y el costo en tiempo computacional, en detrimento de la deseada eficiencia en el proceso de obtención de las soluciones (Sabie y Mestra 2011). De allí la gran importancia que posee el desarrollo continuo de nuevas técnicas y algoritmos que aporten soluciones de alta calidad y un bajo costo en tiempo computacional. Las posibilidades de aprovechar las ventajas que los algoritmos y los nuevos métodos de optimización aportan, es prácticamente ilimitada, siendo utilizables en infinidad de aplicaciones científicas, industriales, económicas, tecnológicas, etc. Además, en vista de que los problemas de optimización surgen en muchos campos diferentes, numerosos estudios se centran sobre todo en el desarrollo de nuevos métodos de optimización (Kuo y Zulvia 2015)**.**

Hasta donde el autor tiene conocimiento, la investigación existente en torno al tema que más se aproxima a este proyecto está enfocada a la comparación entre vehículos refrigerados y no refrigerados, diseñando y aplicando un modelo a pequeña escala para un alcance de 10 cuadras y resolviéndolo mediante un método heurístico (Song y Ko 2016), sin embargo, aún no se ha considerado el diseño de un modelo a mayor escala, tampoco se ha solucionado a partir de métodos meta heurísticos, los cuales, generalmente, obtienen mejores soluciones que los heurísticos gracias a la búsqueda iterativa de óptimos globales en el espacio de soluciones y a la aplicación de principios basados en métodos de búsqueda intrínsecos en la naturaleza.

# CAPÍTULO 2: MARCO TEÓRICO

En este capítulo, se presentan los conceptos teóricos necesarios para entender el problema y los avances en torno al mismo, y conocer algunos de los métodos desarrollados hasta el momento para su solución.

## 2.1 INTRODUCCIÓN

Hoy en día, el problema de distribuir productos desde ciertos depósitos a sus usuarios finales juega un papel central en la gestión de los sistemas logísticos y su adecuada planificación puede significar considerables ahorros. Estos, justifican en gran medida la utilización de técnicas de Investigación Operativa como facilitadoras de la planificación, dado que se estima que los costos del transporte representan entre el 10% y el 20% del costo final de los bienes y los ahorros estimados por el uso de procesos computarizados para la planeación de la distribución están, generalmente, entre el 5% y el 20% de los costos globales de transporte (Toth y Vigo 2002).

Es posible afirmar que normalmente los costos de transporte se hallan entre un tercio y dos tercios de los costos logísticos totales, por lo que se busca mejorar la eficiencia mediante la preocupación del personal de trabajo, y aprovechar al máximo el equipo de transporte, así como mejorar el servicio al cliente, encontrando los mejores caminos que debería seguir un vehículo en una red (Ballou 2004).

## 2.2 PROBLEMA DE RUTEO DE VEHÍCULOS (VRP)

El problema VRP puede definirse como: "la determinación de la ruta óptima para una flota de vehículos que parten de uno o más depósitos (almacenes) para satisfacer la demanda de varios clientes dispersados geográficamente" (Dantzig y Ramser 1959)**.**

La variante más simple y mayormente estudiada es el CVRP, o ruteo de vehículos con restricciones de capacidad, el cual puede describirse como un problema de teoría de grafos. Sea $G = (V, A)$ un grafo completo, donde $V = \{0, \ldots, n\}$, es el conjunto de vértices y A es el conjunto de arcos. Los vértices $i = 1, \ldots, n$ representan a los clientes y el depósito puede ser el vértice $0$, o los vértices $0$ y $n+1$. También se asocia un costo no negativo, $c_{ij}$, a cada arco $(i, j) \in A$, y representa el costo de viaje de ir desde el vértice $i$ hasta el vértice $j$. Cada cliente $i$ $(i = 1, \ldots, n)$, se asocia a una demanda conocida no negativa $d_i$. También, se cuenta con una flota de vehículos idénticos de tamaño $K$ con una capacidad $C$, para la cual debe comprobarse que $d_i \leq C$ para cada $i = 1, \ldots, n$. En general, el problema se debe solucionar de tal forma que: cada vehículo realice exactamente un circuito y regrese al depósito; cada cliente sea visitado estrictamente por un vehículo; y la suma de las demandas de los clientes visitados en un circuito no excedan la capacidad del vehículo, $C$ (Toth y Vigo 2002)**.**

A través del tiempo, han aparecido muchas variantes del problema de ruteo de vehículos, las cuales han surgido a partir de requerimientos de aplicaciones prácticas en una infinidad de escenarios. Estos casos particulares a menudo dependen de requerimientos de los clientes (como ventanas de tiempo, múltiples periodos de planeación, entre otros), de las características de la red y de los vehículos (como múltiples depósitos, congestión, flota heterogénea, entre otros), de las necesidades de conducción (como regulación de horas

de trabajo, descansos para el almuerzo, entre otros) o de una mejora en la integración de decisiones en una planeación táctica o estratégica (ruteo de inventario o de localización) (Vidal et al 2014). Cabe señalar que cada vez se van generando más casos particulares de acuerdo a las necesidades específicas de problemas reales.

## 2.3 VRP PARA VEHÍCULOS REFRIGERADOS Y DE TIPO GENERAL PARA LA ENTREGA DE PRODUCTOS PERECEDEROS

Los productos altamente perecederos tienen un importante rol dentro de los procesos operativos de distribución, particularmente en las tareas de planeación de rutas de vehículos (Amorim y Almada-Lobo 2014). En este contexto, el nivel de frescura de los productos al momento de la entrega es relevante y puede verse influenciado por el tiempo de duración y la temperatura ambiente. Además, cuando los productos de este tipo son entregados a una gran cantidad de clientes, es difícil mantener la frescura debido a largos tiempos de viaje y frecuentes paradas. Para reducir el deterioro de los productos con una vida útil corta, es importante que estos sean entregados de la forma más oportuna posible (Hsu et al. 2007).

Ahora bien, si se considera la frescura como variable fundamental en la satisfacción de los clientes de productos perecederos, se hace posible la formulación de un modelo logístico de transporte como un problema de ruteo de vehículos para este tipo de productos que permita optimizar el plan de ruta en función de esta variable. La dificultad en la solución de los problemas de enrutamiento radica en que la mayoría de ellos pertenecen a la clase de problemas de optimización combinatoria NP-Hard (o de difícil solución), lo que ha llevado al desarrollo e implementación de una infinidad de técnicas heurísticas y

meta heurísticas con el fin de encontrar soluciones de alta calidad en un tiempo computacional razonable (Dyer y Stougie 2006).

En torno a productos de tipo perecedero, han existido contadas investigaciones que han hecho énfasis en la planeación de la cadena de suministro. Se destaca la programación y ruteo de vehículos con flota heterogénea para la distribución de leche fresca (Tarantilis y Kiranoudis 2001), el desarrollo de un proveedor de servicios de distribución logística basada en técnicas metaheurísticas para centrales de alimentos que venden y distribuyen productos alimenticios frescos (Prindezis et al. 2003) y la construcción de un algoritmo para la distribución de vegetales frescos tomando la frescura como un factor crítico, incluyendo restricciones de ventanas tiempo y tiempos de viaje según la hora del día (Osvald y Stirn 2008). Con relación a la distribución de alimentos fríos o congelados existen contadas investigaciones. Wang y Yu (2012) propone un modelo de optimización para la distribución de alimentos refrigerados con ventanas de tiempo.

Algunos autores consideran el diseño de la cadena de frío para el control de la frescura del producto entregado, sin embargo, la mayoría están enfocados en el diseño de la red de suministro y no abordan específicamente el ruteo de vehículos (Song y Ko 2016). Otros trabajos relacionados van desde la aplicación de heurísticas para la distribución de concreto pre-mezclado desde plantas de producción hasta sitios de construcción con una flota heterogénea (Schmid et al. 2009) hasta la formulación de estrategias de innovación logística en el sector hortofrutícola (Espín 2004).

De Keizer et al. (2015) realiza optimización híbrida y simulación para el diseño de una red logística para la distribución de productos perecederos, tales como flores y otros productos agrícolas. A partir de su investigación, demuestra que, si la pérdida de calidad

en los productos transportados no es tenida en cuenta en la optimización, se entregarán productos de baja calidad al cliente final. Por su parte, Song y Ko (2016) formuló un problema de ruteo de vehículos con vehículos refrigerantes y de tipo general para la entrega de múltiples productos perecederos en el que maximiza la suma total de satisfacción del cliente, la cual depende de la frescura del producto entregado. Esta investigación buscaba comparar la eficiencia de ambos tipos de vehículos para la conservación de la frescura y la disponibilidad de los mismos, considerando un problema de ruteo con un alcance de 5 km x 5 km. Este problema fue resuelto mediante la aplicación de una heurística. En el mismo trabajo, es posible evidenciar aún no han sido empleadas metaheurísticas para la solución del problema de ruteo con vehículos refrigerados y vehículos sin sistemas de refrigeración para la entrega de productos perecederos gestionando la frescura.

Para el desarrollo de esta investigación, se tomará la frescura como un factor crítico para la satisfacción de los clientes. Este estudio aborda el problema de ruteo de vehículos refrigerantes y de tipo general para la entrega de productos perecederos, incluyendo aspectos como flota heterogénea y tiempos de servicio para cada cliente. Para la construcción de la función objetivo, será necesario tomar conceptos del problema de mínima latencia (Minimum Latency Problem (MLP)), también conocido como VRP con capacidad acumulativa (Cumulative Capacited Vehicle Routing Problem (CCVRP)) (Ngueveu et al. 2010), cuyo objetivo es minimizar la suma de tiempos de llegada hasta los vértices (clientes). Para conocer más detalles de esta variante del problema del agente viajero (Traveling Salesman Problam (TSP)) puede consultarse el trabajo de Silva et al. (2012)**.** Así mismo, se buscará introducir simplicidad y flexibilidad al modelo previamente

formulado por Song y Ko (2016)**,** presentándolo como una variante de VRP con tipo de vehículo como atributo de asignación y una función objetivo diseñada para minimizar la pérdida de frescura como atributo de evaluación de acuerdo a las categorías presentadas en Vidal et al. (2014)**.**

## 2.4 MÉTODOS DE SOLUCIÓN

La dificultad en la solución de los problemas de enrutamiento es que la mayoría de ellos pertenecen a la clase de problemas de optimización combinatoria que se caracterizan como NP-Hard y, por lo tanto, no son fácilmente abordables utilizando técnicas exactas considerando cantidades de tiempo no razonables requeridas para su solución, por lo que es necesario hacer uso de métodos aproximados como las técnicas heurísticas y meta heurísticas.

### 2.4.1 Métodos exactos

Pese a las limitaciones relacionadas con los altos tiempos de cómputo necesarios para la obtención de la mejor solución a través de este tipo de métodos, se han hecho esfuerzos aras de mejorar la eficiencia de los mismos. Se destacan los algoritmos de ramificación y acotación. El algoritmo Branch and Bound (B&B) tiene como objetivo realizar una enumeración sistemática de las soluciones, se evalúan los subconjuntos de la solución respecto a su contribución a la función objetivo, se establecen unas cotas inferiores y superiores y de acuerdo a esto, se decide si se ramifica o no el árbol de soluciones. Ante este algoritmo, se destaca una mejora propuesta por Padberg y Rinaldi (1991)**,** donde al B&B clásico se le integra el método de corte de planos, dando origen al método Branch

and Cut (B&C). Así mismo, se realiza una combinación entre el B&B clásico y el algoritmo de generación de columnas, lo que da origen a un algoritmo conocido como Branch and Price (B&P) (Barnhart et al. 1998).

Diferentes métodos exactos destacados en la solución VRP involucran relajación lagrangiana (Li et al. 2009), descomposición de Dantzig-Wolfe, programación con restricciones y programación dinámica (Righini y Salani 2006). Para mayor detalle de estos métodos, en el trabajo de Kallehauge (2008) se encuentra una revisión de los algoritmos exactos propuestos en las últimas 3 décadas para la solución del VRPTW.

### 2.4.2 Heurísticas

Actualmente, el problema de ruteo de vehículos es difícil de solucionar, de hecho, con ayuda de métodos exactos sólo pueden resolverse óptimamente instancias relativamente pequeñas. Las heurísticas tienen también otra ventaja sobre los métodos exactos, que es que son menos sofisticadas algorítmicamente, con lo que es más fácil programarla y a la vez son más fáciles de comprender. En el momento en que los métodos exactos se vuelven inadecuados, las heurísticas se convierten en las más usadas en la práctica (Cordeau et al. 2002).

Las heurísticas, son procedimientos que realizan una exploración limitada del espacio de búsqueda y dan soluciones de calidad aceptable (no necesariamente óptimas) en tiempos de cálculo moderados. Los primeros esfuerzos en torno a este tipo de herramientas, estuvieron centrados en la velocidad de obtención de las soluciones factibles y la realización de procesos posteriores de optimización a dichas soluciones, es el caso del algoritmo de ahorro (Clarke y Wright 1964), el algoritmo de barrido (Gillett y Miller 1974) y el algoritmo de Fisher y Jaikumar (Fisher y Jaikumar 1981).

De acuerdo a Vásquez Morales (2007), existen cinco familias de heurísticas para resolver el VRP: de ahorro, de inserción, de mejora/ intercambio, Cluster First, Route Second y Route First, Cluster Second.

### 2.4.3 Metaheurísticas

Las metaheurísticas son procedimientos genéricos de exploración del espacio de soluciones para problemas de optimización y búsqueda. Proporcionan una línea de diseño que, adaptada en cada contexto, permite generar métodos de solución. (Vásquez Morales 2007). Aunque consumen más tiempo que las heurísticas, las metaheurísticas son capaces de producir consistentemente soluciones de alta calidad (Cordeau et al. 2002).

Desde sus inicios, las metaheurísticas fueron construidas basadas en los principios de búsqueda local y búsqueda de población. Los métodos de búsqueda local, realizan una búsqueda intensiva en el espacio de soluciones, moviéndose desde una solución anterior hacia otra más prometedora, siendo ejemplos de este principio el Recocido Simulado (Kirkpatrick et al. 1983) y la Búsqueda Tabú (Glover 1989, 1990)**.** Por su parte, los métodos de búsqueda de población se centran en obtener y mantener un grupo de "buenos padres" y producir crías a partir de estos, como es el caso los algoritmos genéticos (Holland 1975; Goldberg 1989)**.**

Entre las metaheurísticas más populares se encuentran aquellas que han sido inspiradas por la naturaleza, las cuales están basadas en métodos que representan el éxito en los comportamientos animales y de microorganismos que actúan en grupo, y han sido adaptadas para llevar a cabo tareas de optimización en dominios complejos de datos o información (Marinakis y Marinaki 2010)**.** Algunos de ellos son el enjambre de partículas

o inteligencia congregada, inspirada por bandadas de aves y bancos de peces (Kennedy et al. 2001), los sistemas inmunes artificiales, que imitan los sistemas inmunes biológicos (DasGupta 1993; De Castro y Timmis 2002)**,** el rendimiento optimizado por abejas (Baykasoglu et al. 2007)**,** la optimización por colonia de hormigas, basada en el comportamiento de hormigas forrajeras (Dorigo y Stützle, 2004)**,** y los algoritmos genéticos, inspirados en fenómenos que ocurren en la naturaleza como la reproducción y la mutación (Holland 1975; Goldberg 1989)**.**

Existen algoritmos recientemente desarrollados que pueden obtener buenas soluciones en un tiempo razonable, sin embargo, sigue siendo común encontrar que una infinidad de problemas han sido abordados haciendo uso de versiones mejoradas de los algoritmos más tradicionales. Es tarea de la ciencia considerar nuevas opciones que permitan ampliar la frontera del conocimiento, tomando como referencia campos poco explorados y buscando patrones intrínsecos en la naturaleza que permitan mejorar la calidad y el tiempo de obtención de las soluciones a través de algoritmos de inteligencia artificial.

En esta investigación, se diseñará un algoritmo para solucionar eficientemente un modelo matemático particular de VRP que será formulado en este mismo trabajo: ruteo de vehículos refrigerados y de tipo general para la entrega de productos perecederos.

# CAPÍTULO 3: MATERIALES Y MÉTODOS

En este capítulo, se presenta información sobre el tipo de investigación, sus etapas, el diseño de experimentos y los métodos de análisis. De igual manera, se proporcionan datos relacionados con el tamaño de la muestra, los parámetros de las metaheurísticas utilizadas y la generación de instancias.

## 3.1 METODOLOGÍA

La presente investigación es de tipo correlacional, debido a que se estudia la relación entre los resultados arrojados por los distintos algoritmos y su significancia; con diseño experimental, dado que se tendrá control sobre todas las variables del problema.

A continuación, se muestran las etapas de la investigación:

### 3.1.1 Diagnóstico y contextualización del problema

Consiste en realizar una revisión a priori, para determinar la información disponible y afianzar el estado del arte de la presente investigación.

### 3.1.2 Construcción del marco teórico y conceptual

Luego de reunir información vital para la investigación relacionada con todos los detalles de la misma y sus temas relacionados, así como verificar el estado actual del tema en mención, se procede a consolidar el marco teórico y conceptual de la presente investigación.

### 3.1.3 Determinación de las restricciones del problema

En este punto, se analizan los factores que puedan incidir en la pérdida de frescura de los productos alimenticios transportados, de igual manera, se planea la formulación del modelo, evaluando distintos escenarios y definiendo las posibles restricciones.

### 3.1.4 Desarrollo del modelo matemático

Consiste en la formulación del modelo considerando los aspectos teóricos analizados y las restricciones definidas para el problema.

### 3.1.5 Selección y programación de algoritmos

En esta etapa, se realiza un barrido de la información recolectada y se procede a seleccionar y programar los algoritmos meta heurísticos recomendados por la literatura para solucionar problemas de ruteo de vehículos. Estos algoritmos deberán verificarse para asegurar la factibilidad de las soluciones generadas.

### 3.1.6 Diseño e implementación de una meta heurística que solucione el problema

En este punto, se analizarán los operadores de cada uno de los algoritmos programados, valorando sus características de exploración y explotación. Los primeros propician descubrir regiones sin explorar y evitar la convergencia prematura, mientras que los segundos guían la búsqueda teniendo en cuenta la información que se obtiene de las mejores soluciones encontradas hasta el momento (Vargas 2011)**.** Posteriormente, se construye paso a paso un algoritmo que contenga operadores de ambos tipos y que solucione eficientemente el problema planteado. Se considerará la posibilidad de diseñar e introducir nuevos operadores en esta etapa del proyecto.

### 3.1.7 Diseño experimental simple de comparación

Luego de testear las metaheurísticas tomadas de la literatura y la propuesta, se procede a realizar un experimento de comparación que consiste en la generación de instancias y en la solución de cada instancia en cada metaheurística un número de veces determinado, fijando como constante el tiempo de ejecución, el número de veces que se ejecuta cada instancia y los parámetros entrada de cada metaheurística. Estas serán evaluadas en función del valor obtenido en cada ejecución. Para poder comparar valores de instancias con dimensiones distintas (variarán de acuerdo al número de clientes, a su demanda y a la matriz de tiempos de viaje), será necesario llevar los datos a una distribución estándar.

### 3.1.8 Determinación de la mejor metaheurística para solucionar el problema

Con ayuda de un software estadístico, se verificarán los supuestos y se realizará una tabla ANOVA y prueba de múltiples rangos. En caso de no cumplir con los supuestos, se hará uso de pruebas estadísticas no paramétricas.

Cabe resaltar que, antes de realizar la construcción de un modelo matemático de ruteo de vehículos para minimizar la frescura en múltiples tipos de productos perecederos, se desarrollará un modelo mono producto. Por tal motivo, las etapas de esta investigación serán ejecutadas para el desarrollo y evaluación de ambos modelos, a excepción del diagnóstico y contextualización del problema y la construcción del marco teórico. De acuerdo a esto, se construirán dos modelos matemáticos y se programarán y compararán metaheurísticas para cada uno de ellos.

## 3.2 CONSIDERACIONES GENERALES DEL PROBLEMA

Antes de adentrarnos más en el problema, se hace necesario establecer los aspectos que se tuvieron en cuenta a la hora de formular el modelo.

Entre los productos alimenticios de tipo perecedero que pueden considerarse, es posible realizar una clasificación de acuerdo a qué tan perecibles son estos productos. Aquellos que son altamente perecederos, sólo deben ser transportados por vehículos refrigerados con el fin de mantener las condiciones de seguridad y calidad de los mismos. Por otro lado, existen productos menos perecibles (también conocidos como moderadamente perecederos), los cuales podrían ser transportados por vehículos no refrigerados bajo ciertas condiciones:

- Compatibilidad del producto con las condiciones ambientales de la zona en la que se realiza el transporte (tales como la temperatura y la humedad relativa)
- El uso de un sistema First In, First Out (FIFO) en el centro de distribución
- Diseño del empaque enfocado a la preservación de la frescura del producto
- Cercanía de los centros de producción y de los clientes al centro de distribución.

Para largos viajes, los productos moderadamente perecederos podrían ser transportados por vehículos refrigerados.

Cabe resaltar que los productos perderán frescura a lo largo de la ruta, ya sea por el tiempo transcurrido desde el depósito hasta el cliente o por las aperturas de puerta de los vehículos a través de la ruta.

Así, se busca minimizar la pérdida de frescura a través de la inclusión de este factor en la planificación de las rutas.

Para los modelos matemáticos resultantes en esta investigación, los productos solo podrán ser moderadamente perecederos, teniendo en cuenta que se permitirá el uso de vehículos refrigerados y no refrigerados. En los vehículos refrigerados, la frescura del producto podrá verse afectada principalmente por las aperturas de puerta del vehículo durante la

ruta y, en menor medida, por el tiempo que transcurre desde que el vehículo parte del depósito hasta que llega al cliente. Para los vehículos no refrigerados, la frescura del producto se verá afectada por el tiempo transcurrido desde que el vehículo parte del depósito hasta que llega al cliente y, en menor medida, por las aperturas de puerta, dado que para estos vehículos no se asume una bodega de almacenamiento totalmente cerrada.

## 3.3 MODELACIÓN MATEMÁTICA

Se presentan dos modelos matemáticos basados en el concepto del problema de ruteo de vehículos acumulativo para vehículos capacitados (CCVRP) para medir el tiempo desde que el vehículo sale del depósito hasta un cliente específico (Ngueveu et al. 2010) y en las restricciones presentadas en Song y Ko (2016) relativas a la contabilización del número de aperturas de puerta desde que un vehículo parte del depósito hasta cuando llega a un cliente.

Ambos modelos fueron validados con ayuda del software de optimización GAMS. Esta validación consistió en introducir la función objetivo y cada una de las restricciones en el software. Luego de esto, se realizó una prueba con una instancia pequeña para identificar problemas lógicos en la modelación matemática propuesta y realizar las correcciones pertinentes. A continuación, se analizaron los resultados que arroja el software, observando su coherencia con lo previsto para las variables definidas. Finalmente, se tomó el valor de la función objetivo proporcionado por GAMS para la mejor solución obtenida y se comparó con el valor que se obtiene realizando los cálculos de forma manual para la solución proporcionada por el software.

### 3.3.1 Modelo matemático para un producto perecedero (Caso I)

Para formular la función objetivo, se calculará la pérdida de frescura de un producto perecedero en la ruta, teniendo en cuenta tanto el número de aperturas de puerta como el tiempo de viaje desde el depósito hasta los clientes.

En la figura 1, se muestra el comportamiento de la frescura con respecto al tiempo a gran escala. Si se observa más de cerca la curva de frescura, esta describe rápidos cambios causados por las aperturas de puerta a lo largo de la ruta, como puede verse en la figura 2. Nótese que alfa es la pendiente de la recta, que es la tasa que afecta la frescura del producto con respecto al tiempo transcurrido y beta es la tasa que afecta a la frescura cuando se abre la puerta de la bodega de almacenamiento del vehículo en cada cliente.

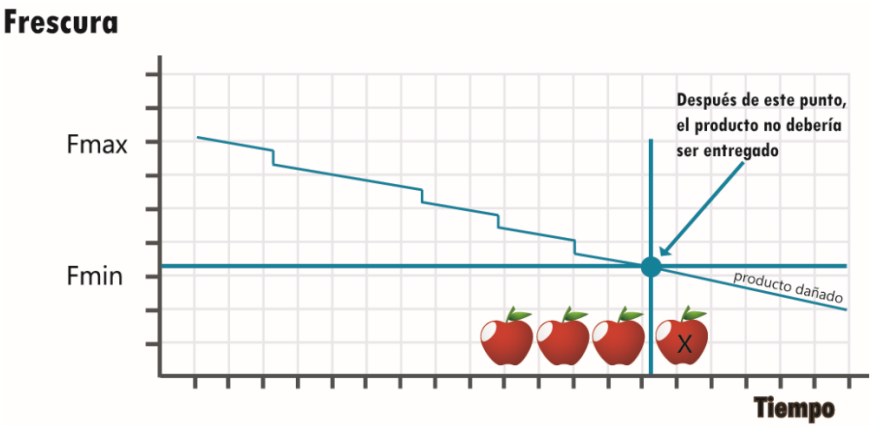



**Figura 1.** Frescura versus tiempo para un producto en una ruta.

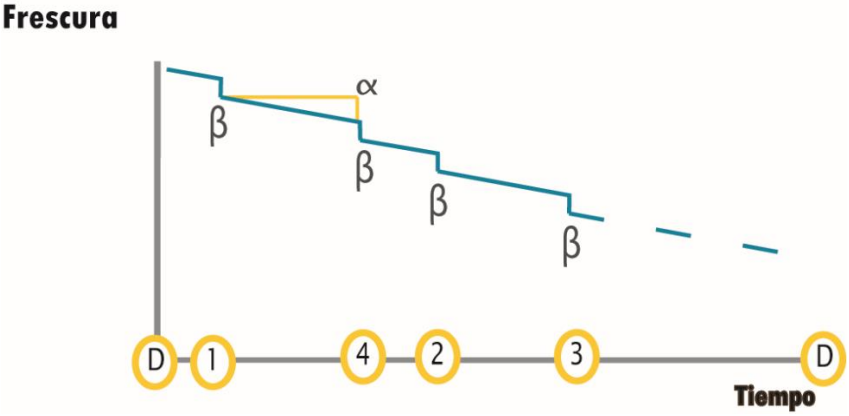



**Figura 2.** Descripción gráfica de la incidencia de alfa y beta en la frescura

El valor de las tasas de reducción de frescura alfa y beta estará ligado al tipo de vehículo que recorra la ruta, a qué tan perecedero es el producto y a las condiciones en las que es transportado.

Si se considera un valor de frescura inicial del producto que va decreciendo a través de la ruta, la función objetivo que maximiza la frescura podría incluir los parámetros alfa y beta. Ahora bien, al orientar la función objetivo a minimizar la reducción de frescura en lugar de maximizar esta variable, no es necesario contar con un valor de frescura inicial. De esta manera, se construye un modelo que minimiza la reducción de frescura del producto transportado en función de alfa y beta. A continuación, se muestran los conjuntos, parámetros y variables del modelo:

***Conjuntos*:**
$V$: Conjunto de nodos *(i, j)* $\in \{0, \ldots, n\}$
$V'$: Conjunto de nodos *(i, j)* $\in V \setminus \{0\}$
$R$: Conjunto de vehículos $k \in \{1, \ldots, K\}$

***Parámetros:***
$t_{ij}$: Matriz de tiempos de viaje entre los nodos *i e j.* $((i,j) \in V)$
$d_i$: Demanda del cliente *i.* $(i \in V')$
$s_i$: Tiempo de servicio en el cliente i. $(i \in V')$
$C_k$:Capacidad del vehículo *k.* $(k \in R)$
$\propto^k$: Tasa de reducción de frescura por unidad de tiempo recorrida cuando el producto es transportado por el vehículo *k.* $(k \in R)$
$\beta^k$: Tasa de reducción de frescura asociada a la apertura de puertas cuando el producto es transportado por el vehículo *k.* $(k \in R)$
$K$: Tamaño de la flota
$M$: Número positivo muy grande.

***Variables:***
$Z$: Pérdida de frescura total
$t_i^k$: Tiempo de llegada del vehículo *k* desde el depósito hasta el cliente *i.*
$na_{ik}$:Número de aperturas de la puerta de almacenamiento del vehículo *k* desde el depósito hasta el cliente *i.*
$x_{ijk}$: Variable de decisión binaria que es *1* si el vehículo *k* viaja desde el nodo *i* hasta el nodo *j*

$y_{ik}$: Variable binaria que es *1* si el cliente *i* es atendido por el vehículo *k*
$L_{ik}$: Carga del vehículo *k* cuando deja el nodo *i*.

El problema consiste en encontrar una colección de, a lo sumo, *K* circuitos simples con mínima pérdida de frescura, tal que:

i. *Cada circuito visita el depósito (nodo 0);*

ii. *Cada cliente es visitado exactamente por un circuito;*

iii. *La suma de las demandas de los clientes visitados por el vehículo k no excede su capacidad Ck;*

iv. *Para cada cliente i, el servicio comienza en el tiempo* $t_i^k$*, y el vehículo se detiene por un tiempo* $s_i$.

v. *El producto transportado puede entregarse en cualquiera de los vehículos de la flota R, es decir, tanto en vehículos refrigerados como en no refrigerados*

A partir de las premisas anteriores, se formuló el siguiente modelo matemático:

$$\text{Min(Z)} = \sum_{k\in R} \propto^k \sum_{i\in V} \sum_{j\in V'} t_j^k \, x_{ijk} + \sum_{k\in R} \beta^k \sum_{i\in V} \sum_{j\in V'} na_{jk}.x_{ijk} \quad (1)$$

Restricciones:

$$\sum_{j\in V} x_{jik} = \sum_{j\in V} x_{ijk}, \forall i \in V, k \in R \quad (2)$$

$$\sum_{k\in R} \sum_{j\in V} x_{0jk} = \sum_{k\in R} \sum_{i\in V} x_{i0k} \quad (3)$$

$$\sum_{k\in R} \sum_{i\in V} x_{i0k} \leq K \quad (4)$$

$$\sum_{j\in V} x_{jik} = y_{ik}, \forall i \in V, k \in R \quad (5)$$

$$\sum_{k\in R} y_{ik} = 1, \forall i \in V' \quad (6)$$

$$1 - na_{jk} \leq M(1 - x_{0jk}), \forall j \in V', k \in R \quad (7)$$

$$na_{ik} + 1 - M(1 - x_{ijk}) \leq na_{jk}, \forall (i,j) \in V', k \in R \quad (8)$$

$$t_{0j} - t_j^k \leq M\left(1 - x_{0jk}\right), \forall j \in V', k \in R \quad (9)$$

$$t_i^k + s_i + t_{ij} - M\left(1 - x_{ijk}\right) \leq t_j^k, \forall (i,j) \in V', k \in R \quad (10)$$

$$d_j - M\left(1 - x_{0jk}\right) \leq L_{jk}, \forall j \in V', k \in R \quad (11)$$

$$L_{ik} + d_j - M\left(1 - x_{ijk}\right) \leq L_{jk}, \forall (i,j) \in V', k \in R \quad (12)$$

$$L_{jk} \leq C_k, \forall j \in V', k \in R \quad (13)$$

$na_{ik}$ es un entero positivo, $\forall_i \in V'$ (14)

$$L_{ik} \geq 0, \forall i \in V', k \in R \quad (15)$$

$$t_i^k \geq 0, \forall i \in V', k \in R \quad (16)$$

$$x_{ijk} \in \{0,1\}, \forall (i,j) \in V, i \neq j, k \in R \quad (17)$$

$$y_{ik} \in \{0,1\}, \forall i \in V', k \in R \quad (18)$$

En la función objetivo (1), la expresión que acompaña a la sumatoria de $\propto^k$, calcula la suma de tiempos transcurridos desde el deposito hasta cada uno de los clientes, mientras que la expresión que acompaña a la sumatoria de $\beta^k$ calcula la suma acumulada de aperturas de puerta en todas las rutas. Las ecuaciones (2) a (6) son relativas a la conservación del flujo a través de la red, al número máximo de vehículos disponibles de ambos tipos y a la asignación de vehículos por cliente. (7) y (8) calculan y regulan el conteo acumulativo de aperturas de puerta por cliente. (9) y (10) calculan el tiempo transcurrido desde el depósito hasta los clientes, además de regular la aparición de subtours (Ngueveu et al. 2010). Por su parte, (11), (12) y (13) son restricciones de capacidad y (14) a (18) obedecen a restricciones de integralidad y no negatividad de las variables del problema.

### 3.3.2 Modelo matemático para múltiples productos perecederos (Caso II)

Para la entrega de múltiples productos perecederos, se mantienen los conceptos de pérdida de frescura establecidos en el modelo anterior (figura 1 y 2), sin embargo, se incluye en la función objetivo la demanda de los clientes para priorizar la entrega de acuerdo al volumen de producto y a su pérdida de frescura. También, alfa y beta serán reemplazados por theta y psi, para efectos de comprensión e identificación entre los dos modelos. A continuación, se muestran los conjuntos, parámetros y variables definidos:

**Conjuntos:**
$V$: Conjunto de nodos *(i, j)* $\in \{0, \ldots, n\}$
$V'$: Conjunto de nodos *(i, j)* $\in V \setminus \{0\}$
$R$: Conjunto de vehículos $k \in \{1, \ldots, K\}$
$P$: Conjunto de productos $p \in \{1, \ldots, m\}$

***Parámetros:***
$t_{ij}$*:* Matriz de tiempos de viaje entre los nodos *i e j.* $((i,j) \in V)$
$d_{jp}$: Demanda del cliente *j* del producto *p.* $(j \in V', p \in P)$
$s_i$: Tiempo de servicio en el cliente *i.* $(i \in V')$
$C_k$: Capacidad en volumen del vehículo *k.* $(k \in R)$
$V_p$: Volumen del producto *p.* $(p \in P)$
$\theta_p^k$: Tasa de reducción de frescura del producto *p* por unidad de tiempo recorrida cuando el producto es transportado por un vehículo *k.* $(p \in P, k \in R)$
$\psi_p^k$: Tasa de reducción de frescura asociada a la apertura de la bodega de almacenamiento cuando el producto p es transportado por un vehículo *k.* $(p \in P, k \in R)$
$K$: Tamaño de la flota
$M$: Número positivo muy grande.

***Variables:***
$Z$: Pérdida de frescura total
$t_i^k$: Tiempo de llegada del vehículo k desde el depósito hasta el cliente i.
$na_{ik}$: Número de aperturas de la puerta de almacenamiento del vehículo k desde el depósito hasta el cliente *i.*
$x_{ijk}$*:* Variable de decisión binaria que es *1* si el vehículo *k* viaja desde el nodo *i* hasta el nodo *j*
$y_{ik}$*:* Variable binaria que es *1* si el cliente i es atendido por el vehículo *k*
$L_{ik}$*:* Carga del vehículo *k* cuando deja el nodo *i.*

En este caso, se deben diseñar, como máximo, *K* circuitos simples con mínima pérdida de frescura, tal que:

i. *Cada circuito visita el depósito (nodo 0);*

ii. *Cada cliente es visitado exactamente por un circuito;*

iii. *La suma de los volúmenes de producto requeridos por los clientes visitados en un circuito no excede la capacidad en volumen Ck del vehículo k;*

iv. *Para cada cliente i, el servicio comienza en el tiempo* $t_i^k$*, y el vehículo se detiene por un tiempo* $s_i$.

v. *Los productos transportados pueden ser entregados en ambos tipos de vehículos (refrigerados y no refrigerados), que hacen parte de la flota R.*

La función objetivo se construye buscando minimizar la pérdida de frescura con prioridad de demanda, es decir, poniendo énfasis en el factor demanda en el proceso de ruteo. Además, se tiene en cuenta la pérdida de frescura por aperturas de puerta y la pérdida de frescura por tiempo trascurrido, estableciendo tasas de pérdida de acuerdo al tipo de producto y al tipo de vehículo, que puede ser refrigerado o no refrigerado. La modelación de esta ecuación se divide en 2 partes:

*Parte 1:*

$$\mathrm{Min}(Z_1) = \sum_{k\in R}\sum_{i\in V}\sum_{j\in V'} t_j^k \, x_{ijk} \sum_{p\in P} \theta_p^k d_{jp} \tag{19}$$

La parte 1, incluye el tiempo transcurrido cuando el vehículo *k* visita al cliente *j*, la demanda del cliente *j* del producto *p* y la tasa de reducción de frescura asociada a cada producto por unidad de tiempo.

*Parte 2:*

$$\mathrm{Min}(Z_2) = \sum_{k\in R}\sum_{i\in V}\sum_{j\in V'} na_{jk} \, x_{ijk} \sum_{p\in P} \psi_p^k d_{jp} \tag{20}$$

La segunda parte, abarca el número de aperturas de puerta acumulado cuando el vehículo k visita al cliente j (inclusive), la demanda del cliente j del producto p y la tasa de reducción de frescura asociada al producto por aperturas de puerta. A continuación, se muestra la ecuación que encierra ambas partes:

$$\mathrm{Min}(Z_1 + Z_2) = \sum_{k\in R}\sum_{i\in V}\sum_{j\in V'} x_{ijk} \sum_{p\in P} d_{jp}\left(t_j^k \theta_p^k + na_{jk}\psi_p^k\right) \tag{21}$$

A continuación, se presenta el modelo matemático para este caso:

$$\mathrm{Min}(Z) = \sum_{k\in R}\sum_{i\in V}\sum_{j\in V'} x_{ijk} \sum_{p\in P} d_{jp}\left(t_j^k \theta_p^k + na_{jk}\psi_p^k\right) \tag{22}$$

Sujeto a:

$$\sum_{j\in V} x_{jik} = \sum_{j\in V} x_{ijk}, \forall i \in V, k \in R \tag{23}$$

$$\sum_{k\in R}\sum_{j\in V} x_{0jk} = \sum_{k\in R}\sum_{i\in V} x_{i0k} \tag{24}$$

$$\sum_{k\in R}\sum_{i\in V} x_{i0k} \leq K \tag{25}$$

$$\sum_{j\in V} x_{jik} = y_{ik}, \forall i \in V, k \in R \tag{26}$$

$$\sum_{k\in R} y_{ik} = 1, \forall i \in V' \tag{27}$$

$$1 - na_{jk} \leq M\left(1 - x_{0jk}\right), \ \forall j \in V', k \in R \tag{28}$$

$$na_{ik} + 1 - M\left(1 - x_{ijk}\right) \leq na_{jk}, \forall (i,j) \in V', k \in R \tag{29}$$

$$t_{0j} - t_j^k \leq M\left(1 - x_{0jk}\right), \ \forall j \in V', k \in R \tag{30}$$

$$t_i^k + s_i + t_{ij} - M\left(1 - x_{ijk}\right) \leq t_j^k, \forall (i,j) \in V', k \in R \tag{31}$$

$$\sum_{p\in P} d_{jp}V_p - M\left(1 - x_{0jk}\right) \leq L_{jk}, \forall j \in V', k \in R \tag{32}$$

$$L_{ik} + \sum_{p\in P} d_{jp}V_p - M\left(1 - x_{ijk}\right) \leq L_{jk}, \ \forall (i,j) \in V', k \in R \tag{33}$$

$$L_{jk} \leq C_k, \ \forall j \in V', k \in R \tag{34}$$

$$na_{ik}\ es\ un\ entero\ positivo, \forall_i \in V', k \in R \quad (35)$$

$$L_{ik} \geq 0, \forall i \in V', k \in R \quad (36)$$

$$t_i^k \geq 0, \forall i \in V', k \in R \quad (37)$$

$$x_{ijk} \in \{0,1\},\ \forall (i,j) \in V, i \neq j, k \in R \quad (38)$$

$$y_{ik} \in \{0,1\}, \forall i \in V', k \in R \quad (39)$$

En la función objetivo (22), se calcula la pérdida de frescura total de acuerdo a las ecuaciones 19 y 10. Las ecuaciones (23) a (27) son relativas a la conservación del flujo a través de la red, al número máximo de vehículos disponibles de ambos tipos y a la asignación de vehículos por cliente. (28) y (29) calculan y regulan el conteo acumulativo de aperturas de puerta por cliente. (30) y (31) calculan el tiempo transcurrido desde el depósito hasta los clientes, además de regular la aparición de subtours. Por su parte, (32), (33) y (34) son restricciones de capacidad y (35) a (39) obedecen a restricciones de integralidad y no negatividad de las variables del problema.

### 3.4 DESARROLLO DE ALGORITMOS

A partir de la revisión bibliográfica, se definieron cuatro metaheurísticas para la solución del caso 1: algoritmo genético (AG), la optimización por cúmulo de partículas (PSO), el algoritmo cromático (AC) y un algoritmo propuesto (N1) que combina dos metaheurísticas: genético y cromático.

Por otra parte, para el caso 2, fueron escogidas 2 metaheurísticas del primero caso (algoritmo genético y algoritmo cromático) y se propuso una nueva metaheurística (N2) para la solución del problema.

La función de evaluación para ambos casos, las metaheurísticas seleccionadas y los nuevos algoritmos propuestos, fueron programados en el software MATLAB.

### 3.4.1 Encoding

Es importante definir la forma en que se codifican las soluciones en cada una de las metaheurísticas programadas. Para el caso 1, las soluciones son representadas como se muestra en la figura 3 y la codificación es de tipo permutada. En este caso, las soluciones se programaron de tal manera que cada una de ellas es un vector que representa el orden en que los clientes son asignados de derecha a izquierda a un vector de fijo de vehículos que va desde los vehículos de menor capacidad hasta los vehículos de mayo capacidad. Una solución es modificada cuando se implementa un operador que realiza un cambio en el orden en que serán asignados los clientes al vector de vehículos. Se muestra el vector que representa una solución para un problema con seis clientes (figura 3).

| | | | | | | |
|---|---|---|---|---|---|---|
| 2 | 7 | 1 | 3 | 6 | 4 | 5 |

**Figura 3.** Representación de una solución. Caso 1 - Entrega de un producto perecedero.

Para la entrega de múltiples productos perecederos (caso 2), las soluciones son codificadas de forma distinta. Cada solución está compuesta por dos vectores con una cantidad de elementos igual al número de clientes. La primera parte de la solución es un vector permutado y representa a los clientes. Este vector será afectado de acuerdo a los cambios que se realicen a partir de la implementación de operadores de los distintos algoritmos.

La segunda parte contiene índices de los vehículos que se asignarán a los clientes, este vector no es permutado y se genera a partir de número aleatorios enteros entre uno y el número de vehículos de la flota. Las rutas se construyen a partir de las parejas cliente – vehículo. En caso de que una de las soluciones generadas no sea factible, se repara la

solución generando un nuevo vector de asignación de vehículos. Se muestra una solución para un problema con siete clientes y una flota de cuatro vehículos (figura 4).

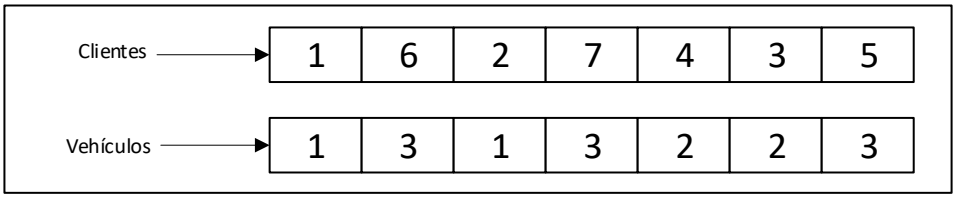



**Figura 4.** Representación de una solución. Caso 2 - Entrega multi producto.

Nota: La segunda parte de la solución no es modificada por los operadores de las distintas metaheurísticas implementadas.

### 3.4.2 Algoritmo genético (AG)

En un algoritmo genético, se presenta la evolución de una población de cadenas de bits, o cromosomas, donde cada cromosoma representa una solución codificada para una instancia en particular. La evolución cromosómica ocurre gracias a la aplicación de operadores como la reproducción y la mutación, que imitan el fenómeno observado en la naturaleza (Toth, P. y Vigo 2002).

Para esta metaheurística, es común encontrarnos con las siguientes fases: la generación de la población inicial, la selección de padres, el operador de cruzamiento, el operador de mutación y el operador de reemplazo de cada generación. Existen varios métodos que pueden ser usados para completar un algoritmo genético, uno de ellos consiste en observar cada individuo por separado, evitando que este sea influenciado o tenga alguna interacción con los demás individuos de la población, esto es, haciendo uso de estrategias más complejas de búsqueda local. Otra forma consiste en propiciar la interacción entre los individuos (Marinakis y Marinaki 2010). En la figura 5, se muestra un diagrama de flujo que describe cómo opera esta metaheurística.

El algoritmo evolutivo empleado ha sido adecuado para mejorar su rendimiento, aplicando el cruzamiento mejorado SB2OX (Ruiz, Maroto, y Alcaraz 2005)**.** Por su parte, la probabilidad de cruzamiento (Pc) y probabilidad de mutación (Pm) se han ajustado a partir de una cantidad razonable de pruebas con porcentajes propuestos aleatoriamente. Una característica de este algoritmo es su buen desempeño en la exploración del conjunto de soluciones a través de la búsqueda global.

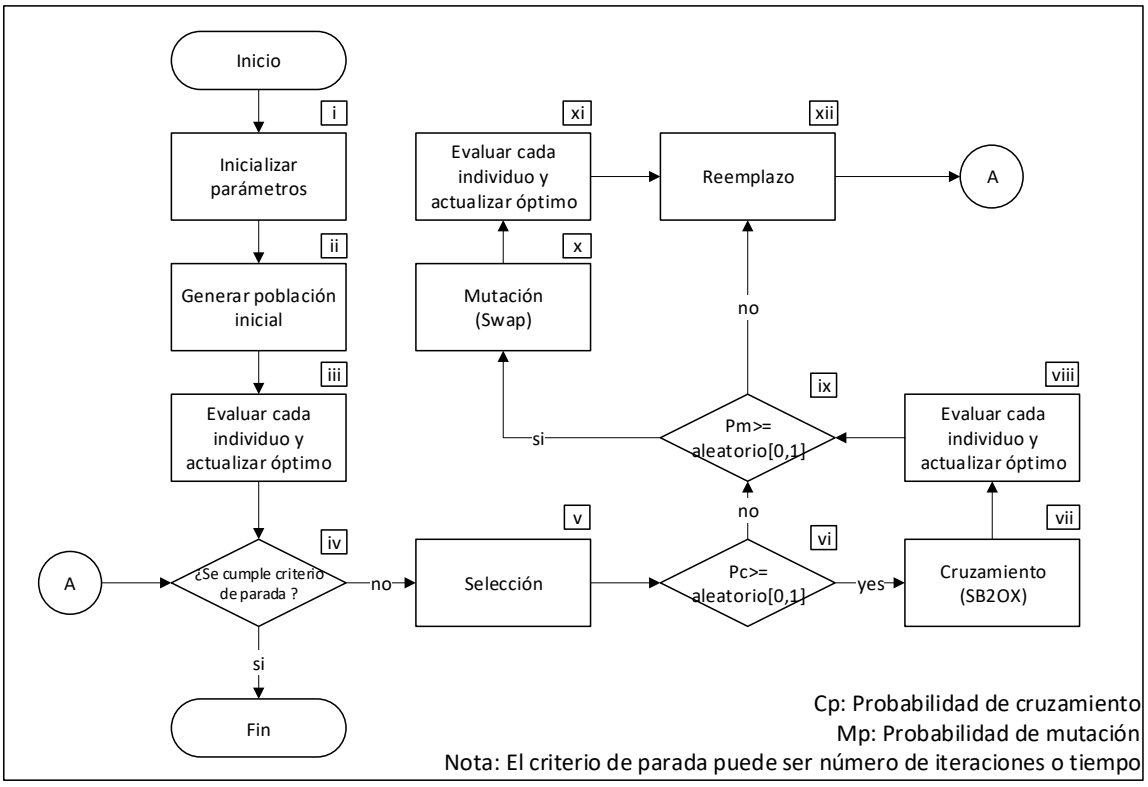



**Figura 5.** Diagrama de flujo del algoritmo genético

*i.* *Inicializar parámetros.* Se comprueban los parámetros del algoritmo (criterio de parada, probabilidad de cruzamiento, probabilidad de mutación y tamaño de la población) y los parámetros del problema.

*ii.* *Generar población inicial.* Se genera una población de soluciones de acuerdo al tamaño definido en el paso anterior.

*iii.* *Evaluar cada individuo y actualizar óptimo.* Se toma cada una de las soluciones generadas y se evalúa de acuerdo a la función objetivo del problema y a los parámetros definidos en el paso *i.*

*iv.* *¿Se cumple criterio de parada?* Se evalúa el cumplimiento del criterio de parada definido en el paso *i,* que puede ser número de iteraciones o tiempo. Si el criterio se cumple, el algoritmo finaliza su ejecución, de lo contrario, se avanza al paso *v.*

*v.* *Selección.* Se realiza una selección por torneo, escogiendo 4 individuos de la población inicial de forma aleatoria, evaluándolos y escogiendo simultáneamente dos de ellos de acuerdo al valor resultante de su evaluación. En la selección se forma la matriz de "padres".

*vi.* *Pc >= aleatorio [0,1]*. Se genera un número aleatorio entre 0 y 1 para cada par de individuos de la población resultante del *v*. Si *Pc* es mayor que el número generado, el par de "padres" pasan al paso *vii* para realizar el cruzamiento, de lo contrario, el par de "hijos" es una réplica de los "padres", se procede al paso *viii.*

*vii.* *Cruzamiento.* Se aplica el cruzamiento mejorado SB2OX propuesto en el trabajo de (Ruiz et al., 2005)**.** En este operador, los "padres" se cruzan para generar nuevos individuos, formando una matriz de "hijos".

*viii.* *Evaluar cada individuo y actualizar óptimo.* Se evalúan los individuos (soluciones) resultantes del paso *vii.* Ver paso *iii.*

*ix.* *Pm >= aleatorio [0,1]*. Se genera un número aleatorio entre 0 y 1 para cada "hijo". Si *Pm* es mayor que el número generado, el individuo muta (ver paso *x*), de lo contrario, se procede al paso *xii.*

*x.* *Mutación.* La mutación se realiza a través de la selección y el intercambio de dos elementos de un individuo de forma aleatoria.

*xi.* *Evaluar cada individuo y actualizar óptimo.* Se evalúan los individuos (soluciones) resultantes del paso *x.* Ver paso *iii.*

*xii.* *Reemplazo.* En este paso, la población de "hijos" se convierte en la siguiente generación. Se procede al paso *iv.*

### 3.4.3 Algoritmo de optimización por enjambre de partículas (PSO)

Este algoritmo consiste en un conjunto de partículas en movimiento (o enjambre) que es inicialmente "lanzado" dentro del espacio de búsqueda. Cada partícula posee una posición y una velocidad, conoce su posición y el valor de la función objetivo para esta posición, conoce a sus vecinos, recuerda la mejor posición anterior y el valor de la función objetivo de la misma (Clerc 2004).

Originalmente, el PSO (Particle Swarm Optimization) fue desarrollado para la optimización en un dominio continuo. Para esta aplicación, ha sido necesario implementar una adaptación de este algoritmo basada en optimización discreta para codificación permutada propuesta por Nieto (2006), la cual se basa en las especificaciones de Clerc (2004). Cabe resaltar que este algoritmo no es especialmente competitivo para problemas de optimización discreta, sin embargo, vale la pena compararlo para evaluar su desempeño frente a otras metaheurísticas. Para este tipo de codificación, se utilizan los siguientes objetos y operaciones matemáticas:

- Posición de una partícula
- Velocidad de una partícula
- Resta de posiciones ($\ominus$): La sustracción de dos posiciones produce una velocidad

- Multiplicación externa ($\otimes$): La multiplicación externa de un número real con una velocidad produce otra velocidad
- Suma ($\bigcirc$): La suma de dos velocidades produce una nueva velocidad
- Movimiento ($\oplus$): Si se opera una posición y una velocidad, se obtiene una nueva posición

Para entender a detalle cómo se realizan las operaciones anteriormente descritas, puede consultarse el trabajo desarrollado por Nieto (2006). En la figura 6, se observa un diagrama de flujo que describe la operación del algoritmo.

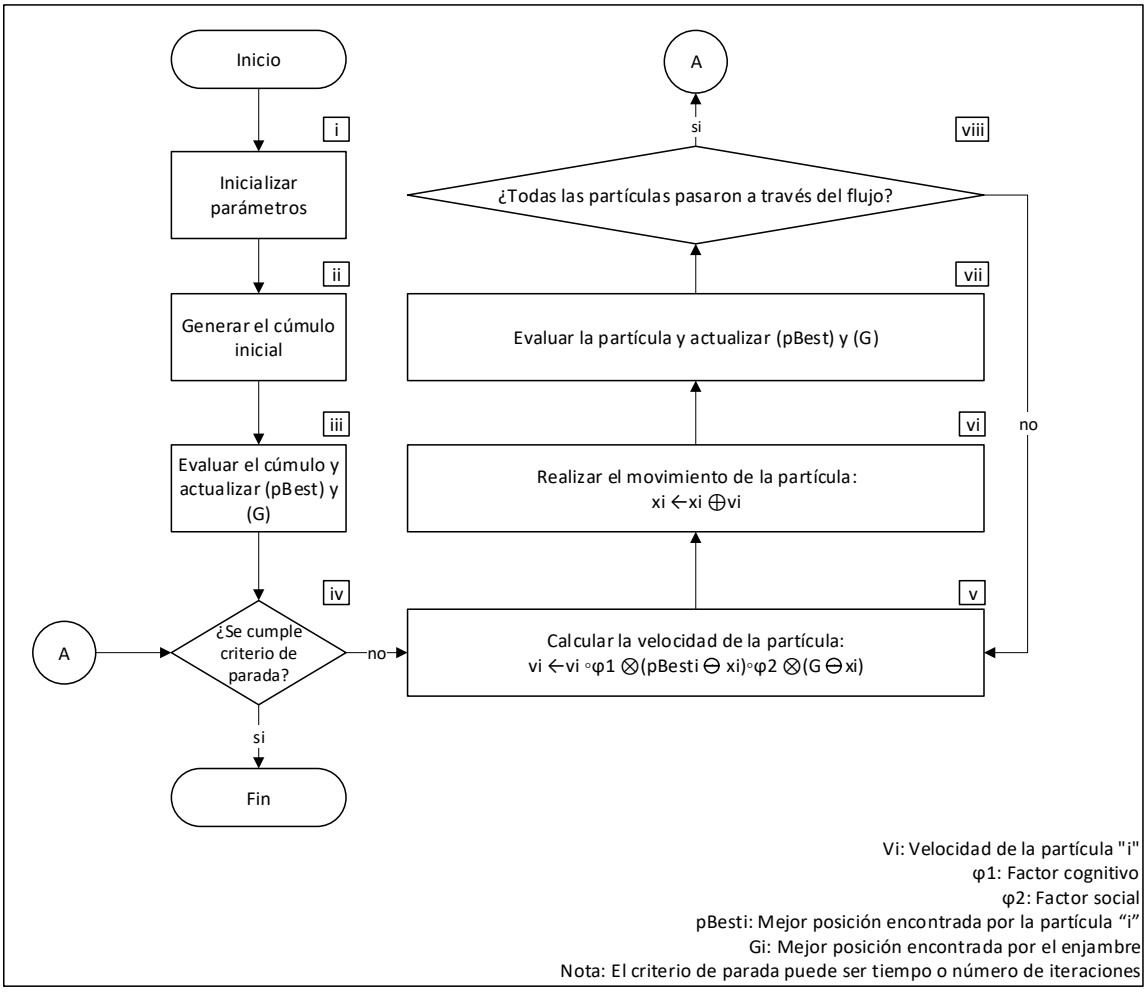



**Figura 6.** Diagrama de flujo del algoritmo PSO.

*i.* *Inicializar parámetros.* La metaheurística comprueba sus parámetros (criterio de parada, tamaño del cúmulo, factor cognitivo y factor social) y los parámetros asociados al problema.

*ii.* *Generar cúmulo inicial.* Se genera aleatoriamente un grupo de soluciones iniciales de acuerdo al parámetro "tamaño de cúmulo" del paso *i.*

*iii.* *Evaluar cúmulo y actualizar pBest y G.* Se evalúan todas las partículas del cúmulo y se actualiza la matriz de mejores posiciones encontradas por cada partícula (*pBest*) y la mejor posición encontrada (*G*).

*iv.* *¿Se cumple criterio de parada?* Si el criterio de parada se cumple, el algoritmo finaliza su ejecución. De lo contrario, se avanza al paso *v.*

*v.* *Calcular velocidad de la partícula.* Previa generación de una velocidad inicial para cada partícula, se calcula una nueva velocidad a partir de la posición de la partícula ($x_i$), de la mejor posición encontrada por la partícula ($pBest_i$), de la mejor posición encontrada por el cúmulo (*G*) y de los factores cognitivo ($\varphi 1$) y social ($\varphi 2$). Luego de esto, se realiza el paso *vi.*

*vi.* *Realizar movimiento de la partícula.* Se implementa el operador de movimiento, que consiste aplicar una velocidad a una partícula. La velocidad representa una serie de intercambios en una solución.

*vii.* *Evaluar la partícula y actualizar (pBest) y (G).* En este punto, se realiza lo propuesto en el paso *iii*, pero de forma individual.

*viii.* *¿Todas las partículas pasaron a través del flujo?* Se comprueba que todas las partículas hayan pasado por el flujo. Si ya pasaron todas, se procede al paso *iv*, de lo contrario, se continua el proceso con la siguiente partícula hasta completar el cúmulo.

### 3.4.4 Algoritmo cromático (AC)

El algoritmo cromático es una metaheurística desarrollada por (Sabie y Mestra 2011)**,** un nuevo método de optimización combinatoria inspirado y relacionado con las distintas formas con las que un músico o artista realiza y escoge la melodía más apropiada para una pieza musical, a partir de la combinación de notas musicales de la escala cromática. La idea principal del método radica en que el músico elige la mejor melodía, la cual es la que mejor se adapta y representa apropiadamente una tonada específica. Esta metaheurística se caracteriza porque no necesita de un gran tamaño en el grupo de melodías que se tiene como opción de búsqueda, es decir, que no necesita de un gran número de soluciones iniciales para obtener buenos resultados.

Esta metaheurística fue diseñada inicialmente para resolver problemas de optimización en un dominio continuo de datos. Se han realizado dos aplicaciones de su versión inicial: en el problema de pronósticos de series de tiempo (Sabie y Lopez 2011) y en el problema de secuenciación de proyectos con recursos limitados (RCPSP) (Hernández y Poveda 2014)**.** En aras de realizar optimización discreta haciendo uso de esta herramienta, se hizo necesario adaptar cada uno de sus operadores a codificación permutada, intentando mantener la esencia del algoritmo cromático. Una adaptación de este algoritmo fue desarrollada para resolver el problema de asignación de horarios en la Universidad de Córdoba, Colombia (Arrieta y López 2013). En el presente trabajo, se ha desarrollado una nueva adaptación de este algoritmo, la cual incluye algunas de las consideraciones de la adaptación realizada por Arrieta y López (2013). A continuación, se describen algunos conceptos, así como los parámetros y operadores requeridos para su funcionamiento.

### 3.4.4.1 Escala cromática

La base fundamental de este algoritmo gira en torno a esta escala, ya que gracias a ella se logra realizar una nueva selección de vecinos entre los valores más próximos a cada variable en cuestión. La idea es inspeccionar los espacios más reducidos de cada segmento de la recta de los números reales, con el fin de obtener mejores resultados. Con el fin de adaptar el operador a una codificación permutada, se trabaja con números naturales. La escala cromática de las notas musicales se caracteriza principalmente por tener medio tono de distancia entre una nota y otra, lo que hace que esta cumpla con unas distancias específicas y definidas para cada nota.

| C | Db | D | Eb | E | F | Gb | G | Ab | A | Bb | B |
|---|---|---|---|---|---|---|---|---|---|---|---|

**Figura 7.** Escala cromática musical general

En esta escala, a medida que se incrementa o se sube en el orden de notas respectivo, también se aumenta el grado, lo que hace que existan notas de igual nombre, pero con distinto sonido, en otras palabras, se tendrían notas "altas" o sonidos agudos a la derecha de la escala debido a que están en un grado mayor y notas "bajas" o sonidos graves a la izquierda de la escala.

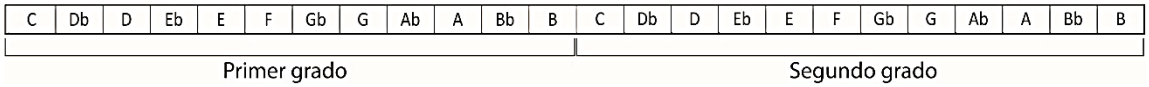



**Figura 8.** Primer y segundo grado de la escala cromática

Como se muestra anteriormente (figura 8), la escala sigue aumentado su grado a medida que se avanza a través de esta, por lo que se considera que existe un número infinito de notas y de grados respectivos. Dado que se puede realizar un acenso infinito en las notas

de la escala, se encontró que tiene una equivalencia con la escala infinita de los números naturales, lo que indica que a una nota de un grado específico se le podría asignar una serie de números naturales correspondientes. Cabe resaltar que la escala cromática cuenta con 12 notas, lo cual justifica el uso de este número en algunos operadores de este algoritmo.

#### 3.4.4.2 Grupo de melodías iniciales

En este algoritmo se genera un grupo de melodías que constituirán las soluciones iniciales del algoritmo, estas se generan de manera aleatoria, respetando su carácter permutado.

#### 3.4.4.3 Arranque múltiple (ARRAM)

Para que una metaheurística salga de óptimos locales, es de gran importancia tener distintas estrategias y métodos que diversifiquen las características de las soluciones con las que se está trabajando, el algoritmo cromático realiza esta diversificación a través de arranques múltiples (ARRAM).

Básicamente, cada vez que se presenta un ARRAM, se generan nuevas soluciones iniciales con el fin de combinar diferentes características de nuevas melodías con la mejor melodía obtenida hasta el momento. Cabe resaltar, que una iteración implica que cada melodía sea tratada por varios de los operadores del algoritmo según las probabilidades de aceptación en cada caso, como podremos ver más adelante.

#### 3.4.4.4 Arreglo melódico (AM)

El arreglo melódico es un ajuste en las melodías para realizar la búsqueda de la MEJOR MELODÍA (MM), este procedimiento se realiza para cada una de las melodías del grupo total que se tiene en la búsqueda. En este operador se utiliza el parámetro ARREGLO MELÓDICO (AM). Si la probabilidad del parámetro (AM) es mayor que un número

aleatorio entre cero y uno proveniente de una distribución uniforme, hay lugar a un arreglo melódico.

### 3.4.4.5 Utilización de la mejor melodía (UM)

Cuando se cumple la probabilidad de ARREGLO MELÓDICO (AM), se utiliza el parámetro de UTILIZACIÓN DEL MEJOR (UM) que se rige bajo una probabilidad de utilización, lo que implica que, si la probabilidad de utilización UM es mayor que un número aleatorio entre cero y uno proveniente de una distribución uniforme, se utiliza la mejor melodía. Esta utilización consiste en que, para una melodía i del grupo de melodías iniciales, se reemplaza el valor de la posición j por el valor que se encuentra en la posición j de la MEJOR MELODÍA (MM), que es la mejor solución global encontrada hasta el momento, y se repara la solución i buscando en ella la posición en la que se encontraba inicialmente el valor en j de la MEJOR MELODÍA (MM) y reemplazando este valor con el valor que en principio estaba en la posición j de la melodía i. Esto puede entenderse mejor con el ejemplo ilustrado en la figura 9**.** Supongamos una melodía i con 12 posiciones y un valor j = 1.

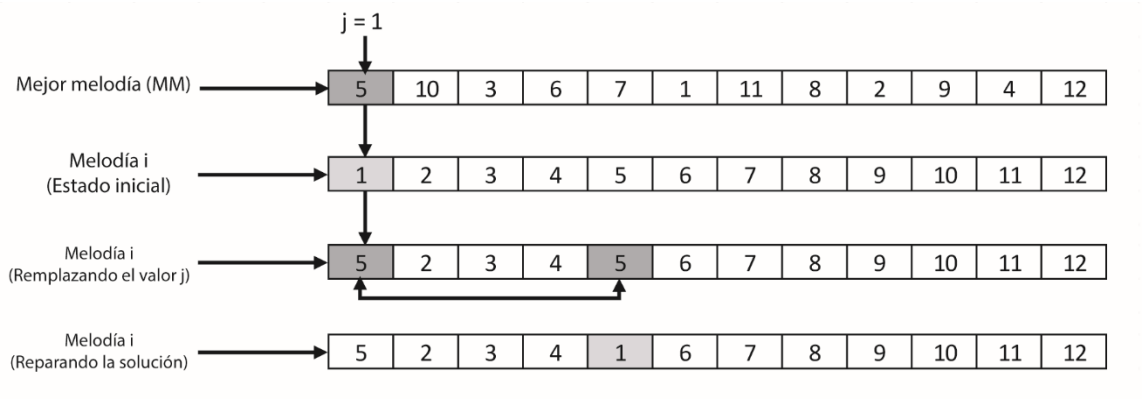



**Figura 9.** Utilización de la mejor melodía

Si se da el caso contrario, es decir, que el número aleatorio sea mayor que el parámetro UM, entonces al valor de la posición j de la melodía i se le realizara una de BÚSQUEDA DE VECINOS DE ESCALA CROMÁTICA.

Este procedimiento se realiza para todas las posiciones del vector que representa cada melodía.

**3.4.4.6 Búsqueda de vecinos de escala cromática**

En este operador se realiza una búsqueda por intervalos con una distancia de 12 unidades, realizando una equivalencia de la recta numérica con la escala cromática. Cada grado consta de 12 notas, por lo que los potenciales vecinos de cada elemento j de una solución serán aquellos números naturales que se encuentren dentro del mismo grado.

A continuación, se muestran los pasos que se realizan en la búsqueda de vecinos de escala cromática para cada elemento de una solución.

**Paso 1.** Calcular el valor del grado de la nota $(\delta)$ para el número natural en la posición j de un vector de solución dado haciendo uso de la ecuación 40.

$$\delta = \left[\!\left[\frac{numero\ natural}{12}\right]\!\right] + 1 \tag{40}$$

**Paso 2.** Calcular el intervalo con base en el grado de la nota $(\delta)$, haciendo uso de las ecuaciones 41 y 42.

$$L\ inf: Limite\ inferior$$

$$L\ sup: Limite\ superior$$

$$k: Número\ de\ posiciones\ de\ la\ solución$$

$$L\ inf = (12)\delta - 12 \tag{41}$$

$$L\ sup = L\ sup + 12 \tag{42}$$

$$L\ inf \geq 1 \tag{43}$$

$$L\ sup \leq k \tag{44}$$

Para casos en los que se incumpla la ecuación 43, el límite inferior será igual a 1. De manera análoga, si se calcula un límite superior que esté por encima del número k, este límite será el número de posiciones de la solución o número de clientes en el problema de ruteo, de conformidad con la ecuación 44.

**Paso 3.** Dividir en intervalo calculado en dos intervalos más pequeños.

$$Intervalo\ 1 \rightarrow \left(L\ inf, L\ inf + \frac{L\ sup - l\ inf}{2}\right), \tag{45}$$

$$Intervalo\ 2 \rightarrow \left(L\ inf + \frac{L\ sup - l\ inf}{2}\ , L\ sup\right), \tag{46}$$

En este punto, los intervalos se dividen entre dos para realizar una búsqueda más minuciosa (ver ecuaciones 45 y 46).

**Paso 4.** Si el valor j de la posición de una solución al que se le realiza una búsqueda de vecinos de escala cromática se encuentra en el intervalo 1 (ecuación 45), se generan dos números aleatorios, así:

- s → entero aleatorio entre *Linf* y el valor j, proveniente de una distribución uniforme
- p → entero aleatorio entre *j* y el valor C (ecuación 47), proveniente de una distribución uniforme

$$C = \left(\left[\!\left[\mathrm{L\ inf} + \frac{\mathrm{L\ sup - L\ inf}}{2}\right]\!\right] + 1\right) \tag{47}$$

Si, por el contrario, este valor se encuentra en el intervalo 2 (ecuación 46), se generan dos números aleatorios de la siguiente forma:

- s → entero aleatorio entre *C'* (ecuación 48) y el valor j, proveniente de una distribución uniforme

- p → entero aleatorio entre *j* y el valor *Lsup*, proveniente de una distribución uniforme

$$C' = \left(\mathrm{L\,inf} + \frac{\mathrm{L\,sup} - \mathrm{L\,inf}}{2}\right) \quad (48)$$

**Paso 5.** Intercambiar el valor del número aleatorio s con el valor de la posición j de la solución, colocando el antiguo valor de la posición j en el lugar donde se encontraba previamente el valor de s dentro de la misma. Este proceso es análogo para el valor p calculado.

**Paso 6.** Evaluar las dos nuevas soluciones generadas usando los valores s y p, reemplazando la solución actual por la que más satisfaga a la función objetivo. Esta serie de pasos deberá repetirse para cada una de las soluciones o "melodías" que hacen parte del grupo de melodías iniciales (GMI).

#### 3.4.4.7 Melodía conveniente (MC)

Si no se cumple la probabilidad de arreglo melódico (AM), se usa una melodía conveniente. Este operador parte de la elección de una solución inicial, la cual es la combinación de una serie de notas musicales y se asume como la mejor melodía conveniente a utilizar. Esta melodía se escoge a partir del parámetro MELODÍA CONVENIENTE (MC) que se rige bajo una probabilidad de aceptación, lo que implica que, si la probabilidad de aceptación MC es mayor que un número aleatorio entre cero y uno proveniente de una distribución uniforme, se acepta como melodía conveniente a la mejor melodía encontrada en la iteración, o MELODÍA LOCAL (ML). En otras palabras, esta melodía conveniente corresponde a la solución local encontrada hasta el momento con el mejor valor en la función objetivo.

En caso de que MC sea menor que el número aleatorio generado, se escogen dos melodías (S, P) aleatoriamente del conjunto de melodías que se tienen como opción y se le asigna a la primera melodía S escogida el valor de uno y a la segunda melodía P el valor de dos, se genera un numero aleatorio j, esta vez entre uno y dos, y dependiendo de cuál sea su valor, se tomara la melodía asignada a este número, luego entonces la primera posición de la melodía conveniente es igual al primer valor de la melodía correspondiente al número j escogido. Este proceso se repite hasta alcanzar el máximo número de variables que se tenga en la codificación de la solución, en caso de que el número escogido correspondiente a esa posición ya fuese asignado en la melodía conveniente, se escoge el de la siguiente posición en que no allá sido asignado. La figura 10 ilustra un ejemplo.

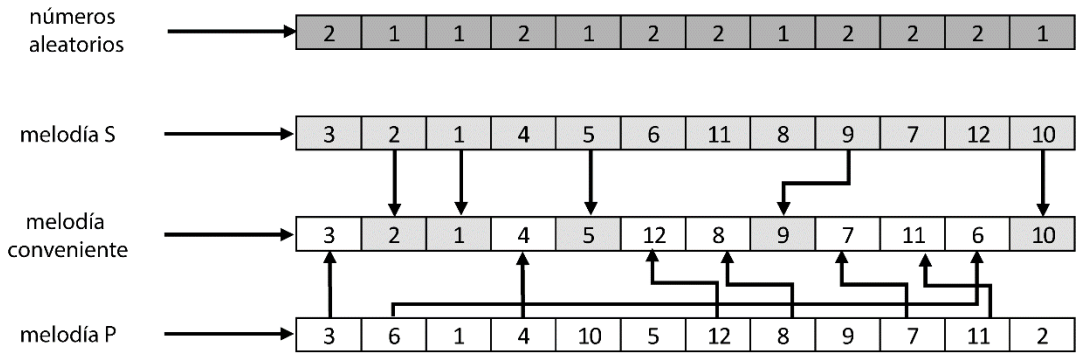



**Figura 10.** Ejemplo combinación de dos melodías aleatorias S y P

Cuando se crea una melodía conveniente combinando dos melodías S y P, esta deberá ser evaluada.

### 3.4.4.8 Variación de la Melodía Conveniente (VMC)

El parámetro VARIACIÓN DE MELODÍA CONVENIENTE (VMC) implica cambiar la melodía conveniente temporal mediante una búsqueda de vecinos, se consideran dos tipos de vecindarios: los que se generan con base en los VECINOS DE INSPIRACIÓN **y** los

que se generan través los VECINOS DE ROTACIÓN DE NOTAS. La variación de la melodía conveniente, se realiza a través del parámetro VMC que se rige bajo una probabilidad de variación, si la probabilidad VMC es mayor que un número aleatorio entre cero y uno proveniente de una distribución uniforme, se modifica la melodía conveniente a través de la INSPIRACIÓN, en caso contrario, se realiza la variación de la melodía conveniente mediante la ROTACIÓN DE NOTAS.

#### 3.4.4.9 Vecinos de inspiración

Este tipo de variación resulta de un cambio de pensamiento y sentido en la posición de una nota específica que es cambiada de forma imprevista en busca de mejorar la melodía que se tiene actualmente. Vecinos de inspiración es un operador equivalente al operador Swap, el cual ha sido ampliamente utilizado en la literatura**.**

#### 3.4.4.10 Vecinos de rotación de notas

Este movimiento tiene como objeto mover las posiciones claves para la escala de una nota en particular, en otras palabras, se hace una semejanza con el proceso musical en el que para una nota los cambios melódicos más importantes y básicos, son los cambios de en su primera, cuarta, quinta y octava nota de su escala musical. Para poder realizar esta variación de melodía se requiere un mínimo de ocho variables, teniendo en cuenta que los movimientos que se realizan necesitan cumplir con ciertas distancias que se simplifican en la escala general de las notas musicales.

Las distancias para los movimientos de vecinos de rotación pueden ser de dos clases que son las descendentes y ascendentes, ambas se explican a continuación:

*a) Vecinos de Rotación Descendentes*

Para los vecinos de rotación descendentes se obtiene la octava de la nota correspondiente, a partir de esta, se obtienen la quinta, la cuarta y la primera y se realizan los movimientos pertinentes. El procedimiento se describe en la siguiente tabla:

**Tabla 1.** Pasos para realizar la búsqueda de rotación descendente.

| | | Paso a paso | |
|---|---|---|---|
| 1 | Octava de la nota correspondiente | Esta se genera con un número aleatorio j entre 8 y las k variables del problema | $8^{va}$→ un número aleatorio *j* entre 8 y las *k* notas de la melodía conveniente actual |
| 2 | Quinta de la nota correspondiente | Esta encuentra al restarle 3 posiciones a la $8^{va}$ | $5^{ta}$ → se obtiene de → $8^{va}$ – 3 |
| 3 | Cuarta de la nota correspondiente | Esta encuentra al restarle 4 posiciones a la $8^{va}$ | $4^{ta}$ → se obtiene de → $8^{va}$ – 4 |
| 4 | Primera de la nota correspondiente | Esta encuentra al restarle 7 posiciones a la $8^{va}$ | $1^{ra}$ → se obtiene de → $8^{va}$ – 7 |

Una vez seleccionadas las posiciones clave, se realizan los movimientos de rotación con sentido hacia la derecha, cambiando así el orden y las posiciones de cada nota. Cada nota gira hacia delante tomando la posición clave más cercana, tal y como se muestra en la figura 11**.**

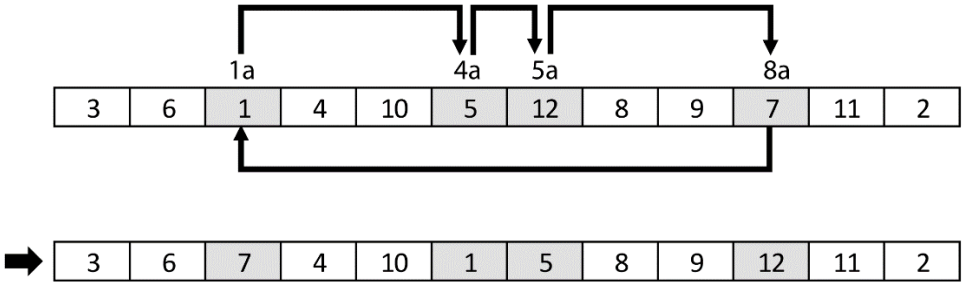



**Figura 11.** Vecinos de rotación descendentes

*b) Vecinos de Rotación Ascendentes*

El procedimiento es similar, sin embargo, la rotación de notas se realiza hacia la izquierda. El procedimiento se describe en la siguiente tabla:

**Tabla 2.** Pasos para realizar la búsqueda de rotación ascendente.

| | | Paso a paso | |
|---|---|---|---|
| 1 | Primera de la nota correspondiente | Esta se genera con un número aleatorio j entre 1 y las (k – 8) variables del problema. | $1^{ra} \rightarrow$ un número aleatorio j entre 1 y las (k - 8) posiciones de la melodía conveniente actual. |
| 2 | Cuarta de la nota correspondiente | Esta encuentra al sumarle 3 posiciones a la $1^{ra}$ | $4^{ta} \rightarrow$ se obtiene de $\rightarrow 1^{ra} + 3$ |
| 3 | Quinta de la nota correspondiente | Esta encuentra al sumarle 4 posiciones a la $1^{ra}$ | $5^{ta} \rightarrow$ se obtiene de $\rightarrow 1^{ra} + 4$ |
| 4 | Octava de la nota correspondiente | Esta encuentra al sumarle 7 posiciones a la $1^{ra}$ | $8^{va} \rightarrow$ se obtiene de $\rightarrow {}^{1ra} +7$ |

Una vez seleccionadas las posiciones claves, se realizan los movimientos de rotación con sentido hacia la derecha, cambiando así el orden y las posiciones de cada nota. Cada nota gira hacia delante tomando la posición clave más cercana, tal y como se muestra en la figura 12**.**

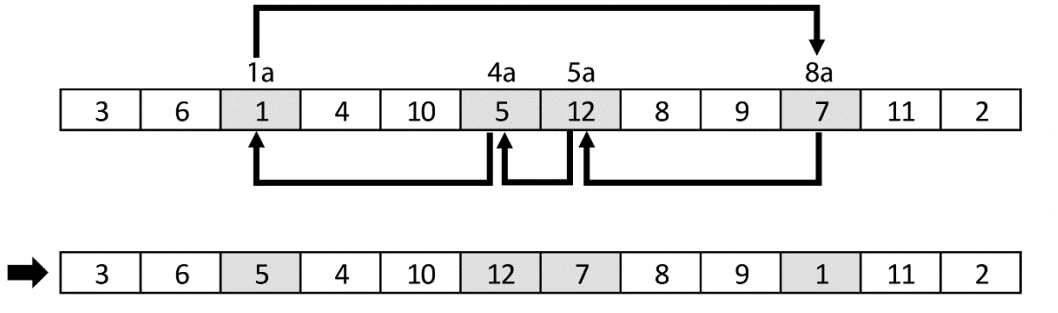



**Figura 12.** Vecinos de rotación ascendentes

### 3.4.4.11 Diagrama de flujo del algoritmo cromático

A continuación, se muestra un diagrama de flujo del algoritmo cromático:

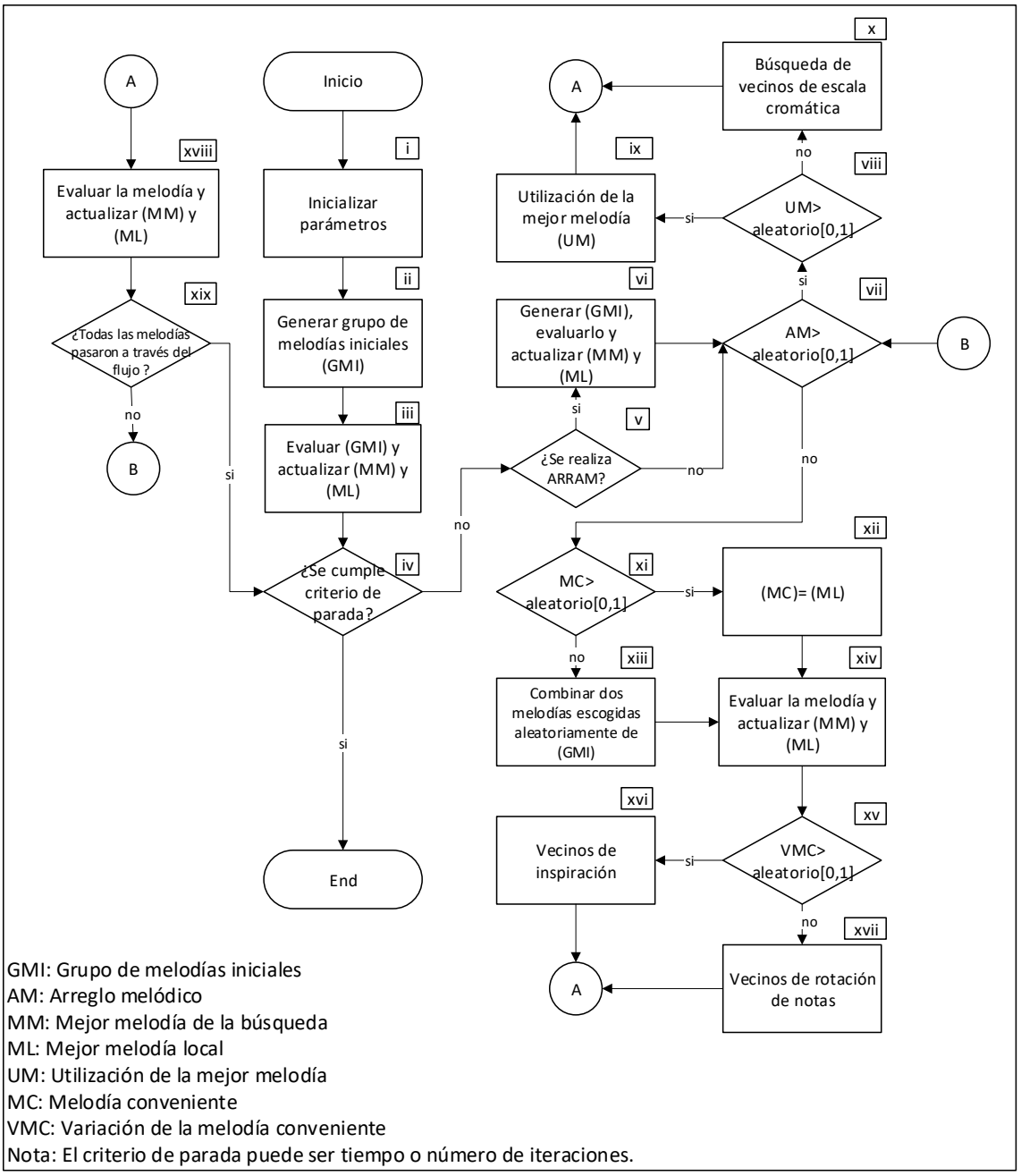



**Figura 13.** Diagrama de flujo del algoritmo cromático

*i. Inicializar parámetros.* Se verifican los parámetros del algoritmo (criterio de parada, tamaño del grupo de melodías iniciales, iteraciones para un ARRAM y probabilidades AM, UM, MC y VMC) y los parámetros para el problema.

*ii.* *Generar grupo de melodías iniciales.* Se crea un conjunto de "melodías iniciales" (GMI) de acuerdo al tamaño definido en *i.*

*iii.* *Evaluar (GMI) y actualizar (MM) y (ML).* De acuerdo a la función de evaluación, se obtienen valores para cada una de las melodías de (GMI) y se actualiza la mejor melodía de la búsqueda (MM) y la mejor melodía de la iteración o melodía local (ML), según corresponda.

*iv.* *¿Se cumple criterio de parada?* Si el criterio de parada se cumple, el algoritmo finaliza su ejecución. De lo contrario, se avanza al paso *v.*

*v.* *¿Se realiza ARRAM?* Se verifica si se cumple el número de iteraciones para realizar un arranque múltiple (ARRAM). Si se cumple, se avanza al paso *vi*, de otro modo, se procede al paso *vii.*

*vi.* *Generar (GMI), evaluarlo y actualizar (MM) y (ML).* Se realizan los procedimientos descritos en los pasos *ii* y *iii*, se reinicia el contador de iteraciones para realizar un ARRAM y se avanza al paso *vii.*

*vii.* *AM > aleatorio [0,1].* Si la probabilidad de arreglo melódico *(AM)* es mayor que un número aleatorio generado, se toma una melodía y se avanza al paso *vii*, de lo contrario, se avanza al paso *xi.*

*viii.* *UM > aleatorio [0,1].* Si la probabilidad de utilización del mejor (*UM*) es mayor a un número aleatorio generado, se realiza la utilización de la mejor melodía (paso *ix*), de lo contrario, se realiza una búsqueda de vecinos de escala cromática para esa melodía (paso *x*).

*ix.* *Utilización de la mejor melodía.* Se le realiza a una melodía, de acuerdo a lo descrito en el título 3.4.4.5 del presente trabajo.

x. *Búsqueda de vecinos de escala cromática.* Se realiza para una melodía, basándonos en el título 3.4.4.6.

xi. *MC > aleatorio [0,1].* Si la probabilidad de melodía conveniente es mayor a un número aleatorio generado, se procede al paso *xii*, en caso contrario, se avanza al paso *xiii.*

xii. *(MC)=(ML).* Se acepta como melodía conveniente a la mejor melodía encontrada en la iteración, o (*ML*).

xiii. *Combinar dos melodías escogidas aleatoriamente de (GMI).* Se realiza de acuerdo a lo descrito en el título 3.4.4.7.

xiv. *Evaluar la melodía y actualizar (MM) y (ML).* Se calcula el valor de la función objetivo para la melodía en cuestión, y se actualizan las melodías (*MM*) y (*ML).*

xv. *VMC > aleatorio [0,1].* Si la probabilidad de variación de la melodía conveniente es mayor a un número aleatorio entre 0 y 1, se procede al paso *xvi*, de lo contrario, se avanza al paso *xvii.*

xvi. *Vecinos de inspiración.* Se realiza el proceso de acuerdo al título 3.4.4.9.

xvii. *Vecinos de rotación de notas.* El procedimiento se basa en el título 3.4.4.10.

xviii. *Evaluar la melodía y actualizar (MM) y (ML).* Se calcula el valor de la función objetivo para la melodía y se actualizan las melodías (*MM*) y (*ML*).

xix. *¿Todas las melodías pasaron a través del flujo?* Teniendo en cuenta que los pasos *vii* a *xviii* se realizan melodía por melodía, en cada iteración, se debe verificar que todas hayan pasado a través del flujo. Si la respuesta es afirmativa, procedemos a verificar el cumplimiento del criterio de parada, por el contrario, si la respuesta es negativa, se procede al paso *vii* con la siguiente melodía.

### 3.4.4.12 Pseudocódigo del algoritmo cromático

A continuación, se muestra el pseudocódigo del algoritmo cromático de (Sabie y Pereira 2011). Se incluyen algunas modificaciones realizadas en este trabajo.

```
INICIO
   INICIALIZAR LOS PARÁMETROS INCLUYENDO, TAMAÑO(N) DE GRUPO DE MELODÍAS INICIALES(GMI)
   MÁXIMO DE ITERACIONES POR ARRAM, AM, UM, MC Y VMC
   GENERAR GRUPO DE MELODÍAS INÍCIALES (GMI) DE TAMAÑO(N)
   EVALUAR CADA (Xi) MELODÍA DEL GRUPO (GMI) Y ACTUALIZAR (MM) Y (ML)
   MIENTRAS QUE NO SE ALCANCE LA CONDICIÓN DE PARADA
         SI SE CUMPLE EL MÁXIMO DE ITERACIONES POR ARRAM
               GENERAR GRUPO DE MELODÍAS INÍCIALES (GMI) DE TAMAÑO(N)
               EVALUAR CADA (Xi) MELODÍA DEL GRUPO (GMI) Y ACTUALIZAR (MM) Y (ML)
         FIN SI
         PARA i = 1 HASTA EL TAMAÑO (N) DEL GRUPO DE MELODÍAS
            SI   LA PROBABILIDAD AM ES MAYOR QUE UN NUMERO ALEATORIO
                 ENTRE CERO Y UNO   PARA LA Xi MELODÍA
                 PARA j = 1 HASTA EL NUMERO K DE VARIABLES DE LA MELODÍA Xi
                    SI   LA PROBABILIDAD UM ES MAYOR QUE UN NUMERO ALEATORIO
                         ENTRE CERO Y UNO
                          REMPLAZAR EL VALOR DE LA VARIABLE j DE LA MELODÍA Xi
                          POR EL VALOR DE LA VARIABLE j DE LA MM
                    SI NO
                       REALIZAR UNA BÚSQUEDA DE VECINOS DE ESCALA CROMÁTICA
                       PARA EL VALOR DE LA VARIABLE j DE LA MELODÍA Xi
                    FIN SI
                 FIN PARA
            SI NO
                 SI   LA PROBABILIDAD MC ES MAYOR QUE UN NUMERO ALEATORIO
                      ENTRE CERO Y UNO PARA LA Xi MELODÍA
                       SE REMPLAZA LA Xi MELODÍA POR LA ML
                 SI NO
                      ESCOGER DOS MELODÍAS ALEATORIAS (S, P) DEL GRUPO DE MELODÍAS
                      ASIGNAR EL VALOR UNO A LA MELODÍA (S) Y DOS A LA MELODÍA (P)
                          PARA j = 1 HASTA EL NUMERO K DE VARIABLES DE LA MELODÍA Xi
                                GENERAR UN NÚMERO ALEATORIO ENTRE UNO Y DOS
                                SI NUMERO ALEATORIO ES IGUAL A UNO
                                   REMPLAZAR EL VALOR DE LA VARIABLE j DE LA MELODÍA Xi
                                   POR EL VALOR DE LA VARIABLE j DE LA MELODÍA S
                                 SI NO
                                   REMPLAZAR EL VALOR DE LA VARIABLE j DE LA MELODÍA Xi
                                   POR EL VALOR DE LA VARIABLE j DE LA MELODÍA P
                                FIN SI
                           FIN PARA
                 FIN SI
                 SI   LA PROBABILIDAD VMC ES MAYOR QUE UN NUMERO ALEATORIO
                      ENTRE CERO Y UNO PARA LA Xi MELODÍA
                      SE REALIZA UNA BÚSQUEDA DE VENOS DE INSPIRACIÓN
                 SI NO
                     SE REALIZA UNA BÚSQUEDA DE VENOS DE ROTACIÓN DE NOTAS
                 FIN SI
            FIN SI
         FIN PARA
   FIN MIENTRAS
FIN
```

### 3.4.5 Algoritmo propuesto I (N1)

En orden cronológico, ese fue el primer algoritmo propuesto por el autor para la solución del problema. Esta metaheurística ha sido desarrollado a partir de la combinación de operadores del algoritmo genético (cruzamiento SB2OX) con operadores del algoritmo cromático (melodía conveniente, búsqueda de vecinos de escala cromática y utilización de la mejor melodía).

Las soluciones son "melodías" y la mejor solución encontrada es (*MM).* Se introducen los parámetros lambda (λ) y gamma (γ). El parámetro lambda (λ), sirve como probabilidad de decisión, permitiendo elegir si se realiza un cruzamiento (SB2OX) de un grupo de melodías iniciales (GMI) o se activa el operador de búsqueda de rotación de notas, que, a diferencia del algoritmo cromático, actúa no sólo en algunos individuos sino en toda la matriz que contiene el grupo de melodías.

Por su parte, el parámetro gamma (γ) funciona como probabilidad decisoria entre dos operadores: búsqueda de vecinos cromática y utilización de la mejor melodía. Luego de esto, se evalúan los resultados y se sigue a la siguiente iteración hasta que el contador iguale el número de iteraciones dispuestas para recombinar las melodías.

Para este algoritmo se agregó un operador de recombinación que consiste en diversificar el grupo de melodías después de cierto número de iteraciones en las que el mejor valor encontrado no cambia, ayudando a que la metaheurística no se estanque en óptimos locales. Este operador renueva el 50% de las melodías y deja intacto el otro 50%.

En el diagrama de flujo de la figura 14, se muestra la forma de operar de este algoritmo, así como los operadores empleados.

### 3.4.5.1 Diagrama de flujo del algoritmo propuesto I (N1)

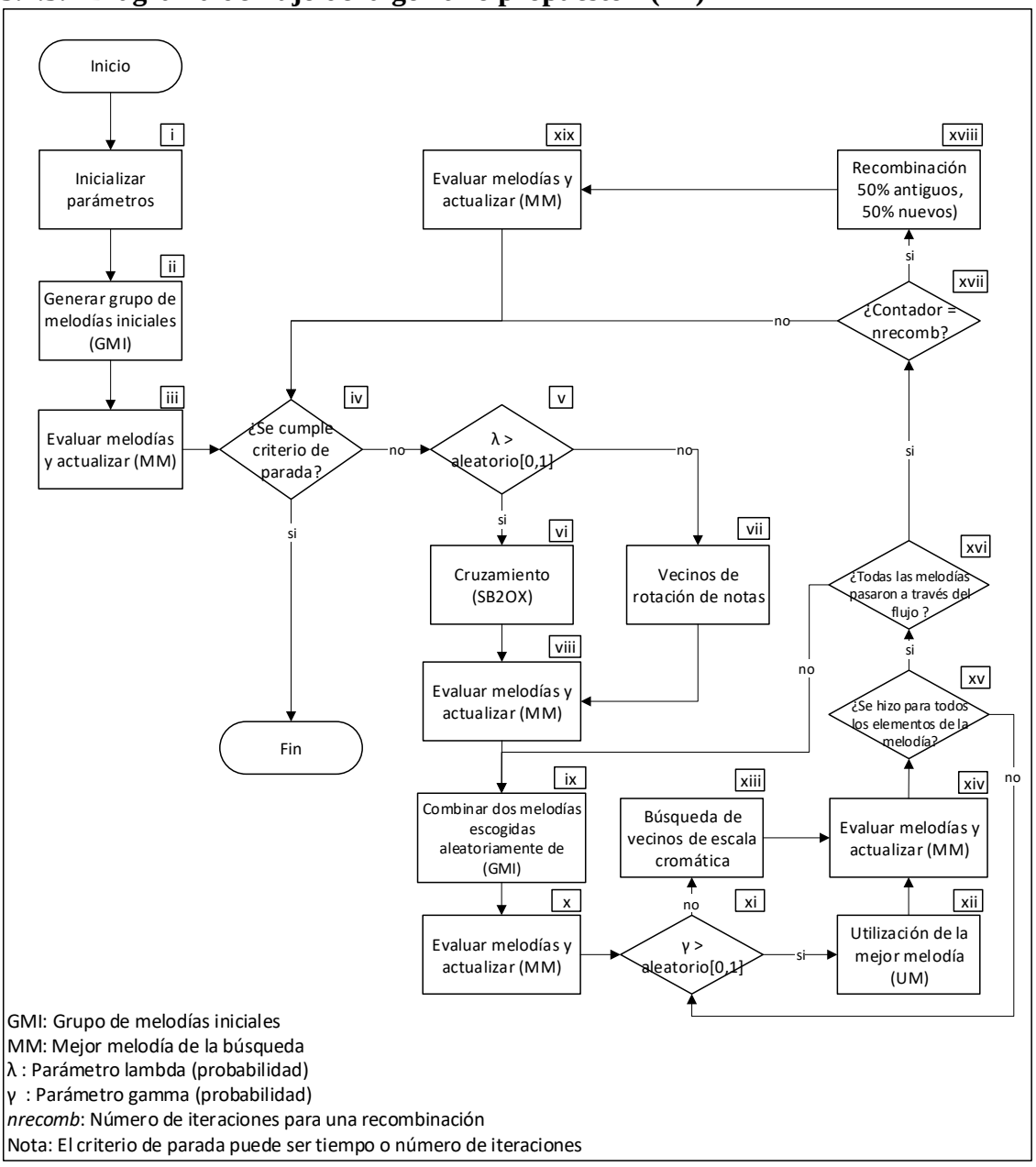



**Figura 14.** Diagrama de flujo del algoritmo propuesto I.

*i.* *Inicializar parámetros.* Se verifican los parámetros del algoritmo (criterio de parada, tamaño del grupo de melodías iniciales, lambda, gamma y *nrecomb*) y los parámetros del problema.

*ii.* *Generar grupo de melodías iniciales (GMI).* Se genera un grupo de soluciones iniciales, en este caso llamadas "melodías", conforme al tamaño establecido en *i.*

*iii.* *Evaluar melodías y actualizar (MM).* Se realiza la evaluación de (GMI) y se actualiza el valor de la mejor melodía de la búsqueda (*MM*).

*iv.* *¿Se cumple el criterio de parada?* Si se cumple, la metaheurística finaliza su ejecución, de lo contrario, se avanza al paso *v.*

*v.* $\lambda >$ *aleatorio [0,1]*. Si el parámetro lambda es mayor que un número aleatorio generado, se realiza el paso *vi*, de otro modo, se avanza al paso *vii*

*vi.* *Cruzamiento (SB2OX).* Se realiza cruzamiento mejorado (SB2OX) (Ruiz et al. 2005) de dos en dos, para toda la matriz (GMI), con una probabilidad de cruzamiento igual al parámetro lambda. En caso de no cruzarse, las melodías no sufren modificación.

*vii.* *Vecinos de rotación de notas.* Se aplica el operador de vecinos de rotación de notas a toda la matriz (GMI), no aplicándolo melodía por melodía (rotación de elementos en una sola solución), sino a todo el conjunto de soluciones (GMI) de una sola vez (rotación de columnas en la matriz de soluciones).

*viii.* *Evaluar melodías y actualizar (MM).* Se realiza la evaluación de (GMI) y se actualiza el valor de la mejor melodía de la búsqueda (*MM*).

*ix.* *Combinar dos melodías escogidas aleatoriamente de (GMI).* En este punto, se empieza a modificar melodía por melodía. Se realiza el procedimiento de combinación de dos melodías de acuerdo a lo establecido en el título 3.4.4.7.

*x.* *Evaluar melodía y actualizar (MM).* Se evalúa la melodía modificada en el paso *ix* y se actualiza (*MM*).

*xi.* $\gamma >$ *aleatorio [0,1]*. Si el parámetro gamma es mayor que un número generado de forma aleatoria entre 0 y 1, se procede a realizar el paso *xii*, de lo contrario,

se avanza al paso *xiii.* En este punto empieza la transformación elemento a elemento.

*xii.* *Utilización de la mejor melodía (UM).* Se lleva a cabo el procedimiento descrito en el título 3.4.4.5 para la melodía en el elemento en cuestión.

*xiii.* *Búsqueda de vecinos de escala cromática.* Se aplica la búsqueda de vecinos para la melodía en el elemento en cuestión, de acuerdo a lo establecido en el título 3.4.4.6.

*xiv.* *Evaluar melodía y actualizar (MM).* Se evalúa la melodía y, si esta es mejor que la mejor solución encontrada, se realiza la actualización de (*MM*).

*xv.* *¿Se hizo para todos los elementos de la melodía?* Se verifica que el procedimiento *xii* o *xiii* se haya realizado para todos los elementos de la melodía. En caso afirmativo, procedemos a verificar si se modificaron todas las melodías del grupo (paso *xvi*), de otro modo, continuamos con la modificación del siguiente elemento de la melodía (paso *xi*).

*xvi.* *¿Todas las melodías pasaron a través del flujo?* Se comprueba que todas las melodías hayan pasado a través del flujo. Si se cumple, se avanza al paso *xvii*, de lo contrario, recorremos el flujo con la siguiente melodía desde el paso *ix*.

*xvii.* *¿Contador = nrecomb?* Si la mejor solución (*MM)* permanece idéntica durante un número *nrecomb* de iteraciones, el valor de *Contador* será igual a *nrecomb* y deberá llevarse a cabo el procedimiento descrito en el paso *xviii.* Si no se cumple, nos dirigimos al paso *iv*.

*xviii.* *Recombinación (50% antiguos, 50% nuevos).* Se mantiene el 50% de las melodías y se reemplaza la otra mitad de (GMI) por nuevas melodías para procurar el escape de óptimos locales.

### 3.4.5.2 Pseudocódigo del algoritmo propuesto I

A continuación, se presenta el pseudocódigo del algoritmo (N1) desarrollado por el autor.

```
INICIALIZAR LOS PARÁMETROS, INCLUYENDO CRITERIO DE PARADA, TAMAÑO (N) DEL GRUPO DE MELODÍAS
INICIALES (GMI), LAMBDA (λ), GAMMA (γ) Y NÚMERO DE (nrecomb).
GENERAR GRUPO DE MELODÍAS INÍCIALES (GMI) DE TAMAÑO(N)
EVALUAR CADA (Xi) MELODÍA DEL GRUPO (GMI) Y ACTUALIZAR (MM)
MIENTRAS NO SE CUMPLA EL CRITERIO DE PARADA
        SI EL PARÁMETRO LAMBDA(λ) ES MAYOR QUE UN NÚMERO ALEATORIO ENTRE 0 Y 1
                REALIZAR CRUZAMIENTO SB2OX A TODO EL GRUPO (GMI), CON PROBABILIDAD DE CRUZAMIENTO (λ)
        SI NO
                REALIZAR VECINOS DE ROTACIÓN DE NOTAS POR COLUMNAS AL GRUPO (GMI)
        FIN SI
        EVALUAR CADA (Xi) MELODÍA DEL GRUPO (GMI) Y ACTUALIZAR (MM)
        PARA i = 1 HASTA EL TAMAÑO (N) DEL GRUPO DE MELODÍAS(GMI)
                ESCOGER DOS MELODÍAS ALEATORIAS (S, P) DEL GRUPO DE MELODÍAS
                ASIGNAR EL VALOR UNO A LA MELODÍA (S) Y DOS A LA MELODÍA (P)
                PARA j = 1 HASTA EL NUMERO K DE VARIABLES DE LA MELODÍA Xi
                        GENERAR UN NÚMERO ENTERO ALEATORIO ENTRE UNO Y DOS
                        SI NUMERO ALEATORIO ES IGUAL A UNO
                                REMPLAZAR EL VALOR DE LA VARIABLE j DE LA MELODÍA Xi POR EL
                                VALOR DE LA VARIABLE j DE LA MELODÍA S
                        SI NO
                                REMPLAZAR EL VALOR DE LA VARIABLE j DE LA MELODÍA Xi
                                POR EL VALOR DE LA VARIABLE j DE LA MELODÍA P
                        FIN SI
                FIN PARA
                EVALUAR LA MELODÍA (X_i) Y ACTUALIZAR (MM)
                PARA j = 1 HASTA EL NUMERO K DE VARIABLES DE LA MELODÍA Xi
                        SI EL PARÁMETRO GAMMA (γ) ES MAYOR QUE UN NÚMERO ALEATORIO ENTRE 0 Y 1
                                REMPLAZAR EL VALOR DE LA VARIABLE j DE LA MELODÍA Xi
                                POR EL VALOR DE LA VARIABLE j DE LA (MM)
                        SI NO
                                REALIZAR UNA BÚSQUEDA DE VECINOS DE ESCALA CROMÁTICA
                                PARA EL VALOR DE LA VARIABLE j DE LA MELODÍA Xi
                        FIN SI
                FIN PARA
                EVALUAR LA MELODÍA (X_i) Y ACTUALIZAR (MM)
                SI (MM) PERMANECE IGUAL DURANTE (nrecomb) ITERACIONES
                        REALIZAR RECOMBINACIÓN (50%, 50%)
                FIN SI
        FIN PARA
FIN MIENTRAS
```

### 3.4.6 Algoritmo propuesto II (N2)

Esta segunda propuesta encierra aspectos de distintas metaheurísticas, tales como la optimización por enjambre de partículas (PSO), el algoritmo cromático (AC) y la búsqueda tabú (TS).

De la metaheurística PSO, se adoptó el concepto de "partículas", tomando cada solución como una partícula independiente de un "enjambre de partículas" que exploran y explotan el espacio de soluciones. Cada partícula representa una solución ($X_i$), la cual tiene asociado un valor de evaluación en la función objetivo ($F_i$), además de la mejor posición visitada por esta partícula ($pBest_i$) y su evaluación ($FpBest_i$). Así mismo, se define ($G$) como la mejor posición encontrada por el "enjambre de partículas" y ($F_g$) como el valor de su evaluación. Cada partícula es obtenida de la generación y evaluación de ($Ngen_n$) soluciones aleatorias, de las cuales se escoge la que mejor valor presente en la función objetivo y se conforma un enjambre con ($N$) partículas.

En este punto, entra en juego el algoritmo cromático. De esta metaheurística, son tomados los siguientes operadores: Utilización de la mejor melodía, combinación de dos melodías seleccionadas aleatoriamente y búsqueda de vecinos de rotación de notas. Para cada partícula, se genera un número aleatorio ($alt$) entre cero y uno en cada iteración, si *alt* es menor que 0.25, a la partícula se le aplica el operador de utilización de la mejor melodía (que en este caso será la partícula *G*), lo que implica reemplazar , uno a uno, los valores en cada posición *j* de la partícula ($X_i$), por los en la posición *j* de la partícula ($G$) (y realizar la respectiva reparación de la solución en cada caso), teniendo en cuenta que, cada vez que se realiza un cambio de *valores* en la partícula ($X_i$), se verifica que el nuevo ($F_i$) sea menor que ($FpBest_i$), de no ser así, ($X_i$) vuelve a su mejor posición histórica, es decir, se

convierte en ($pBest_i$); si *alt* es mayor o igual que 0.25 y estrictamente menor que 0.5, se implementa la combinación de dos melodías seleccionadas aleatoriamente (que, en este caso, representa la fusión de las posiciones de dos partículas escogidas al azar del enjambre de partículas), lo cual implica tomar dos partículas y utilizar la "información de sus posiciones" para crear una nueva partícula, haciendo uso del procedimiento planteado en el título (3.4.4.7) para *MC < aleatorio [0,1]*. Así, la partícula ($X_i$) es reemplazada por una nueva partícula ($X_i'$) que resulta de este operador, si y solo si, ($F_i'$) es menor que ($FpBest_i$), de lo contrario, ($X_i$) es reemplazada por ($pBest_i$); si *alt* es estrictamente mayor que 0.5 y menor que 0.75, se implementa una búsqueda de vecinos de rotación de notas (que, para este caso, se entiende como la rotación, ascendente o descendente, de los elementos de una partícula, aplicando conceptos relacionados con las posiciones "clave" para modificar una melodía en el ámbito musical), con la que se obtiene la partícula ($X_i'$), la cual sólo podrá reemplazar a ($X_i$), si ($F_i'$) es menor que ($FpBest_i$), de otro modo, ($X_i$) es reemplazada por ($pBest_i$); si *alt* es mayor o igual a 0.75, no se aplica ninguno de los operadores mencionados y ($X_i$) avanza a la siguiente etapa del algoritmo sin modificaciones.

Cabe resaltar que, luego de generar *alt* y aplicar el operador definido a la partícula, se realiza una actualización de la mejor solución encontrada, (*G)*, y su valor en la función objetivo, ($F_g$).

En este punto, es el turno de la búsqueda tabú, o *Tabú Search* (Batista y Glover 2006), de la cual se adoptan algunos conceptos. El primero es el concepto de memoria a corto plazo, el cual es implementado para cada partícula de forma individual. Se tiene una matriz ($MInd_i$) de tres dimensiones que relaciona y almacena los intercambios en la lista tabú de

cada partícula durante ($t$) periodos, registrando sólo aquellos cambios realizados mediante el uso de un operador de búsqueda tabú. Si una partícula $i$ sufre una transformación producto del uso de un operador distinto, su lista tabú, ($MInd_i$), es reiniciada. A continuación, se muestra la estructura de la lista tabú para cada partícula:

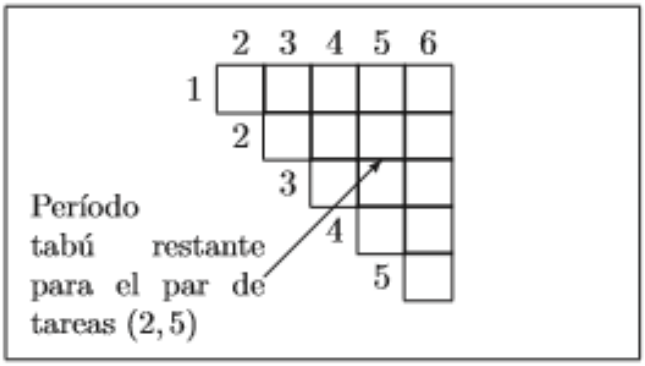



**Figura 15.** Estructura de datos tabú.

Tomada de: Introducción a la búsqueda tabú, 2006

Para entender la estructura de datos planteada en la figura 5, supongamos un problema de ruteo de vehículos de 6 clientes. Si se realiza el intercambio del cliente 2 con el cliente 5 (figura 16), se colocará el valor del periodo tabú ($t$) en el espacio señalado por la flecha (figura 15), el cual, mientras sea mayor que cero, restringirá parcialmente el intercambio del par (5,2) en el operador de búsqueda tabú. Este valor disminuye en una unidad con cada iteración, siendo cero su valor mínimo.

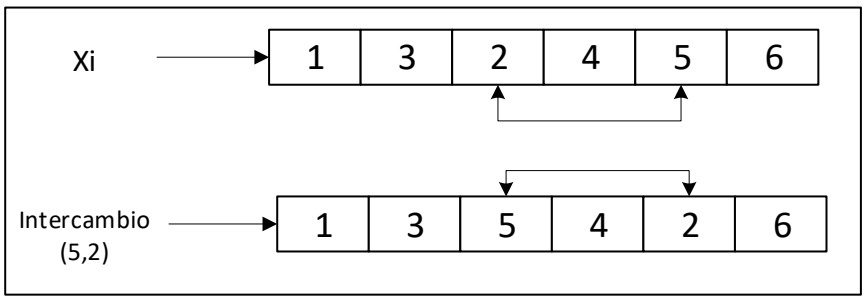



**Figura 16.** Ejemplo de intercambio del par de clientes (5,2) en una solución

Así, la matriz tridimensional ($MInd_i$) estará compuesta de *i* capas (cada capa es la lista tabú de una partícula), y cada capa tendrá una estructura similar a la mostrada en figura 15 para representar la lista tabú de cada partícula. A modo de ejemplo, la figura 17 describe la forma en que esta metaheurística almacena las listas (*TS*) para *i=4* partículas.

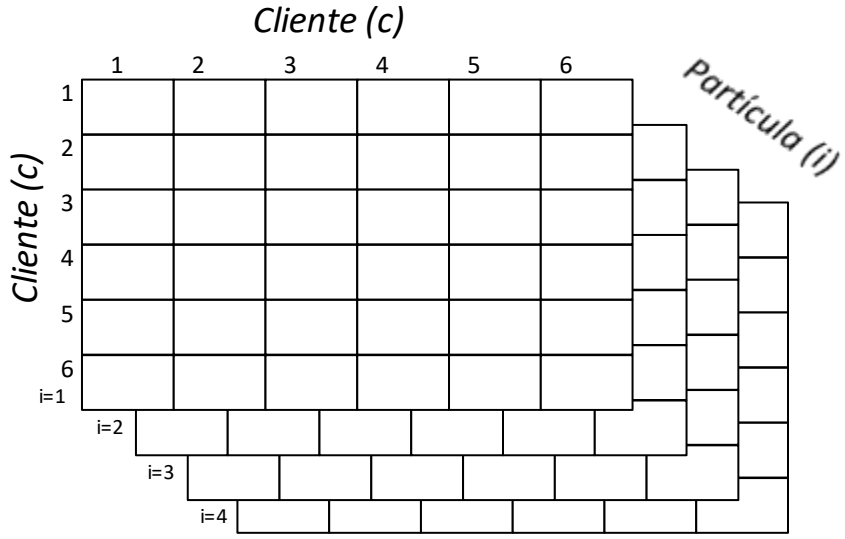



**Figura 17.** Estructura multi – capas para la lista tabú por partícula.

Un segundo concepto tomado de la búsqueda tabú es el de memoria a largo plazo, el cual ha sido adaptado al problema de ruteo de vehículos (en especial para problemas con flota heterogénea). Se ha diseñado una memoria global ($MGlobal_{cpv}$) de tamaño (*k, k, K*), donde *k* representa el número de elementos de la solución (o número de clientes) y *K* es el número de vehículos de una flota heterogénea. En esta memoria se guardará la frecuencia histórica global de la relación cliente-posición-vehículo. De esta forma, cada vez que ($X_i$) cambie de posición a una que mejore a ($FpBest_i$) en cualquiera de los 3 operadores del algoritmo cromático, o bien, cada vez que ($X_i$) se mueva hacia un vecino como resultado de la implementación del operador (*TS*), esta memoria almacenará los datos de la relación cliente-posición-vehículo de la nueva posición. Estos datos serán

utilizados para calcular una *tasa media de permanencia,* la cual se halla para cada vecino potencial dentro del marco del operador (*TS*), y se perfila como una herramienta de diversificación que busca penalizar a los candidatos a vecinos de ($X_i$) que tengan un alto grado de permanencia con relación a un cliente en una misma posición y asignado a un mismo vehículo. El desarrollo de esta tasa se atribuye al autor de esta metaheurística.

Ahora bien, una casilla de la memoria $MGlobal_{cpv}$ aumenta en una unidad si, en una solución ($X_i$') que mejora ($X_i$) en términos de valor de evaluación, el cliente *c* está en la posición *p* y está asignado al vehículo *v*. Supongamos un problema con 6 clientes y una flota de 4 vehículos. La estructura de datos es como se muestra a continuación.

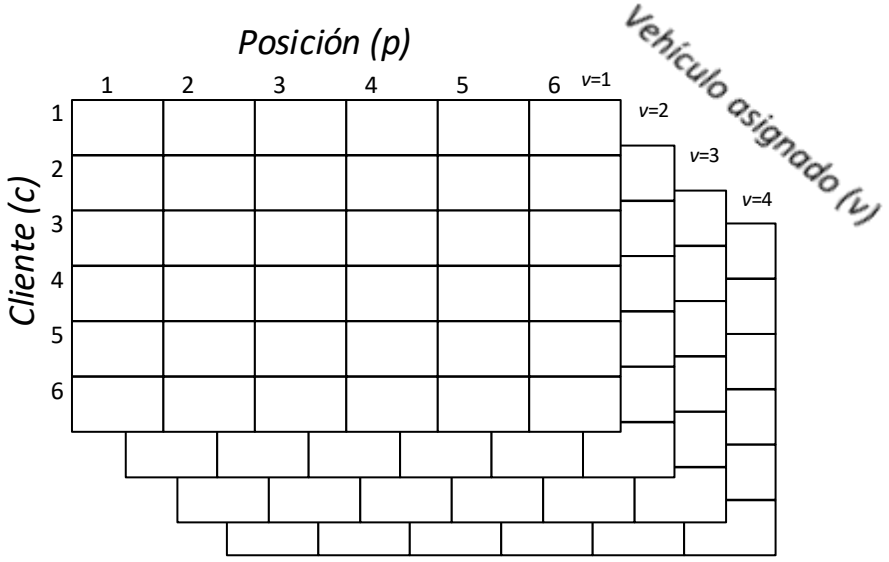



**Figura 18.** Estructura de datos de la memoria a largo plazo $MGlobal_{cpv}$.

Si se presenta una solución $X_i$' = {1, 4, 5, 6, 3, 2} que mejora la solución con respecto a ($X_i$), y esta solución tiene un vector de asignación de vehículos $V_i$' = {1, 4, 2, 1, 2, 2}, en la memoria $MGlobal_{cpv}$ se guardaría, por ejemplo, un uno en la fila uno, columna uno,

capa número uno. A continuación, observamos cómo quedaría esta memoria luego de ingresar esta solución.

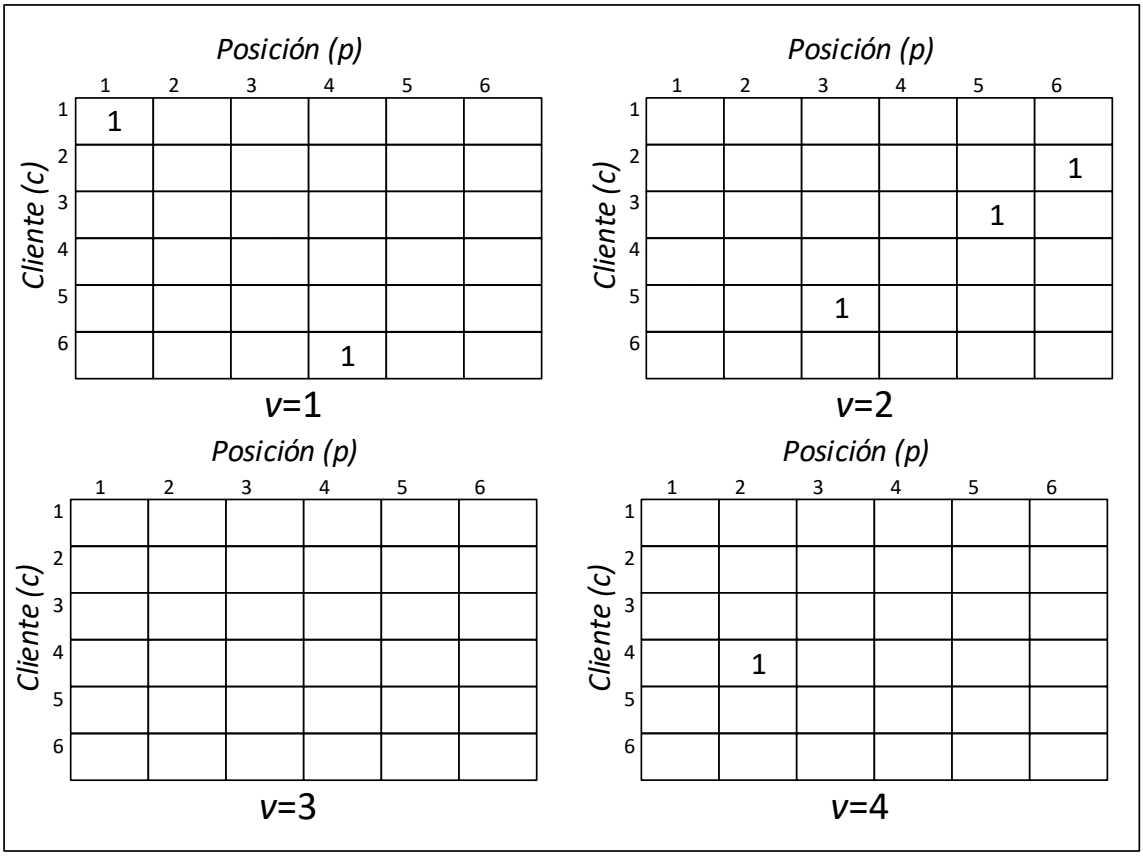



**Figura 19.** Ejemplo – almacenamiento de memoria a largo plazo $MGlobal_{cpv}$.

Cabe mencionar que los vecinos de cada partícula no son generados y evaluados en su totalidad, sino que se cuenta con un parámetro (*nran)*, que se define como el porcentaje de vecinos a generar dentro del rango total de vecinos (ver ecuaciones 49 y 50).

$$Número\ de\ vecinos\ posibles = k\ \boldsymbol{C}\ 2 = \frac{k!}{2!\,(k-2)!} \tag{49}$$

$$\binom{Número\ de\ vecinos}{a\ generar} = \left[\!\left[\begin{matrix} numero\ de\ vecinos \\ posibles * nran \end{matrix}\right]\!\right] \tag{50}$$

Los vecinos son representados como ($Xvecinos_{im}$), donde *i* es el índice de la partícula de la cual es vecino y *m* es el identificador o número de vecino.

A continuación, se usará "$cliente(Xvecinos_{im}, j)$", para referirnos al cliente que se encuentra en la posición *j* del vecino *m* de la partícula *i*. Análogamente, se usará "$vehículo(Xvecinos_{im}, j)$", para hacer referencia al vehículo que está asignado al cliente de la posición *j* del vecino *m* de la partícula *i*. Teniendo en cuenta esto, se presenta el cálculo de la *tasa media de permanencia* para el vecino $Xvecinos_{im}$.

Sea $c(j) = cliente(Xvecinos_{im}, j)$, y $v(j) = vehículo(Xvecinos_{im}, j)$.

$$Tmperm(Xvecinos_{im}) = \sum_{j=1}^{k} \frac{MGlobal_{(c(j),\ j\ ,v(j)\ )}}{it - actual}, \forall m \qquad (51)$$

Donde *Tmperm* es la tasa media de permanencia e *it-actual* es el número de iteraciones realizadas hasta el momento, arrancando con valor de uno en la primera iteración.

El concepto de *valor de movimiento* también será empleado, ya que se calculará para cada uno de los vecinos de cada partícula *i*, y de este valor, además de la tasa *Tmperm*, dependerá el orden en la lista tabú para la elección del vecino que reemplazará a $X_i$. Así, el valor del movimiento se calcula mediante la ecuación 52.

$$Valor\ del\ movimiento = F(Xvecinos_{im}) - Factual(X_i), \forall m \qquad (52)$$

Mientras que el valor del movimiento brinda información sobre si un vecino mejora (valor de movimiento menor que cero), iguala (valor igual a cero) o desmejora el valor $Factual(X_i)$ (valor mayor que cero), el resultado de *Tmperm* permitirá penalizar a los vecinos de acuerdo a este criterio. Para el cálculo del valor (*Vvecino*), que definirá el orden y hará más o menos atractivo el movimiento hacia un vecino, se utiliza la ecuación 53.

$$Vvecino = Valor\ del\ movimiento + Tmperm * 10 \qquad (53)$$

Así, de acuerdo al valor *Vvecino*, se organizan todos los vecinos de menor a mayor y se selecciona aquel de menor valor que **no se encuentre en la lista tabú** de la partícula *i.* Ahora bien, es razonable pensar que las restricciones tabúes deberían ser flexibles en ciertos casos, en este punto entra en juego el concepto de *Criterio de aspiración*, el cual nos permite saltarnos una restricción tabú, si y solo si, el vecino al que nos vamos a mover proporciona una mejor solución que la mejor solución encontrada ($F_g$)

Luego de realizar una búsqueda tabú con un porcentaje establecido para el rango (*nran)* y de que la partícula $X_i$ realice un movimiento hacia uno de sus vecinos, se realiza la actualización de la memoria a corto plazo ($MInd_i$), de la memoria única de largo plazo ($MGlobal_{cpv}$), de la mejor solución encontrada por la partícula ($pBest_i$) y su evaluación ($FpBest_i$), y, de ser necesario, de la mejor posición encontrada por el enjambre ($G$) y su evaluación ($F_g$).

Con esto se completaría el recorrido de una partícula por los operadores de este algoritmo. Se requiere que todas las partículas atraviesen el flujo para completar una iteración, y el criterio de parada del algoritmo podrá ser expresado en términos de tiempo o de número de iteraciones.

Por último, se incluye un operador de arranque múltiple, o de reinicio, que funciona de acuerdo al parámetro *ARRAM,* generando y evaluando un nuevo enjambre cada vez que se cumpla un número *ARRAM* de iteraciones.

### 3.4.6.1 Diagrama de flujo del algoritmo propuesto II (N2)

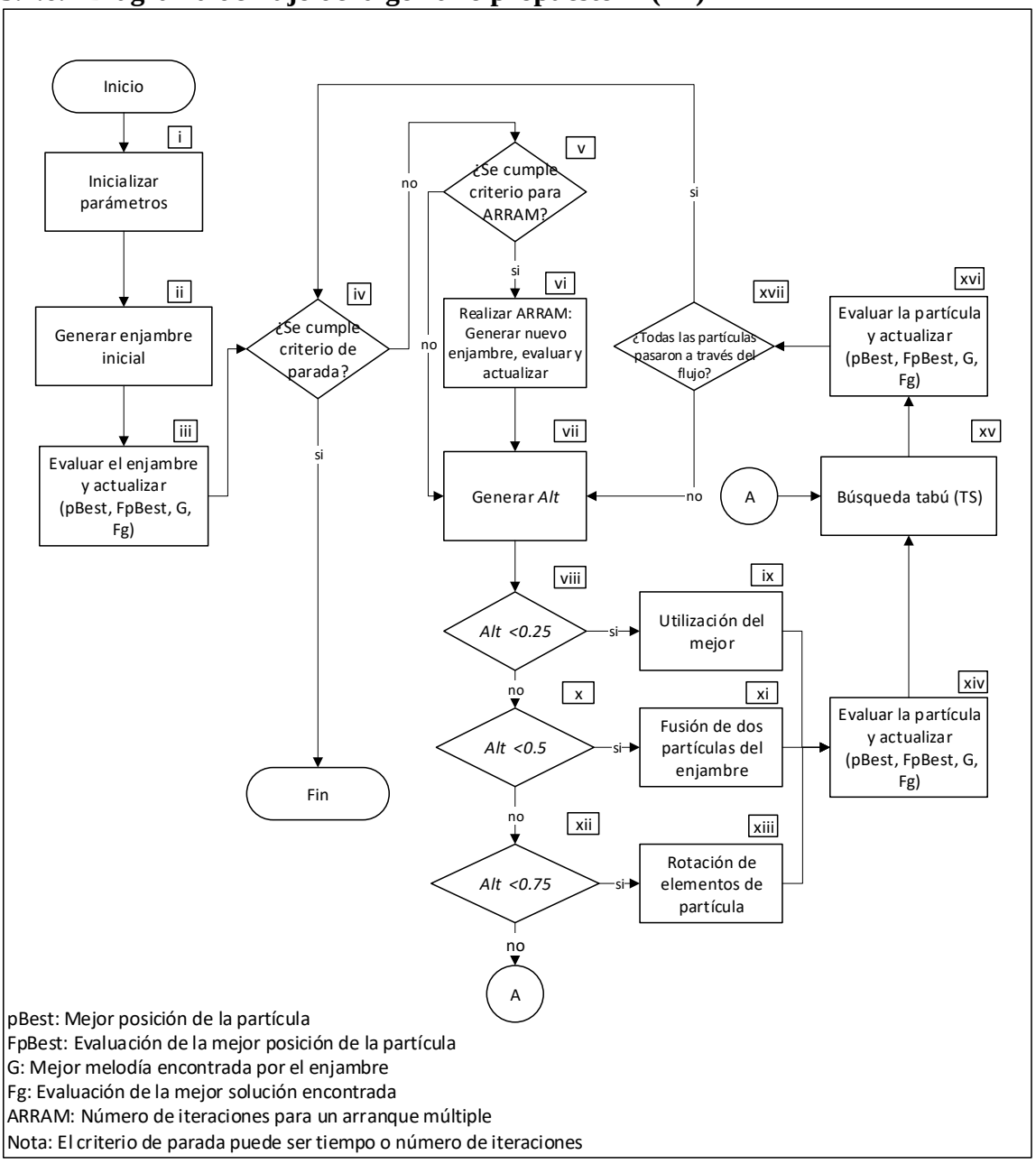



**Figura 20.** Diagrama de flujo del algoritmo propuesto II.

*i. Inicializar parámetros.* Se verifican los parámetros de la metaheurística ($Ngen_n$, N, *ARRAM*, *t*, *nran*) y los parámetros del problema.

*ii. Generar enjambre inicial.* Se genera un enjambre de tamaño *N*.

*iii. Evaluar el enjambre y actualizar.* Se actualiza *pBest, FpBest, G* y *Fg,* para las *N* partículas del enjambre.

*iv. ¿Se cumple criterio de parada?* Si el criterio de parada se cumple, el algoritmo finaliza su ejecución. De lo contrario, se avanza al paso *v.*

*v. ¿Se cumple criterio para ARRAM?* Se verifica si se cumple el número de iteraciones para realizar un arranque múltiple (ARRAM). Si se cumple, se avanza al paso *vi,* de otro modo, se procede al paso *vii.*

*vi. Generar nuevo enjambre, evaluar y actualizar.* Se realizan los procedimientos descritos en los pasos *ii* y *iii,* se reinicia el contador de iteraciones para realizar un ARRAM y se avanza al paso *vii.*

*vii. Generar Alt.* Se genera un número aleatorio *Alt* que definirá el camino a seguir para cada partícula en cada iteración.

*viii. Alt <0.25.* Si el número *Alt* es menor que 0.25, se procede al paso *ix,* de lo contrario, se avanza al paso *x.*

*ix. Utilización del mejor.* Se utilizan los elementos de la mejor partícula, de acuerdo con lo establecido en el título 3.4.4.5.

*x. Alt <0.5.* Si *Alt* es mayor o igual que 0.25 y menor que 0.5, se procede al paso *xi,* de otro modo, se avanza al paso *xii.*

*xi. Fusión de dos partículas del enjambre.* Se rige por el principio de combinar dos melodías escogidas aleatoriamente, de acuerdo a lo establecido en el título 3.4.4.7 para *MC<aleatorio.*

*xii. Alt <0.75. Si Alt* es mayor o igual a 0.5 y menor a 0.75, se realiza el paso *xiii,* de otro modo, se avanza al paso *xv.*

xiii. *Rotación de elementos de partícula.* Se aplica este operador a la partícula $X_i$, de acuerdo a lo establecido en el título 3.4.4.10.

xiv. *Evaluar la partícula y actualizar (pBest, FpBest, G, Fg).* Se toma la partícula y se actualizan los parámetros asociados después de algún movimiento de la misma.

xv. *Búsqueda Tabú (TS).* Se generan los vecinos de la partícula $X_i$ de acuerdo al porcentaje de vecinos a visitar, (*nran).* Se calculan los valores de *Vvecino* y *Tmperm* para cada uno de ellos, y, de acuerdo a estos valores, a la lista tabú de la partícula y al criterio de aspiración, se determina el intercambio que se realizará en $X_i$.

xvi. *Evaluar la partícula y actualizar (pBest, FpBest, G, Fg).* Después de salir del operador de Búsqueda tabú, se toma la partícula y se actualizan los parámetros asociados.

xvii. *¿Todas las partículas pasaron a través del flujo?* Se comprueba que todas las partículas hayan pasado a través del flujo. Si se cumple, se avanza al paso *vi,* de lo contrario, recorremos el flujo con la siguiente partícula desde el paso *vii*.

## 3.4.6.2 Pseudocódigo del algoritmo propuesto II

```
INICIALIZAR LOS PARÁMETROS, INCLUYENDO CRITERIO DE PARADA, NÚMERO DE SOLUCIONES PARA GENERAR UNA PARTÍCULA (Ngen_n), NÚMERO DE PARTÍCULAS DEL ENJAMBRE (N), NÚMERO DE ITERACIONES SIN MEJORA PARA UN ARRANQUE MÚLTIPLE (ARRAM), NÚMERO DE PERIODOS TABÚ (t), Y PORCENTAJE DE VECINOS A GENERAR (nran)
PARA i=1 HASTA EL TAMAÑO (N) DE PARTÍCULAS
        GENERAR Y EVALUAR (Ngen_n) SOLUCIONES
        CONVERTIR LA MEJOR DE LAS (Ngen_n) SOLUCIONES GENERADAS EN LA PARTÍCULA X(i)
FIN PARA
ACTUALIZAR LA MEJOR POSICIÓN DE CADA PARTÍCULA (pBest) Y SU EVALUACIÓN (FpBest)
ACTUALIZAR LA MEJOR SOLUCIÓN ENCONTRADA (G) y SU EVALUACIÓN (Fg).
MIENTRAS NO SE CUMPLA EL CRITERIO DE PARADA
        SI SE CUMPLE EL MÁXIMO DE ITERACIONES POR ARRAM
                PARA i=1 HASTA EL TAMAÑO (N) DE PARTÍCULAS
                        GENERAR Y EVALUAR (Ngen_n) SOLUCIONES
                        CONVERTIR LA MEJOR DE LAS (Ngen_n) SOLUCIONES GENERADAS EN LA PARTÍCULA Xi
                FIN PARA
                ACTUALIZAR LA MEJOR POSICIÓN DE CADA PARTÍCULA (pBest) Y SU EVALUACIÓN (FpBest)
                ACTUALIZAR LA MEJOR SOLUCIÓN ENCONTRADA (G) y SU EVALUACIÓN (Fg).
        FIN SI
        PARA i = 1 HASTA EL TAMAÑO (N) DE PARTÍCULAS DEL ENJAMBRE
                GENERAR UN NÚMERO ALEATORIO (Alt) ENTRE CERO Y UNO
                SI Alt < 0.25
                        PARA j = 1 HASTA EL NUMERO k DE ELEMENTOS DE LA PARTÍCULA Xi
                                REEMPLAZAR EL VALOR DEL ELEMENTO j DE LA PARTÍCULA Xi POR EL VALOR DE LA ELEMENTO j DE (G) Y REALIZAR REPARACIÓN
                                EVALUAR LA PARTÍCULA Xi
                                SI F(Xi) < FpBest_i
                                        ACTUALIZAR LA MEJOR POSICIÓN DE LA PARTÍCULA (pBest_i) Y (FpBest_i)
                                        ACTUALIZAR LA MEJOR SOLUCIÓN ENCONTRADA (G) y SU EVALUACIÓN (Fg)
                                        REINICIAR LISTA TABÚ DE Xi (MInd_i)
                                        ACTUALIZAR LA MEMORIA A LARGO PLAZO (MGlobal_cpv)
                                SI NO
                                        Xi ES REEMPLAZADA POR pBest_i
                                        F(Xi) ES REEMPLAZADA POR FpBest_i
                                FIN SI
                        FIN PARA
                SINO SI 0.25 <= Alt < 0.5
                        ESCOGER DOS PARTÍCULAS ALEATORIAS (S, P) DEL ENJAMBRE
                        ASIGNAR EL VALOR UNO A LA PARTÍCULA (S) Y DOS A LA PARTÍCULA (P)
                        PARA j = 1 HASTA EL NUMERO k DE ELEMENTOS DE LA PARTÍCULA Xi
                                GENERAR UN NÚMERO ENTERO ALEATORIO ENTRE UNO Y DOS
                                SI NUMERO ALEATORIO ES IGUAL A UNO
                                        REMPLAZAR EL VALOR DEL ELEMENTO j DE LA PARTÍCULA Xi POR EL VALOR DE LA ELEMENTO j DE LA PARTÍCULA S
                                SI NO
                                        REMPLAZAR EL VALOR DEL ELEMENTO j DE LA MELODÍA Xi POR EL VALOR DEL ELEMENTO j DE LA MELODÍA P
                                FIN SI
                        FIN PARA
                SINO SI 0.5 <= Alt < 0.75
                        REALIZAR ROTACIÓN DE ELEMENTOS DE PARTÍCULA
                FIN SI
                EVALUAR LA PARTÍCULA Xi
                SI F(Xi) < FpBest_i
                        ACTUALIZAR LA MEJOR POSICIÓN DE LA PARTÍCULA (pBest_i) Y (FpBest_i)
                        ACTUALIZAR LA MEJOR SOLUCIÓN ENCONTRADA (G) y SU EVALUACIÓN (Fg)
                        REINICIAR LISTA TABÚ DE Xi (MInd_i)
                        ACTUALIZAR LA MEMORIA A LARGO PLAZO (MGlobal_cpv)
                SI NO
                        Xi ES REEMPLAZADA POR pBest_i
                        F(Xi) ES REEMPLAZADA POR FpBest_i
                FIN SI
                CREAR VECINOS DE Xi TENIENDO EN CUENTA EL PORCENTAJE DE VECINOS A GENERAR (nran)
                CALCULAR TASA MEDIA DE PERMANENCIA (Tmperm) Y EL VALOR DE POSICIÓN (Vvecino) PARA CADA VECINO
                REALIZAR BÚSQUEDA TABÚ, TENIENDO EN CUENTA LA LISTA TABÚ DE Xi
                ACTUALIZAR LISTA TABÚ DE LA PARTÍCULA (MInd_i)
                ACTUALIZAR LA MEMORIA A LARGO PLAZO (MGlobal_cpv)
        FIN PARA
FIN MIENTRAS
```

## 3.5 DISEÑO Y ANÁLISIS DE EXPERIMENTOS

Hasta este punto, se han presentado dos modelos matemáticos desarrollados por el autor, uno enfocado a la minimización de la pérdida de frescura en un producto perecedero en el proceso de ruteo (que llamamos Caso I), y otro dirigido al mismo objetivo, esta vez para múltiples productos perecederos y teniendo en cuenta el volumen de demanda de los clientes (Caso II). También, se presentó la codificación de las soluciones para cada caso, se detallaron algunas metaheurísticas conocidas, como el algoritmo genético (AG) y el algoritmo de optimización por enjambre de partículas (PSO), se presentó una adaptación de un algoritmo recientemente desarrollado, el algoritmo cromático (AC), y se explicaron a fondo dos nuevas metaheurísticas propuestas por el autor, que llamaremos (N1) y (N2). Para el Caso I, se utilizó el algoritmo genético, el algoritmo PSO, el algoritmo cromático y el primer algoritmo desarrollado por el autor, (N1). En el Caso II, se implementó el algoritmo genético, el algoritmo cromático y el segundo algoritmo desarrollado por el autor, (N2). En la tabla 3 se resume la información.

**Tabla 3.** Algoritmos implementados para cada caso

| ALGORITMO | CASO I | CASO II |
|---|---|---|
| AG | X | X |
| PSO | X | |
| AC | X | X |
| N1 | X | |
| N2 | | X |

A continuación, se presentan los parámetros adoptados para las metaheurísticas, el diseño de experimentos, el cálculo del tamaño de la muestra (o número de réplicas del experimento) y la forma de generar las instancias en cada caso.

### 3.5.1 CASO I

Se realizó un experimento de comparación para los cuatro algoritmos, para los cuales se fijaron los parámetros mostrados en la tabla 3.

**Tabla 4.** Parámetros usados para los algoritmos en el experimento (Caso I)

| Algoritmo genético | | | Algoritmo cromático | | | | | | Algoritmo N1 | | | | Algoritmo PSO | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Tamaño de población | Pc | Pm | N | AM | UM | MC | VMC | ARRAM | N | n recomb | λ | γ | # de partículas | φ1 | φ2 |
| 100 | 0,9 | 0,05 | 30 | 0,097 | 0,484 | 0,553 | 0,233 | 1 | 30 | 40 | 0,5 | 0,5 | 100 | 2,7 | 1,7 |

Los parámetros del algoritmo cromático fueron los recomendados por el autor (Sabie y Lopez 2011)**.** Para el caso del algoritmo genético, PSO y N1, se realizaron varios ensayos y se tomaron recomendaciones de los tutores de esta investigación.

En este caso, se utilizaron 45 instancias distintas: 15 de 10 clientes, 15 de 50 clientes y 15 de 100 clientes. Para la construcción de las instancias, se tomaron como base 45 instancias de Solomon (Homberger & Gehring, 1999; Solomon, 1987) para VRPTW de 25, 50 y 100 clientes, agregando los parámetros propios de esta variación del problema de ruteo de vehículos, utilizando los tiempos de servicio y descartando las ventanas de tiempo. Al no disponerse de instancias de 10 clientes, fue necesario tomar las instancias de 25 clientes de Solomon y seleccionar 10 clientes de forma aleatoria para su construcción. En el Anexo 1, se presenta la estructura de las instancias y la forma en que se generaron.

También se calculó el tamaño de la muestra, tomando como soporte el trabajo de (Montgomery 2004), el cual arrojó la necesidad de realizar 12 réplicas para este experimento (Anexo 2).

Así, al resolver 12 veces las 45 instancias con 4 algoritmos distintos, se realizaron un total de 2160 corridas con un tiempo fijo de 3 minutos por corrida.

### 3.5.2 CASO II

El diseño de este experimento fue posterior al experimento del Caso I, por tal motivo, optó por tomar las dos metaheurísticas con mejor desempeño en el Caso I (como se observará en el siguiente capítulo). Así, se realizó un experimento de comparación para los 3 algoritmos: cromático, genético y N2. Para los dos primeros, se fijaron los parámetros mostrados en la tabla 4. Por su parte, el algoritmo N2 sufrió un proceso de parametrización, el cual se describe en el Anexo 3.

**Tabla 5.** Parámetros usados para los algoritmos en el experimento (Caso II)

| Algoritmo genético | | | Algoritmo cromático | | | | | | Algoritmo N2 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Tamaño de población | Pc | Pm | N | AM | UM | MC | VMC | *ARRAM* | N | $Ngen_n$ | $t$ | *nran* | *ARRAM* |
| 2000 | 0,95 | 0,05 | 30 | 0,097 | 0,484 | 0,553 | 0,233 | 100 | 10 | 1500 | 30 | 0,07 | 100 |

A este respecto, se utilizaron 10 instancias: 5 de 50 clientes y 5 de 100 clientes. Para la construcción de las instancias, se adaptaron 10 instancias de Solomon (Homberger & Gehring, 1999; Solomon, 1987) para VRPTW de 50 y 100 clientes, agregando los parámetros propios del Caso II, utilizando los tiempos de servicio y descartando las ventanas de tiempo. En el Anexo 4, se presenta la estructura de las instancias y la forma en que se generaron.

También se calculó el tamaño de la muestra, tomando como soporte el trabajo de (Montgomery 2004), el cual arrojó la necesidad de realizar 10 réplicas para este experimento (Anexo 5).

Así, al resolver 10 veces las 10 instancias con 3 algoritmos distintos, se realizaron un total de 300 corridas con un tiempo fijo de 30 minutos por corrida.

# CAPÍTULO 4: RESULTADOS Y ANÁLISIS

En este capítulo, se presentan los resultados de los experimentos planteados en este trabajo (Caso I y Caso II), mostrando el método empleado para comparar resultados de instancias distintas y presentando e interpretando las distintas pruebas estadísticas aplicadas a los resultados.

## 4.1 ANÁLISIS DE RESULTADOS

Para proceder a realizar comparaciones entre resultados provenientes de instancias distintas, se llevaron los resultados a una distribución estándar usando la ecuación 54.

$$Z_i = \frac{x_{ij} - \bar{x}_i}{s_i} \tag{54}$$

Donde $x_{ij}$ representa el valor de la función objetivo obtenido en la instancia *i* en la réplica j, $\bar{x}_i$ es el promedio de las observaciones para la instancia i, $s_i$ es la desviación estándar de las observaciones de la instancia *i* y $Z_i$ es el valor normalizado.

Todos los análisis estadísticos de los resultados fueron realizados en el software estadístico Statgraphics Centurion XV v15.2.0.6 de la compañía Statpoint, Inc.

### 4.1.1 Análisis de comparación de algoritmos (Caso I)

Recapitulando, este caso encierra el problema de ruteo de vehículos con minimización de la pérdida de frescura de un producto perecedero. Se aplicaron cuatro (4) algoritmos (genético, cromático, PSO y N1) y se resolvieron 45 instancias (15 de 10 clientes, 15 de

50 clientes y 15 de 100 clientes), realizando 12 réplicas, para un total de 2160 corridas de 3 minutos cada una.

Los resultados obtenidos fueron resumidos en la tabla 6 para las instancias de 10 clientes. La información de las instancias de 50 y 100 clientes se encuentra condensada en las tablas 7 y 8, respectivamente. En cada tabla, se subraya con una línea el mejor promedio por instancia, calculado a partir de los resultados de las 12 corridas que cada algoritmo realizó en cada instancia. De igual manera, el mejor valor obtenido en cada instancia por cualquiera de los algoritmos es subrayado a doble línea. También, se muestra la desviación algoritmo – instancia, la desviación estándar general ($s_i$) y el promedio general ($\bar{x}_i$) por cada instancia. Otro dato importante que se incluye en el resumen por tamaño de instancias, es la frecuencia con la que cada algoritmo obtuvo los mejores resultados y los mejores promedios, como se observa en el gráfico 2.

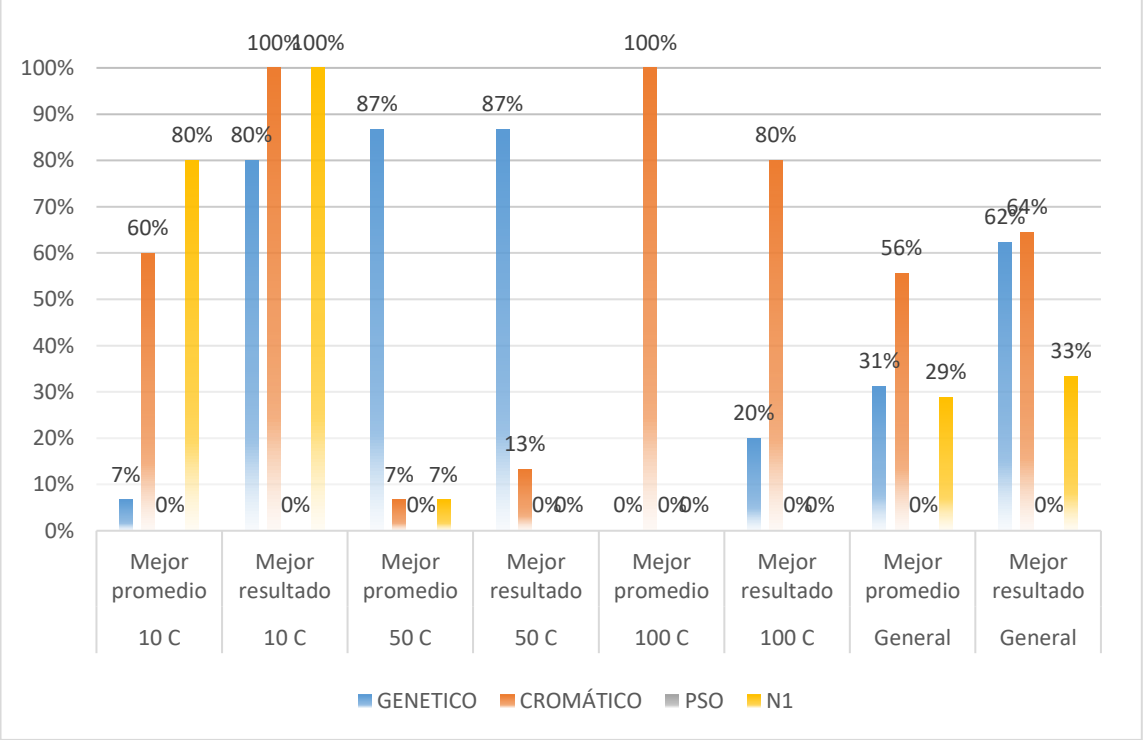



**Gráfico 2.** Frecuencia de mejores resultados y mejores promedios por algoritmo (Caso I)

En las instancias de 10 clientes (10 C), el algoritmo N1 fue superior en mejor promedio e igualó al algoritmo cromático en mejores soluciones, obteniendo el mejor promedio el 80% de las veces y el mejor resultado el 100% de las veces para las 15 instancias; en el segundo lugar se encuentra el algoritmo cromático, con 60% y 100% de las veces para mejor promedio y mejor resultado, respectivamente; el algoritmo genético ocupó el tercer lugar, con el mejor promedio el 7% de las veces y el mejor resultado encontrado el 80% de las veces; por último, se encuentra el algoritmo PSO, el cual no encontró mejores soluciones ni obtuvo mejores promedios en ninguna de las instancias de 10 clientes.

De manera análoga, se realiza este análisis para las 15 instancias de 50 clientes, encontrando mejor posicionado al algoritmo genético, seguido por el algoritmo cromático, el algoritmo N1 y, finalmente, el algoritmo PSO.

Por su parte, en las instancias de 100 clientes se obtuvieron resultados favorables para el algoritmo cromático tomando este enfoque de frecuencia (mejor promedio el 100% de las veces y mejor resultado el 80%), superando al algoritmo genético (0% en mejor promedio, 20% en mejor resultado) y a los algoritmos PSO y N1, que no tuvieron un buen desempeño en estas instancias de gran tamaño.

En el análisis general para las 45 instancias, esta perspectiva de frecuencia posiciona en primer lugar al algoritmo cromático (mejor promedio: 56%, mejor resultado: 64%), seguido por el algoritmo genético (mejor promedio: 31%, mejor resultado: 62%), y el algoritmo N1 (mejor promedio: 29%, mejor resultado: 33%), dejando el último lugar para el algoritmo PSO, el cual obtuvo los resultados menos favorables de acuerdo a este enfoque.

**Tabla 6.** Resumen de los resultados obtenidos por los algoritmos para las instancias de 10 clientes (Caso I)

| INSTANCIA i | ALGORITMO | | | | | | | | | | | | Min(Prom) para la instancia i | Min. de F para la instancia i | $\bar{x}_i$ | $S_i$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | GENÉTICO | | | CROMÁTICO | | | PSO | | | N1 | | | | | | |
| | Prom. de F | Mín. de F | Desvest de F | Prom. de F | Mín. de F | Desvest de F | Prom. de F | Mín. de F | Desvest de F | Prom. de F | Mín. de F | Desvest de F | | | | |
| 1 | 31,82 | 31,48 | 0,34 | 31,48 | 31,48 | 0,00 | 32,56 | 32,02 | 0,28 | 31,57 | 31,48 | 0,16 | 31,48 | 31,48 | 31,86 | 0,49 |
| 2 | 44,45 | 44,45 | 0,00 | 44,48 | 44,45 | 0,10 | 45,61 | 45,35 | 0,25 | 44,45 | 44,45 | 0,00 | 44,45 | 44,45 | 44,74 | 0,52 |
| 3 | 55,66 | 55,33 | 0,26 | 55,37 | 55,33 | 0,14 | 56,56 | 55,56 | 0,46 | 55,33 | 55,33 | 0,00 | 55,33 | 55,33 | 55,73 | 0,57 |
| 4 | 36,57 | 36,52 | 0,08 | 36,52 | 36,52 | 0,00 | 37,60 | 37,30 | 0,25 | 36,52 | 36,52 | 0,00 | 36,52 | 36,52 | 36,80 | 0,48 |
| 5 | 24,24 | 24,23 | 0,01 | 24,24 | 24,23 | 0,01 | 24,30 | 24,25 | 0,02 | 24,23 | 24,23 | 0,00 | 24,23 | 24,23 | 24,25 | 0,03 |
| 6 | 10,99 | 10,52 | 0,63 | 10,11 | 9,90 | 0,31 | 11,61 | 10,59 | 0,50 | 10,21 | 9,90 | 0,32 | 10,11 | 9,90 | 10,73 | 0,76 |
| 7 | 31,54 | 31,50 | 0,04 | 31,51 | 31,50 | 0,03 | 31,68 | 31,64 | 0,02 | 31,51 | 31,50 | 0,03 | 31,51 | 31,50 | 31,56 | 0,08 |
| 8 | 49,97 | 49,76 | 0,29 | 49,80 | 49,76 | 0,15 | 50,88 | 50,44 | 0,24 | 49,84 | 49,76 | 0,20 | 49,80 | 49,76 | 50,12 | 0,50 |
| 9 | 32,26 | 31,82 | 0,44 | 32,13 | 31,82 | 0,31 | 33,49 | 32,78 | 0,45 | 31,92 | 31,82 | 0,29 | 31,92 | 31,82 | 32,45 | 0,72 |
| 10 | 5,85 | 5,73 | 0,08 | 5,76 | 5,68 | 0,06 | 6,20 | 5,97 | 0,15 | 5,73 | 5,68 | 0,03 | 5,73 | 5,68 | 5,88 | 0,21 |
| 11 | 25,25 | 25,18 | 0,10 | 25,19 | 25,18 | 0,01 | 25,45 | 25,26 | 0,10 | 25,19 | 25,18 | 0,01 | 25,19 | 25,18 | 25,27 | 0,13 |
| 12 | 21,82 | 21,81 | 0,02 | 21,81 | 21,81 | 0,01 | 21,95 | 21,83 | 0,05 | 21,81 | 21,81 | 0,01 | 21,81 | 21,81 | 21,85 | 0,07 |
| 13 | 25,58 | 25,57 | 0,03 | 25,57 | 25,57 | 0,00 | 25,64 | 25,59 | 0,03 | 25,57 | 25,57 | 0,00 | 25,57 | 25,57 | 25,59 | 0,03 |
| 14 | 11,64 | 11,63 | 0,01 | 11,62 | 11,62 | 0,00 | 11,65 | 11,64 | 0,01 | 11,62 | 11,62 | 0,00 | 11,62 | 11,62 | 11,63 | 0,01 |
| 15 | 13,04 | 12,69 | 0,21 | 12,89 | 12,69 | 0,22 | 13,70 | 13,43 | 0,25 | 12,71 | 12,69 | 0,08 | 12,71 | 12,69 | 13,09 | 0,42 |
| # de veces con mejor promedio | 1 | | | 9 | | | 0 | | | 12 | | | | | | |
| # de veces con mejor resultado | | 12 | | | 15 | | | 0 | | | 15 | | | | | |

**Tabla 7.** Resumen de los resultados obtenidos por los algoritmos para las instancias de 50 clientes (Caso I)

| INSTANCIA i | ALGORITMO | | | | | | | | | | | | Min(prom) para la instancia i | Min. de F para la instancia i | $\bar{x}_i$ | $S_i$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | GENÉTICO | | | CROMÁTICO | | | PSO | | | N1 | | | | | | |
| | Prom. de F | Mín. de F | Desvest de F | Prom. de F | Mín. de F | Desvest de F | Prom. de F | Mín. de F | Desvest de F | Prom. de F | Mín. de F | Desvest de F | | | | |
| 1 | 209,89 | 209,12 | 0,49 | 210,70 | 210,10 | 0,38 | 215,34 | 214,77 | 0,32 | 213,03 | 212,27 | 0,54 | 209,89 | 209,12 | 212,24 | 2,19 |
| 2 | 280,83 | 280,12 | 0,54 | 282,10 | 280,99 | 0,77 | 287,10 | 286,54 | 0,38 | 284,62 | 283,95 | 0,46 | 280,83 | 280,12 | 283,66 | 2,49 |
| 3 | 307,64 | 306,76 | 0,62 | 308,40 | 307,60 | 0,44 | 312,21 | 311,16 | 0,63 | 310,31 | 309,93 | 0,25 | 307,64 | 306,76 | 309,64 | 1,86 |
| 4 | 280,48 | 265,29 | 12,23 | 273,54 | 267,47 | 3,45 | 303,94 | 300,75 | 1,59 | 290,31 | 283,08 | 4,44 | 273,54 | 265,29 | 287,07 | 13,27 |
| 5 | 360,55 | 353,51 | 3,48 | 369,77 | 365,86 | 2,76 | 401,59 | 399,38 | 1,97 | 384,87 | 382,10 | 1,98 | 360,55 | 353,51 | 379,20 | 15,94 |
| 6 | 201,67 | 193,89 | 4,46 | 215,96 | 208,91 | 4,49 | 261,14 | 255,33 | 3,02 | 233,75 | 221,46 | 5,82 | 201,67 | 193,89 | 228,13 | 22,85 |
| 7 | 91,05 | 87,65 | 4,80 | 94,48 | 90,86 | 1,69 | 110,07 | 108,10 | 1,16 | 104,17 | 101,21 | 1,26 | 91,05 | 87,65 | 99,94 | 8,08 |
| 8 | 174,22 | 170,14 | 3,20 | 184,84 | 175,54 | 4,16 | 211,28 | 209,07 | 1,56 | 197,29 | 191,86 | 2,01 | 174,22 | 170,14 | 191,91 | 14,28 |
| 9 | 116,73 | 110,58 | 4,02 | 132,47 | 124,06 | 6,03 | 179,17 | 171,80 | 3,27 | 152,79 | 146,56 | 3,53 | 116,73 | 110,58 | 145,29 | 23,99 |
| 10 | 277,05 | 271,93 | 2,49 | 277,55 | 253,47 | 9,69 | 292,07 | 286,40 | 3,07 | 283,20 | 277,41 | 4,52 | 277,05 | 253,47 | 282,47 | 8,23 |
| 11 | 171,58 | 160,81 | 6,49 | 192,69 | 175,29 | 8,65 | 256,00 | 247,50 | 4,38 | 217,40 | 207,75 | 7,43 | 171,58 | 160,81 | 209,42 | 32,44 |
| 12 | 156,13 | 155,04 | 0,54 | 156,06 | 154,73 | 0,92 | 158,47 | 156,71 | 0,82 | 155,64 | 155,00 | 0,50 | 155,64 | 154,73 | 156,57 | 1,32 |
| 13 | 136,11 | 129,50 | 4,21 | 149,22 | 138,15 | 5,79 | 199,08 | 191,17 | 4,60 | 175,65 | 160,05 | 7,83 | 136,11 | 129,50 | 165,01 | 25,17 |
| 14 | 100,93 | 95,02 | 4,38 | 118,38 | 107,34 | 5,01 | 168,67 | 163,41 | 3,51 | 136,94 | 127,22 | 6,15 | 100,93 | 95,02 | 131,23 | 25,79 |
| 15 | 175,87 | 172,03 | 2,55 | 183,63 | 178,58 | 2,46 | 207,50 | 203,58 | 1,55 | 192,47 | 187,94 | 2,43 | 175,87 | 172,03 | 189,87 | 12,08 |
| # de veces con mejor promedio | 13 | | | 1 | | | 0 | | | 1 | | | | | | |
| # de veces con mejor resultado | | 13 | | | 2 | | | 0 | | | 0 | | | | | |

**Tabla 8.** Resumen de los resultados obtenidos por los algoritmos para las instancias de 100 clientes (Caso I)

| INSTANCIA i | ALGORITMO | | | | | | | | | | | | Min(prom) para la instancia i | Min. de F para la instancia i | $\bar{x}_i$ | $S_i$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | GENÉTICO | | | CROMÁTICO | | | PSO | | | N1 | | | | | | |
| | Prom. de F | Mín. de F | Desvest de F | Prom. de F | Mín. de F | Desvest de F | Prom. de F | Mín. de F | Desvest de F | Prom. de F | Mín. de F | Desvest de F | | | | |
| 1 | 335,61 | 331,01 | 2,81 | 324,06 | 314,80 | 5,30 | 344,49 | 336,40 | 3,02 | 338,94 | 335,44 | 2,38 | 324,06 | 314,80 | 335,77 | 8,30 |
| 2 | 753,53 | 729,53 | 10,76 | 719,51 | 700,77 | 17,23 | 764,34 | 742,27 | 10,50 | 743,83 | 726,49 | 9,92 | 719,51 | 700,77 | 745,30 | 20,63 |
| 3 | 257,75 | 232,45 | 9,83 | 245,44 | 237,49 | 4,00 | 271,60 | 260,83 | 4,03 | 264,93 | 257,94 | 2,77 | 245,44 | 232,45 | 259,93 | 11,31 |
| 4 | 618,03 | 587,05 | 14,46 | 600,68 | 584,93 | 16,69 | 646,33 | 629,50 | 7,04 | 632,73 | 620,81 | 7,82 | 600,68 | 584,93 | 624,44 | 20,85 |
| 5 | 841,66 | 819,51 | 9,44 | 786,58 | 775,79 | 6,91 | 870,84 | 863,47 | 4,68 | 838,92 | 827,27 | 6,96 | 786,58 | 775,79 | 834,50 | 31,46 |
| 6 | 454,82 | 439,20 | 7,49 | 451,74 | 426,70 | 10,72 | 471,82 | 466,12 | 2,57 | 463,28 | 453,76 | 5,87 | 451,74 | 426,70 | 460,42 | 10,59 |
| 7 | 617,47 | 602,98 | 10,65 | 614,66 | 581,86 | 20,99 | 643,15 | 630,36 | 8,96 | 637,32 | 619,69 | 7,81 | 614,66 | 581,86 | 628,15 | 17,81 |
| 8 | 493,17 | 439,74 | 26,24 | 452,56 | 428,64 | 13,07 | 535,56 | 528,03 | 5,11 | 512,79 | 504,32 | 6,18 | 452,56 | 428,64 | 498,52 | 34,13 |
| 9 | 309,94 | 299,82 | 3,67 | 298,76 | 292,93 | 4,39 | 318,71 | 311,40 | 3,47 | 312,51 | 306,10 | 3,52 | 298,76 | 292,93 | 309,98 | 8,16 |
| 10 | 867,45 | 861,71 | 4,40 | 827,45 | 820,16 | 9,87 | 875,17 | 866,89 | 6,29 | 864,41 | 856,36 | 5,56 | 827,45 | 820,16 | 858,62 | 19,76 |
| 11 | 741,81 | 717,77 | 8,56 | 717,54 | 705,16 | 8,02 | 750,50 | 743,63 | 4,05 | 739,44 | 726,50 | 7,65 | 717,54 | 705,16 | 737,32 | 14,16 |
| 12 | 319,35 | 288,72 | 10,80 | 312,23 | 301,21 | 7,26 | 341,83 | 337,88 | 2,32 | 325,18 | 318,86 | 4,97 | 312,23 | 288,72 | 324,65 | 12,98 |
| 13 | 446,72 | 409,13 | 26,17 | 427,26 | 412,79 | 8,80 | 503,19 | 495,44 | 3,84 | 475,21 | 463,36 | 7,65 | 427,26 | 409,13 | 463,10 | 32,25 |
| 14 | 312,29 | 305,38 | 3,10 | 301,16 | 294,79 | 4,60 | 319,32 | 313,33 | 2,29 | 314,36 | 312,11 | 1,57 | 301,16 | 294,79 | 311,78 | 7,35 |
| 15 | 253,84 | 250,23 | 1,83 | 246,56 | 243,32 | 1,83 | 257,92 | 254,04 | 1,57 | 255,81 | 253,79 | 1,02 | 246,56 | 243,32 | 253,53 | 4,59 |
| # de veces con mejor promedio | 0 | | | 15 | | | 0 | | | 0 | | | | | | |
| # de veces con mejor resultado | | 3 | | | 12 | | | 0 | | | 0 | | | | | |

En este punto, se verificaron los supuestos correspondientes para realizar un análisis de varianza, sin embargo, no se cumplió con el supuesto de normalidad en los residuos de los datos. Por tal motivo, se optó por utilizar la prueba de medianas de mood, la cual obedece a un método no paramétrico en el que podemos basar nuestro análisis.

A continuación, se observa el resultado de esta prueba:

**Tabla 9.** Prueba de la Mediana de Mood para Z por Algoritmo (Caso I)

| *Algoritmo* | *Tamaño de Muestra* | *n<=* | *n>* | *Mediana* | *LC inferior 95,0%* | *LC superior 95,0%* |
|---|---|---|---|---|---|---|
| N1 | 540 | 221 | 319 | 0,125116 | 0,0102231 | 0,183167 |
| CROMATICO | 540 | 503 | 37 | -0,708975 | -0,76345 | -0,691748 |
| GENETICO | 540 | 350 | 190 | -0,558999 | -0,579195 | -0,450128 |
| PSO | 540 | 6 | 534 | 1,30889 | 1,27919 | 1,35197 |
| Estadístico = 983,6 | | Valor-P = 0,0 | | | Total n = 2160 | Gran mediana = -0,119413 |

En esta prueba se evalúa la hipótesis de que las medianas de las muestras correspondientes a cada algoritmo son iguales. Esto se lleva a cabo contando el número de observaciones de cada muestra, a cada lado de la mediana global, la cual es igual a -0,119413. Puesto que el valor-P para la prueba de chi-cuadrada es menor que 0,05, las medianas de las muestras son significativamente diferentes con un nivel de confianza del 95,0%. Puede observarse que los intervalos del 95% de confianza para la mediana de cada algoritmo son mutuamente excluyentes, siendo el de mejor desempeño aquel algoritmo cuyo intervalo se encuentra más a la izquierda.

A partir de la prueba de mediana de mood, se infiere que, de todos los algoritmos comparados, el algoritmo cromático es el que presenta un mejor desempeño en términos de calidad de soluciones, seguido por el algoritmo genético, dejando en tercer lugar el algoritmo híbrido y, por último, el algoritmo de optimización por enjambre de partículas (PSO).

Para reforzar el análisis, se presenta un gráfico de caja y bigotes (gráfico 3), en el cual se agrega un corte a cada gráfico para permitir la comparación entre las medias muestrales a un nivel de confianza del 95%. Si dos de los cortes o "muescas" no se traslapan, existe una diferencia estadística significativa entre las medianas los dos grupos.

En nuestro caso, puede observarse que ninguna de las muescas se traslapa, entonces, es posible afirmar que existen diferencias significativas entre las medianas de todos los algoritmos con un nivel de confianza del 95%. Más aún, el gráfico de caja y bigotes permite reafirmar los resultados y conclusiones de la prueba de medianas de mood.

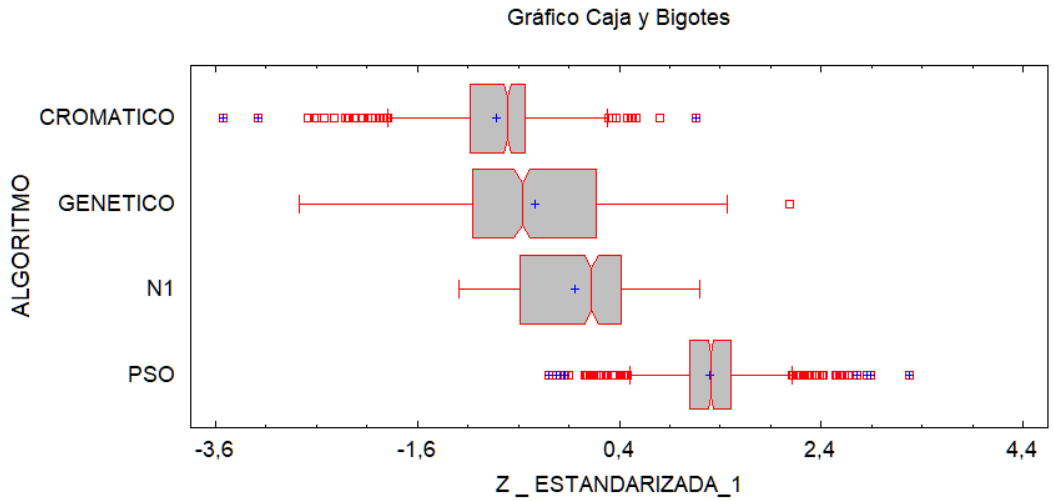



**Gráfico 3.** Gráfico de caja y bigotes para la comparación de algoritmos (Caso I)

### 4.1.2 Análisis de comparación de algoritmos (Caso II)

Para solucionar el problema ruteo de vehículos refrigerados y no refrigerados para la entrega de múltiples productos perecederos con optimización de la frescura, se tomaron los 2 algoritmos que presentaron mejores resultados en términos de calidad de soluciones para el Caso I y se propuso una nueva alternativa para la solución de problemas de optimización combinatoria con codificación permutada, por lo cual, se implementaron tres (3) algoritmos: el algoritmo cromático, el algoritmo genético y el algoritmo N2.

Además de esto, se optó por reducir el número de instancias con respecto al Caso I, resolviendo sólo 10 instancias con cada algoritmo, las cuales se dividen en 5 instancias de 50 clientes y 5 instancias de 100 clientes. Esta reducción de instancias, se sustenta en la necesidad de aumentar el tiempo dispuesto para cada corrida en aras de conocer el comportamiento de los algoritmos cuando se les proporciona más tiempo para su ejecución. Así, en el Caso II, el tiempo fijado para cada corrida es de 30 minutos. Resumiendo, se utilizaron 3 algoritmos y se resolvieron 5 instancias de 50 clientes y 5 de 100 clientes. Se realizaron 10 réplicas del experimento, para un total de 300 corridas de 30 minutos cada una.

Los resultados del experimento se resumen en la tabla 10, en la cual se resalta el mejor promedio (subrayado a una línea) y el mejor valor obtenido (subrayado a dos líneas) por los algoritmos en cada instancia. Además, para cada instancia se muestra la desviación estándar general ($s_i$), el promedio general ($\bar{x}_i$) y la desviación por algoritmo. También, se incluye la frecuencia con la que cada algoritmo obtuvo los mejores resultados y los mejores promedios (Gráfico 4). Se observa una total favorabilidad del algoritmo N2 en torno a la perspectiva de frecuencia de mejores promedios y mejores resultados.

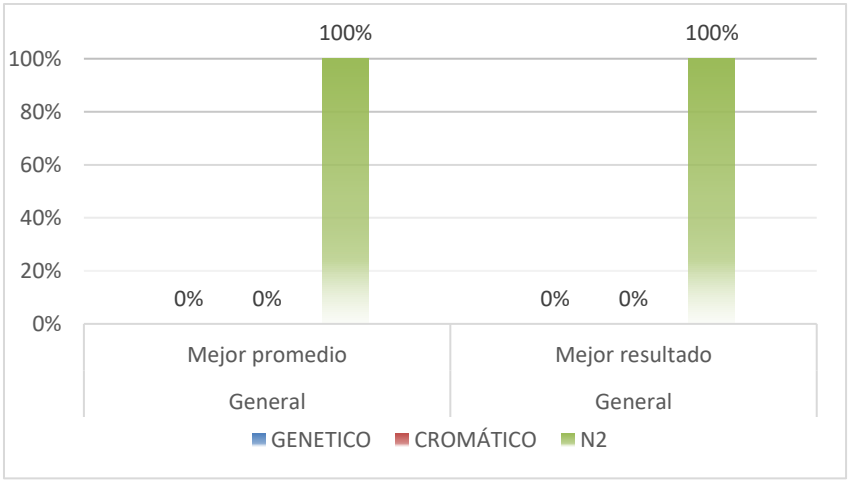



**Gráfico 4.** Frecuencia de mejores resultados y mejores promedios por algoritmo (Caso II)

**Tabla 10.** Resumen general de los resultados obtenidos por los algoritmos (Caso II)

| INSTANCIA (i, # de clientes) | ALGORITMO | | | | | | | | | Min(prom) para la instancia i | Min. de F para la instancia i | $\bar{x}_i$ | $S_i$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | GENÉTICO | | | CROMÁTICO | | | N2 | | | | | | |
| | Prom. de F | Mín. de F | Desvest de F | Prom. de F | Mín. de F | Desvest de F | Prom. de F | Mín. de F | Desvest de F | | | | |
| (1, 50) | 2149,1 | 2105,8 | 24,9 | 2353,1 | 2302,5 | 25,5 | 2115,5 | 2102,1 | 8,3 | 2115,5 | 2102,1 | 2205,9 | 108,7 |
| (2, 50) | 3040,0 | 3009,2 | 25,6 | 3320,3 | 3234,6 | 72,3 | 2995,5 | 2990,0 | 5,6 | 2995,5 | 2990,0 | 3118,6 | 152,4 |
| (3, 50) | 2057,4 | 2031,8 | 21,7 | 2539,6 | 2431,0 | 69,8 | 2017,1 | 1999,0 | 11,9 | 2017,1 | 1999,0 | 2204,7 | 245,0 |
| (4, 50) | 6524,0 | 6510,7 | 6,8 | 6874,1 | 6744,6 | 95,1 | 6508,7 | 6502,6 | 3,7 | 6508,7 | 6502,6 | 6635,6 | 179,7 |
| (5, 50) | 1416,5 | 1292,1 | 96,1 | 1749,7 | 1650,9 | 55,4 | 1196,2 | 1189,6 | 5,4 | 1196,2 | 1189,6 | 1454,2 | 239,6 |
| (1, 100) | 9903,0 | 9524,5 | 160,1 | 9643,4 | 9461,6 | 176,4 | 7816,5 | 7754,1 | 44,2 | 7816,5 | 7754,1 | 9120,9 | 953,9 |
| (2, 100) | 7367,9 | 7215,2 | 68,1 | 7340,5 | 7089,8 | 159,6 | 6051,3 | 6007,2 | 25,6 | 6051,3 | 6007,2 | 6919,9 | 632,4 |
| (3, 100) | 3892,9 | 3813,9 | 60,0 | 3989,3 | 3743,9 | 140,9 | 2780,3 | 2743,2 | 25,1 | 2780,3 | 2743,2 | 3554,2 | 564,7 |
| (4, 100) | 5796,7 | 5621,8 | 107,4 | 6302,0 | 6137,7 | 116,8 | 4225,1 | 4132,6 | 46,0 | 4225,1 | 4132,6 | 5441,3 | 904,2 |
| (5, 100) | 7614,7 | 7493,5 | 64,1 | 7677,4 | 6984,3 | 387,4 | 5539,2 | 5467,6 | 38,8 | 5539,2 | 5467,6 | 6943,8 | 1034,1 |
| # de veces con mejor promedio | 0 | | | 0 | | | 10 | | | | | | |
| # de veces con mejor resultado | | 0 | | | 0 | | | 10 | | | | | |

Para este caso, se intentó verificar los supuestos correspondientes para realizar el análisis de varianza, sin embargo, al igual que en el Caso I, los datos de las observaciones no cumplieron con el supuesto de normalidad de los residuos. Por esta razón, se utilizó un método estadístico no paramétrico: la prueba de medianas de mood.

**Tabla 11.** Prueba de la Mediana de Mood para Z por Algoritmo (Caso II)

| *Algoritmo* | *Tamaño de Muestra* | *n<=* | *n>* | *Mediana* | *LC inferior 95,0%* | *LC superior 95,0%* |
|---|---|---|---|---|---|---|
| CROMATICO | 100 | 2 | 98 | 1,00045 | 0,872455 | 1,1236 |
| GENETICO | 100 | 48 | 52 | 0,264528 | -0,356237 | 0,486742 |
| N2 | 100 | 100 | 0 | -1,19698 | -1,31925 | -0,941969 |
| Estadístico = 192,32 | | Valor-P = 0,0 | | | Total n = 300 | Gran mediana = 0,191055 |

Se evalúa la hipótesis de que las medianas de las muestras correspondientes a cada algoritmo son iguales. Puesto que el valor-P para la prueba de chi-cuadrada es menor que 0,05, las medianas de las muestras son significativamente diferentes con un nivel de confianza del 95,0%. Puede observarse que los intervalos del 95% de confianza para la mediana de cada algoritmo son mutuamente excluyentes, siendo el de mejor desempeño aquel algoritmo cuyo intervalo se encuentra más a la izquierda. De los resultados de esta prueba, se infiere que el algoritmo N2 es presenta un mejor desempeño en términos de calidad de soluciones, seguido por el algoritmo genético, y, por último, el algoritmo cromático. Para corroborar los resultados de la prueba, se presenta un gráfico de caja y bigotes (gráfico 5), donde se observa que las muescas no se traslapan.

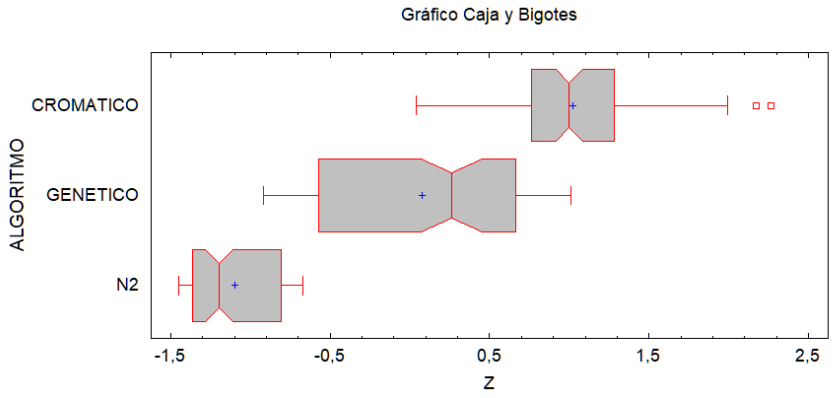



**Gráfico 5.** Gráfico de caja y bigotes para la comparación de algoritmos (Caso II)

# CAPITULO 5: CONCLUSIONES Y RECOMENDACIONES

En este capítulo, se presentan las conclusiones para los dos casos presentados en este trabajo, se resaltan algunas consideraciones y se brindan recomendaciones basadas en los resultados de la investigación para la realización de trabajos futuros.

Para la presente investigación, la meta establecida se centraba en la formulación de un modelo matemático para el problema de ruteo con dos tipos de vehículos (refrigerado y no refrigerado) para representar la minimización de la pérdida de frescura en la entrega de productos perecederos, y el diseño de un algoritmo metaheurístico para su solución; sin embargo, en aras de formular el modelo matemático para múltiples productos perecederos (Caso II), se hizo necesario formular un modelo mono producto que, posteriormente, pudiera ser extendido (Caso I). De esta forma, fue posible el desarrollo de dos (2) modelos matemáticos para los cuales se diseñaron (2) algoritmos independientes: N1 para el Caso I y N2 para el Caso II. Luego de esto, se realizaron 2 experimentos, comparando los algoritmos diseñados con otras metaheurísticas seleccionadas y realizando los análisis estadísticos respectivos.

A partir del análisis estadístico de cada experimento, se obtuvieron conclusiones relacionadas con el desempeño de los algoritmos comparados en términos de calidad de las soluciones generadas. En este orden de ideas, se encontró que, para ambos

experimentos, existen diferencias estadísticamente significativas entre la calidad de los resultados obtenidos por los algoritmos comparados. Adicionalmente, la frecuencia en que cada algoritmo obtuvo las mejores soluciones y los mejores promedios en cada instancia fue considerada en el análisis.

Así, para el Caso I fueron comparados cuatro (4) algoritmos: genético, PSO, cromático y algoritmo N1; resolviendo 45 instancias distribuidas así: 15 de 10 clientes, 15 de 50 clientes y 15 de 100 clientes; con 12 réplicas del experimento y corridas de 3 minutos. Del análisis estadístico del experimento de comparación, se tiene que:

> ***Conclusión 1 (Caso I):*** *En calidad de respuesta, el algoritmo cromático demuestra obtener mejores resultados frente a los algoritmos comparados.*

El algoritmo cromático es seguido por el algoritmo genético, en tercer tenemos al algoritmo N1, y, por último, encontramos al algoritmo PSO.

También, de acuerdo a los resultados del enfoque de frecuencias, el algoritmo cromático fue el mejor posicionado entre los algoritmos comparados, con mejor promedio el 56% de las veces y mejor resultado el 64% de las veces en 45 instancias; seguido por el algoritmo genético, que, si bien no fue el mejor, obtuvo los mejores promedios y loa mejores resultados en las instancias de 50 clientes; el algoritmo N1, el cual superó a los demás algoritmos en la obtención de mejores promedios e igualó al algoritmo cromático en mejores resultados para las instancias de 10 clientes; y el algoritmo de optimización por enjambre de partículas, PSO, el cual no obtuvo mejores resultados ni mejores promedios en ninguna de las 45 instancias.

> ***Conclusión 2 (Caso I):*** *En términos de variabilidad, genético y N1 muestran ser los algoritmos con menor dispersión.*

Esto se sustenta en el gráfico de caja y bigotes para este caso. Esto nos permite intuir que, para esta implementación, los algoritmos mencionados ofrecen respuestas muy similares respecto al valor de la función objetivo en la mayoría de los casos. Por su parte, el algoritmo cromático y PSO muestran mayor variabilidad, por lo que las respuestas obtenidas por estos algoritmos de un mismo problema pueden estar un poco alejadas entre sí.

Luego de realizar este experimento, se destacan algunas consideraciones y recomendaciones del Caso I:

- Del modelo y la implementación de metaheurísticas: la innovación del modelo formulado, radica en su simplicidad y en una disminución importante en la cantidad de restricciones con respecto a otros modelos similares presentados en la literatura. Se resalta que, hasta donde al autor le consta, este problema no había sido resuelto usando metaheurísticas.
- Del modelo matemático mono producto: en trabajos futuros puede estudiarse a fondo el cálculo de los valores para los parámetros alfa y beta de acuerdo a las características del producto a transportar.
- Del algoritmo genético: para esta metaheurística, podrían estudiarse cambios en los valores escogidos para sus parámetros, como, por ejemplo, el aumento del tamaño de población. (*)
- Del algoritmo PSO: se recomienda realizar una optimización previa de los parámetros de este algoritmo para su aplicación en otras investigaciones.
- Del algoritmo cromático: una de las principales contribuciones de este trabajo es la implementación del algoritmo cromático para resolver el problema planteado,

destacando que esta metaheurística no había sido puesta a prueba en el problema de ruteo de vehículos. Se recomienda potencializar la implementación de este algoritmo en el VRP y en otros problemas de optimización combinatoria.

- Del algoritmo N1: En este algoritmo, fueron empleados algunos operadores del algoritmo cromático y algoritmo genético. N1 mostró ser eficaz en términos de calidad de soluciones para problemas con pocos clientes, sin embargo, pudo observarse que cuando se aumenta el número de clientes, la metaheurística cromática pura, al igual que el algoritmo genético, resultan ser más eficiente que este algoritmo. Se recomienda optimizar sus parámetros, o bien sea, estudiar los aspectos a mejorar y realizar una reestructuración del mismo, incorporando nuevas funciones enfocadas más específicamente al problema de ruteo de vehículos, que le permitan lograr un mejor desempeño. (*)
- Del tiempo fijado para cada corrida: puede considerarse la realización de una prueba experimental con mayores tiempos de ejecución por corrida, para estudiar más a fondo el comportamiento de estos algoritmos en otros entornos. (*)

Ahora bien, para el Caso II, se tuvo la oportunidad de mejorar algunos aspectos en la implementación de los algoritmos y en el diseño experimental: (i) se aumentó de 100 a 1000 el tamaño de población para el algoritmo genético. (ii) se diseñó un nuevo algoritmo con innovación en operadores, estructura, y forma de implementación. (iii) se reestructuró la codificación de las soluciones en aras de facilitar la implementación del nuevo algoritmo. (iv) se disminuyó el número de instancias a resolver y se aumentó el tiempo para cada corrida en el experimento: de 3 a 30 minutos (v) se eliminaron los dos algoritmos

con resultados menos favorables en el caso I: PSO y N1. (vi) se realizó una parametrización de tipo simple para la nueva propuesta algorítmica (N2).

En este caso, se evaluaron tres (3) algoritmos: genético, cromático y N2; resolviendo 5 instancias de 50 clientes y 5 de 100 clientes; realizando 10 réplicas, con lo cual se tienen 300 corridas de 30 minutos cada una. Del análisis estadístico del experimento de comparación, se tiene que:

> ***Conclusión 3 (Caso II):*** *En calidad de respuesta, el algoritmo N2 demuestra obtener mejores resultados frente a los algoritmos comparados.*

Esta vez, el algoritmo genético se encuentra en segundo lugar, siendo relegando el algoritmo cromático al tercer lugar.

Desde la perspectiva de frecuencia de mejores promedios y mejores soluciones, los resultados no fueron menos alentadores: el algoritmo N1 encontró el 100% de las mejores soluciones y el 100% de los mejores promedios en todas las instancias del experimento, mientras que los otros dos algoritmos obtuvieron 0% en ambos aspectos.

> ***Conclusión 4 (Caso II):*** *En términos de variabilidad, genético y N2 muestran ser los algoritmos con menor dispersión.*

En esta ocasión, se resalta que el algoritmo cromático tuvo solo 2 datos atípicos en sus resultados, sin embargo, esto no deja de afectar su variabilidad si se tiene en cuenta la poca dispersión presentada en los resultados de los otros dos algoritmos.

Es importante resaltar el cambio de roles entre los algoritmos genético y cromático para el Caso II con respecto al Caso I, donde el cambio en el parámetro “tamaño de población” del algoritmo genético pudo resultar en una mejora en su desempeño, sin embargo, no es posible realizar esta afirmación con toda certeza debido a que estamos tratando con dos problemas con características que no son del todo iguales.

En general, se encontró que, para el Caso I, el algoritmo cromático tuvo los mejores resultados, mientras que el algoritmo N2 fue quien se destacó en el Caso II. Puede inferirse que la optimización de parámetros y la estructura algorítmica planteada en para el algoritmo (N2), lograron cumplir con el objetivo general de esta investigación de diseñar una meta heurística para optimizar la frescura en el problema de ruteo de vehículos refrigerados y de tipo general para la entrega de múltiples productos perecederos. De esta forma, el algoritmo N2 queda planteado y perfilado como una nueva y prometedora herramienta metaheurística que no debería ser limitada a la resolución de problemas de ruteo de vehículos. Se recomienda la exploración de iniciativas entorno a nuevas implementaciones de este algoritmo para la solución de otros problemas de optimización combinatoria.

## BIBLIOGRAFÍA


Amorim, P., & Almada-Lobo, B. (2014). The impact of food perishability issues in the vehicle routing problem. *Computers & Industrial Engineering*, *67*(1), 223–233. https://doi.org/http://dx.doi.org/10.1016/j.cie.2013.11.006

Arrieta, M., & López, E. (2013). *Diseño de un algoritmo de optimización para el problema de asignación de horarios en la Universidad de Córdoba*. Universidad de Córdoba, Montería, Colombia.

Ballou, R. H. (2004). *Logística. Administración de la cadena de suministro* (Quinta Edi). México: PEARSON EDUCACIÓN.

Barnhart, C., Johnson, E. L., Nemhauser, G. L., Savelsbergh, M. W. P., & Vance, P. H. (1998). Branch-and-price: Column generation for solving huge integer programs. *Operations Research*, *46*(3), 316–329. https://doi.org/10.1287/opre.46.3.316

Batista, B. M., & Glover, F. (2006). Introducción a la Búsqueda Tabú, *3*, 1–36.

Baykasoglu, A., Ozbakir, L., & Tapkan, P. (2007). Artificial bee colony algorithm and its application to generalized assignment problem. *Swarm Intelligence: Focus on Ant and Particle Swarm Optimization*, 113–144.

Clarke, G., & Wright, J. W. (1964). Scheduling of Vehicles from a Central Depot to a Number of Delivery Points. *Operations Research*, *12*(4), 568–581. https://doi.org/10.1287/opre.12.4.568

Clerc, M. (2004). Discrete particle swarm optimization, illustrated by the traveling salesman problem. In *New optimization techniques in engineering* (pp. 219–239). Springer.

Cordeau, J.-F., Gendreau, M., Laporte, G., Potvin, J.-Y., & Semet, F. (2002). A guide to

vehicle routing heuristics. *Journal of the Operational Research Society*, *53*(5), 512–522. https://doi.org/10.1057/palgrave/jors/2601319

Dantzig, G. B., & Ramser, J. H. (1959). The Truck Dispatching Problem. *Management Science*, *6*(1), 80–91. https://doi.org/10.1287/mnsc.6.1.80

DasGupta, D. (1993). *An overview of artificial immune systems and their applications*. Springer.

De Castro, L. N., & Timmis, J. (2002). *Artificial immune systems: a new computational intelligence approach*. Springer Science & Business Media.

de Keizer, M., Haijema, R., Bloemhof, J. M., & van der Vorst, J. G. A. J. (2015). Hybrid optimization and simulation to design a logistics network for distributing perishable products. *Computers & Industrial Engineering*, *88*, 26–38.

DNP. (2016). *Pérdida y Desperdicio*. *DNP - Colombia* (Vol. 39).

Dorigo, M., & Stützle, T. (2004). Ant Colony Optmization, Massachusetts Institute of Technology.

Dyer, M., & Stougie, L. (2006). Computational complexity of stochastic programming problems. *Mathematical Programming*, *106*(3), 423–432.

Espín, J. M. G. (2004). Estrategias de innovación en el sector hortofrutícola español y en las empresas encargadas de la logística y transporte de estos productos perecederos. *Papeles de Geografía*, (39), 81–117.

Fisher, M. L., & Jaikumar, R. (1981). A generalized assignment heuristic for vehicle routing. *Networks*, *11*(2), 109–124. https://doi.org/10.1002/net.3230110205

Gillett, B., & Miller, L. (1974). A Heuristic Algorithm for the Vehicle Dispatch Problem. *Operations Research*, *22*(2), 340–349.

https://doi.org/10.1287/opre.22.2.340

Glover, F. (1989). Tabu Search Part I. *INFORMS Journal on Computing*, *1*(5), 190–206. https://doi.org/10.1287/ijoc.1.3.190

Glover, F. (1990). Tabu search, Part II. *ORSA Journal on Computing*, *2*(1), 4–32. https://doi.org/10.1287/ijoc.2.1.4

Goldberg, D. E. (1989). *Genetic algorithms in search optimization and machine learning* (Vol. 412). Addison-wesley Reading Menlo Park.

Golden, B. L., & Assad, A. A. (1989). Vehicle routing: Methods and studies. *European Journal of Operational Research*, *38*(1), 126–127. https://doi.org/10.1016/0377-2217(89)90483-9

Hernández, F., & Poveda, J. (2014). *Aplicación de la metaheurística cromática al problema de secueciación de proyectos con recursos limitados (RCPSP).* Universidad de Córdoba, Montería, Colombia.

Holland, J. H. (1975). *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence*. Oxford, England: U Michigan Press.

Homberger, J., & Gehring, H. (1999). Extended Solomon's VRPTW instances. *Available Electronically at: Http://www.fernuni-hagen.de/WINF/touren/menuefrm/probinst.htm*.

Hsu, C.-I., Hung, S.-F., & Li, H.-C. (2007). Vehicle routing problem with time-windows for perishable food delivery. *Journal of Food Engineering*, *80*(2), 465–475.

Kallehauge, B. (2008). Formulations and exact algorithms for the vehicle routing problem with time windows. *Computers and Operations Research*, *35*(7), 2307–

2330. https://doi.org/10.1016/j.cor.2006.11.006

Kennedy, J., Kennedy, J. F., Eberhart, R. C., & Shi, Y. (2001). *Swarm intelligence*. Morgan Kaufmann.

Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by Simulated Annealing. *Science*, *220*(4598), 671–680. https://doi.org/10.1126/science.220.4598.671

Kuo, R. J., & Zulvia, F. E. (2015). The gradient evolution algorithm: A new metaheuristic. *Information Sciences*, *316*(April), 246–265. https://doi.org/10.1016/j.ins.2015.04.031

Li, J. Q., Mirchandani, P. B., & Borenstein, D. (2009). Real-time vehicle rerouting problems with time windows. *European Journal of Operational Research*, *194*(3), 711–727. https://doi.org/10.1016/j.ejor.2007.12.037

Marinakis, Y., & Marinaki, M. (2010). A hybrid genetic–Particle Swarm Optimization Algorithm for the vehicle routing problem. *Expert Systems with Applications*, *37*(2), 1446–1455. https://doi.org/10.1016/j.eswa.2009.06.085

Montgomery, D. C. (2004). Diseño Y Análisis De Experimentos. *Limusa Wiley*.

Naciones-Unidas. (2015). Objetivos de desarrollo sostenible. 17 objetivos para transformar nuestro mundo. Objetivo 12. Retrieved September 1, 2017, from http://www.un.org/sustainabledevelopment/es/sustainable-consumptionproduction/

Ngueveu, S. U., Prins, C., & Wolfler Calvo, R. (2010). An effective memetic algorithm for the cumulative capacitated vehicle routing problem. *Metaheuristics for Logistics and Vehicle Routing*, *37*(11), 1877–1885. https://doi.org/http://dx.doi.org/10.1016/j.cor.2009.06.014

Nieto, J. M. G. (2006). Algoritmos Basados en Cúmulos de Partículas Para la Resolución de Problemas Complejos, 113. Retrieved from http://neo.lcc.uma.es/staff/jmgn/doc/Memoria_PFC_JMGN.pdf

Osvald, A., & Stirn, L. Z. (2008). A vehicle routing algorithm for the distribution of fresh vegetables and similar perishable food. *Journal of Food Engineering*, *85*(2), 285–295. https://doi.org/http://dx.doi.org/10.1016/j.jfoodeng.2007.07.008

Padberg, M., & Rinaldi, G. (1991). A Branch-and-Cut Algorithm for the Resolution of Large-Scale Symmetric Traveling Salesman Problems. *SIAM Review*, *33*(1), 60–100. https://doi.org/10.1137/1033004

Prindezis, N., Kiranoudis, C. T., & Marinos-Kouris, D. (2003). A business-to-business fleet management service provider for central food market enterprises. *Journal of Food Engineering*, *60*(2), 203–210. https://doi.org/http://dx.doi.org/10.1016/S0260-8774(03)00041-4

Righini, G., & Salani, M. (2006). Symmetry helps: Bounded bi-directional dynamic programming for the elementary shortest path problem with resource constraints. *Discrete Optimization*, *3*(3), 255–273. https://doi.org/10.1016/j.disopt.2006.05.007

Ruiz, R., Maroto, C., & Alcaraz, J. (2005). Solving the flowshop scheduling problem with sequence dependent setup times using advanced metaheuristics. *European Journal of Operational Research*, *165*(1), 34–54.

Sabie, R., & Lopez, J. (2011). Implementation of the new algorithm chromatic metaheuristic a nonlinear regression model in time series forecasts (Vol. IX Congres). Universidad de la Frontera, Temuco, Chile.

Sabie, R., & Mestra, A. (2011). *Un nuevo método de optimización que se fundamenta a*

*través de un algoritmo de búsqueda basado en la escala cromática de las notas musicales.* Universidad de Córdoba, Montería, Colombia.

Sabie, R., & Pereira, J. L. (2011). Implementación de la nueva metaheuristica algoritmo cromático a un modelo regresión no lineal en pronosticos de series de tiempo.

Schmid, V., Doerner, K. F., Hartl, R. F., Savelsbergh, M. W. P., & Stoecher, W. (2009). A Hybrid Solution Approach for Ready-Mixed Concrete Delivery. *Transportation Science*, *43*(1), 70–85. https://doi.org/10.1287/trsc.1080.0249

Silva, M. M., Subramanian, A., Vidal, T., & Ochi, L. S. (2012). A simple and effective metaheuristic for the Minimum Latency Problem. *European Journal of Operational Research*, *221*(3), 513–520. https://doi.org/http://dx.doi.org/10.1016/j.ejor.2012.03.044

Solomon, M. M. (1987). Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, *35*(2), 254–265.

Song, B. D., & Ko, Y. D. (2016). A vehicle routing problem of both refrigerated- and general-type vehicles for perishable food products delivery. *Journal of Food Engineering*, *169*, 61–71. https://doi.org/http://dx.doi.org/10.1016/j.jfoodeng.2015.08.027

Soto, D.-A. (2012). *METAHEURÍSTICAS APLICADAS AL PROBLEMA DINÁMICO DE RUTEO DE VEHÍCULOS CON DEMANDAS Y TIEMPOS ESTOCÁSTICOS, FLOTA HETEROGÉNEA, VENTANAS DE TIEMPO FLEXIBLES Y CLIENTES PREFERENCIALES (SFMVRPMSTWTDCP)*. Universidad de Córdoba.

Tarantilis, C. D., & Kiranoudis, C. T. (2001). A meta-heuristic algorithm for the efficient distribution of perishable foods. *Journal of Food Engineering*, *50*(1), 1–9.

https://doi.org/http://dx.doi.org/10.1016/S0260-8774(00)00187-4

Toth, P. & Vigo, D. (2002). *The Vehicle Routing Problem*.

Toth, P., & Vigo, D. (2002). The vehicle routing problem, Society for industrial and applied mathematics. *SIAM Monographs on Discrete Mathematics and Applications*.

Vargas, Y. N. (2011). Estrategias para mejorar el Balance entre Exploración y Explotación en Optimización de Enjambre de Partículas, 107.

Vásquez Morales, M. A. (2007). Desarrollo de un framework para el problema de ruteo de vehículos.

Vidal, T., Crainic, T. G., Gendreau, M., & Prins, C. (2014). A unified solution framework for multi-attribute vehicle routing problems. *European Journal of Operational Research*, *234*(3), 658–673.

Wang, Y., & Yu, L. Y. (2012). Optimization model of refrigerated food transportation. In *2012 9th International Conference on Service Systems and Service Management - Proceedings of ICSSSM'12* (pp. 220–224). https://doi.org/10.1109/ICSSSM.2012.6252224

# ANEXOS

## **ANEXO 1.** DESCRIPCIÓN DE INSTANCIAS (CASO I)

A continuación, se muestra una de las instancias para 10 clientes. Las demás instancias conservan la misma estructura y están disponibles en el soporte magnético entregado en este trabajo.

- **Vectores de coordenadas:** Esta información fue tomada de las instancias de Solomon. El punto "0", representa la ubicación del depósito.

| | X | Y |
|---|---|---|
| 0 | 40 | 50 |
| 1 | 22 | 85 |
| 2 | 20 | 85 |
| 3 | 18 | 75 |
| 4 | 15 | 75 |
| 5 | 8 | 45 |
| 6 | 5 | 35 |
| 7 | 2 | 40 |
| 8 | 0 | 40 |
| 9 | 40 | 15 |
| 10 | 38 | 15 |

- **Demanda de los clientes:** Se toma de las instancias de Solomon.

| | Rnp |
|---|---|
| 1 | 10 |
| 2 | 20 |
| 3 | 20 |
| 4 | 20 |
| 5 | 20 |
| 6 | 10 |
| 7 | 20 |
| 8 | 20 |
| 9 | 40 |
| 10 | 10 |

- **Tiempo de servicio:** Tomada directamente de las instancias de Solomon.

| | Ts |
|---|---|
| 0 | 0 |
| 1 | 0,166667 |
| 2 | 0,166667 |
| 3 | 0,166667 |
| 4 | 0,166667 |
| 5 | 0,166667 |
| 6 | 0,166667 |
| 7 | 0,166667 |
| 8 | 0,166667 |
| 9 | 0,166667 |
| 10 | 0,166667 |

- **Alfa, Beta, Capacidad y tipo de vehículo:** Dado que las instancias de Solomon son para una flota homogénea y no cuentan con los parámetros Alfa y Beta, estos son generados así:

Capacidad de vehículos (Cvk): aleatoria, múltiplo de 50.

Tipo de vehículo: Número aleatorio entre 1 y 2 (1-general, 2-refrigerado)

Alfa: [aleatorio entre 0,6 y 1 (1-general); aleatorio entre 0 y 0,4 (2-refrigerado)]

Beta: [aleatorio entre 0 y 0,4 (1-general); aleatorio entre 0,6 y 1 (2-refrigerado)]

| Alfa | | Beta | | Cvk | |
|---|---|---|---|---|---|
| 0,698397 | | 0,147991 | | 150 | 1 |
| 0,166591 | | 0,91672 | | 150 | 2 |

- **Matriz de tiempos de viaje:** Se obtiene a partir de la distancia euclidiana entre cada una de las coordenadas y tomando una velocidad media de 80 km/h para todos los vehículos. Se hace uso de la fórmula de movimiento rectilíneo uniforme (X=v*t).

| Tij | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0,491967 | 0,503891 | 0,416271 | 0,441942 | 0,404853 | 0,475986 | 0,491172 | 0,515388 | 0,4375 | 0,438214 |
| 1 | 0,491967 | 0 | 0,025 | 0,134629 | 0,152582 | 0,529741 | 0,660137 | 0,615554 | 0,626124 | 0,903466 | 0,897566 |
| 2 | 0,503891 | 0,025 | 0 | 0,127475 | 0,139754 | 0,522015 | 0,652519 | 0,605831 | 0,615554 | 0,910014 | 0,903466 |
| 3 | 0,416271 | 0,134629 | 0,127475 | 0 | 0,0375 | 0,395285 | 0,525744 | 0,481047 | 0,491967 | 0,798827 | 0,790569 |
| 4 | 0,441942 | 0,152582 | 0,139754 | 0,0375 | 0 | 0,385073 | 0,515388 | 0,466704 | 0,475986 | 0,8125 | 0,803216 |
| 5 | 0,404853 | 0,529741 | 0,522015 | 0,395285 | 0,385073 | 0 | 0,130504 | 0,097628 | 0,117925 | 0,548293 | 0,53033 |
| 6 | 0,475986 | 0,660137 | 0,652519 | 0,525744 | 0,515388 | 0,130504 | 0 | 0,072887 | 0,088388 | 0,503891 | 0,482345 |
| 7 | 0,491172 | 0,615554 | 0,605831 | 0,481047 | 0,466704 | 0,097628 | 0,072887 | 0 | 0,025 | 0,568578 | 0,547865 |
| 8 | 0,515388 | 0,626124 | 0,615554 | 0,491967 | 0,475986 | 0,117925 | 0,088388 | 0,025 | 0 | 0,589624 | 0,568578 |
| 9 | 0,4375 | 0,903466 | 0,910014 | 0,798827 | 0,8125 | 0,548293 | 0,503891 | 0,568578 | 0,589624 | 0 | 0,025 |
| 10 | 0,438214 | 0,897566 | 0,903466 | 0,790569 | 0,803216 | 0,53033 | 0,482345 | 0,547865 | 0,568578 | 0,025 | 0 |

**ANEXO 2.** DETERMINACIÓN DEL TAMAÑO DE LA MUESTRA (CASO I)

Se debe calcular el número de veces que se debe repetir el experimento, o número de réplicas, donde una réplica implica que cada algoritmo debe resolver cada una de las instancias. Para esto, se hizo necesario implementar el método de muestreo con base en las curvas de operación característica presentado en Montgomery (2004).

El método consiste en determinar un número de réplicas que permita generar un alto grado de potencia para el diseño experimental, y así darle una mayor confiabilidad al estudio, es decir, para que el diseño sea sensible a diferencias potenciales importantes entre los tratamientos (Montgomery 2004). Se utiliza la siguiente ecuación:

$$\Phi^2 = \frac{nD^2}{2a\sigma^2}$$

Donde:
$\boldsymbol{a}$**:** Número de niveles del factor estudiado, en este caso, el número de metaheurísticas a comparar.
$\boldsymbol{D}$**:** variación de la variable de respuesta en la cual, a partir de las *n* réplicas, se considera que habrá diferencias significativas. Por tanto, se estableció que a partir del 5% del valor de la media ($0{,}05\bar{x}$), se considerarían diferencias significativas.
$\boldsymbol{\sigma^2}$**:** Representa una estimación de la varianza.

En este caso, se desea comparar los resultados de cuatro (4) metaheurísticas (Genética, Cromática, PSO y N1). Para determinar el número de réplicas de nuestro experimento, se seleccionó una de las metaheurísticas, se fijó una instancia de 100 clientes y se realizaron 30 corridas de prueba fijando un tiempo de 3 minutos por corrida, con lo cual se obtiene una estimación de $\sigma^2$para el proceso.

Metaheurística seleccionada: Cromática
Tiempo: 3 minutos/corrida
Número de corridas: 30

| Resultado de corridas de prueba | |
|---|---|
| $\bar{x}$ | 782,3968 |
| $s^2$ | 516,4419 |
| $D^2 = (0{,}05\bar{x})^2$ | 1530,361898 |

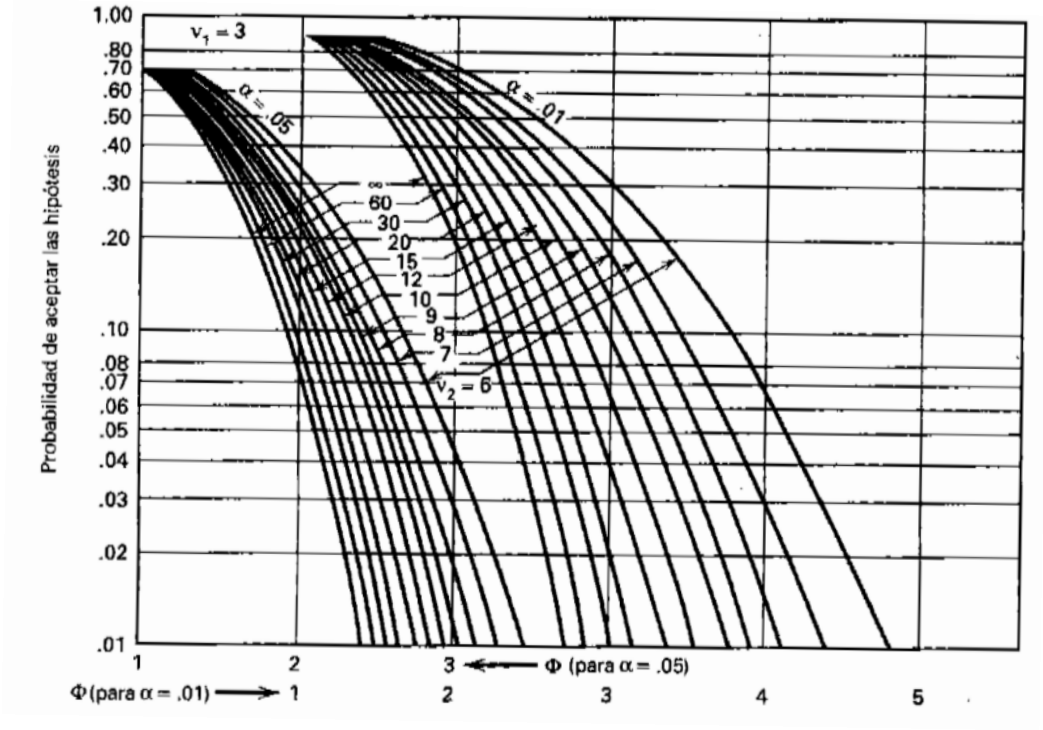
En este punto, se reemplazan los valores para encontrar la ecuación que describe el comportamiento de $\Phi^2$:

$$\Phi^2 = \frac{nD^2}{2a\sigma^2} = \frac{n(1530{,}361898)}{2(4)(516{,}4419)} = 0{,}3704n$$

Luego de esto, se realiza una tabla en la que se calculan las potencias para distintos valores de *n*, teniendo en cuenta el valor *Beta* encontrado en la curva de operación característica que irá de acuerdo a grado de libertad *v1*. Se encuentra una *n* adecuada cuando la potencia supere el 90%. A continuación, se muestran los cálculos realizados y se resalta la fila de los cálculos para *n=12*, debido a que la potencia calculada supera el 90%.

| ***n*** | $\Phi^2$ | $\Phi$ | **v1=a-1** | **v2=a(n-1)** | **Beta** | **Potencia** |
|---|---|---|---|---|---|---|
| 1 | 0,37041 | 0,608613 | 3 | 0 | - | - |
| 2 | 0,74082 | 0,860709 | 3 | 4 | - | - |
| 3 | 1,11123 | 1,054149 | 3 | 8 | 0,7 | 0,3 |
| 4 | 1,48164 | 1,217226 | 3 | 12 | 0,72 | 0,28 |
| 5 | 1,85205 | 1,3609 | 3 | 16 | 0,58 | 0,42 |
| 6 | 2,22246 | 1,490792 | 3 | 20 | 0,47 | 0,53 |
| 7 | 2,59287 | 1,610239 | 3 | 24 | 0,35 | 0,65 |
| 8 | 2,96328 | 1,721418 | 3 | 28 | 0,26 | 0,74 |
| 9 | 3,33369 | 1,82584 | 3 | 32 | 0,23 | 0,77 |
| 10 | 3,7041 | 1,924604 | 3 | 36 | 0,17 | 0,83 |
| 11 | 4,07451 | 2,018542 | 3 | 40 | 0,13 | 0,87 |
| **12** | 4,44492 | 2,108298 | 3 | 44 | 0,085 | 0,915 |

La siguiente es la gráfica con las curvas de operación característica para $v_1 = 3$, utilizada para calcular el valor de *Beta* a partir de $\Phi$ y de $v_2$ para $\alpha = 0.05$.



**ANEXO 3.** PARAMETRIZACIÓN DEL ALGORITMO N2

El algoritmo N2 posee los siguientes parámetros:

| | |
|---|---|
| $N$ | Número de partículas del enjambre |
| $Ngen_n$ | Número de soluciones para la generación de partículas |
| $t$ | Número de periodos tabú |
| $nran$ | Porcentaje de vecinos a visitar |
| ARRAM | Número de iteraciones sin mejora para un arranque múltiple |

Por lo regular los diseños de experimentos utilizados para estos fines, conllevan un extenso tiempo computacional debido a la cantidad de ejecuciones que deben ser estudiadas. Por esta razón, en esta investigación se realiza una optimización de parámetros bastante sencilla, en la cual se fija el valor de 100 para el parámetro *ARRAM*, y se establecen distintos valores para cada uno de los parámetros restantes. Los valores predefinidos se muestran en la siguiente tabla:

| $t$ | $N$ | $Ngen_n$ | $nran$ |
|---|---|---|---|
| 10 | 10 | 1500 | 0,01 |
| 30 | 30 | 2000 | 0,05 |
| | 50 | 2500 | 0,07 |

El objetivo de este experimento, consistió en determinar la combinación de parámetros que proporcionara el mejor resultado para una instancia fija de 50 clientes en un tiempo fijo de 60 minutos por corrida. Así, se realizaron en total 54 corridas, que corresponde a las 54 combinaciones resultantes de los valores predefinidos para los parámetros del algoritmo.

Como resultado de este proceso, se obtuvieron los siguientes valores:

| $t$ | $N$ | $Ngen_n$ | $nran$ |
|---|---|---|---|
| 30 | 10 | 1500 | 0,07 |

Debido a que sólo se realizó una réplica del experimento y se utilizaron valores predefinidos por el autor, los valores obtenidos para los parámetros no cuentan con un alto grado de confiabilidad, sin embargo, se logró identificar hacia donde tiende cada uno de ellos y se definió un conjunto de valores susceptible a ser utilizado para obtener un buen rendimiento del algoritmo N2.

**ANEXO 4.** DESCRIPCIÓN DE INSTANCIAS (CASO II)

A continuación, se muestra un ejemplo de una instancia de 5 clientes y 4 productos. Las instancias conservan la misma estructura y están disponibles en el soporte magnético entregado en este trabajo.

- **Vectores de coordenadas:** Esta información fue tomada de las instancias de Solomon. El punto "0", representa la ubicación del depósito.

| X | | Y | |
|---|---|---|---|
| 0 | 40 | 0 | 50 |
| 1 | 25 | 1 | 85 |
| 2 | 22 | 2 | 75 |
| 3 | 22 | 3 | 85 |
| 4 | 20 | 4 | 80 |
| 5 | 20 | 5 | 85 |

- **Tiempo de servicio:** Tomados directamente de las instancias de Solomon.

| | si |
|---|---|
| 0 | 0 |
| 1 | 0,166667 |
| 2 | 0,166667 |
| 3 | 0,166667 |
| 4 | 0,166667 |
| 5 | 0,166667 |

- **Demanda de los clientes:** Se toma la demanda total de las instancias de Solomon, y se distribuye aleatoriamente entre requerimientos de los *p* productos. En otras palabras, la suma de los requerimientos de los *p* productos para el cliente *i* es igual la demanda del cliente *i* que presenta la instancia de Solomon.

| djp | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 1 | 6 | 3 | 11 | 0 |
| 2 | 13 | 2 | 8 | 7 |
| 3 | 1 | 3 | 4 | 2 |
| 4 | 7 | 20 | 6 | 7 |
| 5 | 4 | 2 | 0 | 14 |

- **Volumen de producto:** Se tiene un volumen por producto. Este valor resulta de la generación de *p* números aleatorios entre 0 y 1.
-

| Vp | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| | 0,911138 | 0,608354 | 0,024995 | 0,128217 |

- **Theta, Psi, Capacidad y tipo de vehículo:** Dado que las instancias de Solomon son para una flota homogénea y no cuentan con los parámetros Theta y Psi, estos son generados así:

-Capacidad de vehículos (Cvk): Entre 150 y 300, múltiplo de 50.
-Tipo de vehículo: Número aleatorio entre 1 y 2 (1-general, 2-refrigerado)

| Cvk | Cap | tipo |
|---|---|---|
| | 150 | 2 |
| | 100 | 2 |
| | 250 | 2 |
| | 150 | 1 |

-Theta: [aleatorio entre 0,6 y 1 (1-general); aleatorio entre 0 y 0,4 (2-refrigerado)]

| Theta | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| | 0,13783 | 0,277148 | 0,159036 | 0,221523 |
| | 0,13783 | 0,277148 | 0,159036 | 0,221523 |
| | 0,13783 | 0,277148 | 0,159036 | 0,221523 |
| | 0,655476 | 0,699672 | 0,764661 | 0,926172 |

-Psi: [aleatorio entre 0 y 0,4 (1-general); aleatorio entre 0,6 y 1 (2-refrigerado)]

| Psi | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| | 0,938368 | 0,714246 | 0,799104 | 0,95344 |
| | 0,938368 | 0,714246 | 0,799104 | 0,95344 |
| | 0,938368 | 0,714246 | 0,799104 | 0,95344 |
| | 0,162298 | 0,314677 | 0,286167 | 0,213158 |

- **Matriz de tiempos de viaje:** Se obtiene a partir de la distancia euclidiana entre cada una de las coordenadas y tomando una velocidad media de 80 km/h para todos los vehículos. Se hace uso de la fórmula de movimiento rectilíneo uniforme (X=v*t).

| Tij | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| 0 | 0 | 0,346636 | 0,257694 | 0,363361 | 0,320156 | 0,189159 |
| 1 | 0,346636 | 0 | 0,107529 | 0,145774 | 0,15052 | 0,176777 |
| 2 | 0,257694 | 0,107529 | 0 | 0,212867 | 0,194052 | 0,072887 |
| 3 | 0,363361 | 0,145774 | 0,212867 | 0 | 0,045069 | 0,254951 |
| 4 | 0,320156 | 0,15052 | 0,194052 | 0,045069 | 0 | 0,225347 |
| 5 | 0,189159 | 0,176777 | 0,072887 | 0,254951 | 0,225347 | 0 |

## **ANEXO 5.** DETERMINACIÓN DEL TAMAÑO DE LA MUESTRA (CASO II)

Se debe calcular el número de veces que se debe repetir el experimento, o número de réplicas, donde una réplica implica que cada algoritmo debe resolver cada una de las instancias. Para esto, se hizo necesario implementar el método de muestreo con base en las curvas de operación característica presentado en Montgomery (2004).

El método consiste en determinar un número de réplicas que permita generar un alto grado de potencia para el diseño experimental, y así darle una mayor confiabilidad al estudio, es decir, para que el diseño sea sensible a diferencias potenciales importantes entre los tratamientos (Montgomery 2004). Se utiliza la siguiente ecuación:

$$\Phi^2 = \frac{nD^2}{2a\sigma^2}$$

Donde:
$\boldsymbol{a}$**:** Número de niveles del factor estudiado, en este caso, el número de metaheurísticas a comparar.
$\boldsymbol{D}$**:** variación de la variable de respuesta en la cual, a partir de las *n* réplicas, se considera que habrá diferencias significativas. Por tanto, se estableció que a partir del 5% del valor de la media ($0{,}05\bar{x}$), se considerarían diferencias significativas.
$\boldsymbol{\sigma^2}$**:** Representa una estimación de la varianza.

En este caso, se desea comparar los resultados de tres (3) metaheurísticas (Genética, Cromática y N2). Para determinar el número de réplicas de nuestro experimento, se seleccionó una de las metaheurísticas, se fijó una instancia de 50 clientes y se realizaron 30 corridas de prueba fijando un tiempo de 30 minutos por corrida, con lo cual se obtiene una estimación de $\sigma^2$para el proceso.

Metaheurística seleccionada: Cromática
Tiempo: 30 minutos/corrida
Número de corridas: 30

| Resultado de corridas de prueba | |
|---|---|
| $\bar{x}$ | 21068,396 |
| $s^2$ | 377180,3237 |
| $D^2 = (0{,}05\bar{x})^2$ | 1109693,275 |

En este punto, se reemplazan los valores para encontrar la ecuación que describe el comportamiento de $\Phi^2$:

$$\Phi^2 = \frac{nD^2}{2a\sigma^2} = \frac{n(1109693{,}275)}{2(3)(377180{,}3237)} = 0{,}49n$$

Luego de esto, se realiza una tabla en la que se calculan las potencias para distintos valores de *n*, teniendo en cuenta el valor *Beta* encontrado en la curva de operación característica que irá de acuerdo a grado de libertad *v1*. Se encuentra una *n* adecuada cuando la potencia supere el 90%. A continuación, se muestran los cálculos realizados y se resalta la fila de los cálculos para *n=10*, debido a que la potencia calculada supera el 90%.

| ***n*** | $\Phi^2$ | $\Phi$ | **v1=a-1** | **v2=a(n-1)** | **Beta** | **Potencia** |
|---|---|---|---|---|---|---|
| 1 | 0,49034604 | 0,70024713 | 2 | 0 | - | - |
| 2 | 0,98069209 | 0,99029899 | 2 | 3 | - | - |
| 3 | 1,47103813 | 1,21286361 | 2 | 6 | 0,7 | 0,3 |
| 4 | 1,96138417 | 1,40049426 | 2 | 9 | 0,58 | 0,42 |
| 5 | 2,45173021 | 1,56580018 | 2 | 12 | 0,47 | 0,53 |
| 6 | 2,94207626 | 1,71524816 | 2 | 15 | 0,33 | 0,67 |
| 7 | 3,4324223 | 1,85267976 | 2 | 18 | 0,25 | 0,75 |
| 8 | 3,92276834 | 1,98059798 | 2 | 21 | 0,17 | 0,83 |
| 9 | 4,41311439 | 2,10074139 | 2 | 24 | 0,14 | 0,86 |
| **10** | 4,90346043 | 2,21437586 | 2 | 27 | 0,09 | 0,91 |

La siguiente es la gráfica con las curvas de operación característica para $v_1 = 2$, utilizada para calcular el valor de *Beta* a partir de $\Phi$ y de $v_2$ para $\alpha = 0.05$.