UNA NUEVA METAHEURÍSTICA APLICADA AL PROBLEMA DE RUTEO DE VEHÍCULOS CAPACITADOS (CVRP) PARA LA DISTRIBUCIÓN DE PRODUCTOS PERECEDEROS

A NEW METAHEURISTIC APPLIED TO THE CAPACITATED VEHICLE ROUTING PROBLEM (CVRP) FOR THE DISTRIBUTION OF PERISHABLE PRODUCTS

Jesús D. Galarcio[1*], Maria P. Buelvas[2], Paula A. Nisperuza[3] , Jorge M. López[4] , Helman E. Hernández[5]

Recibido para publicación: 23 de febrero 2017 - Aceptado para publicación: 2 de mayo 2017

RESUMEN

La distribución representa un gran desafío para las empresas de alimentos perecederos que buscan satisfacer a sus clientes en términos de una calidad cambiante a lo largo de la cadena de suministro. En este contexto, en el siguiente articulo presentamos un nuevo modelo matemático para la distribución de productos perecederos que tiene como objetivo minimizar el tiempo total de viaje y las tardanzas en la entrega de los productos transportados. Con el objetivo de enfrentar este problema complejo, empleamos un algoritmo genético (GA) y una nueva metaheurística hibrida (N2) que combina operadores de tres distintos algoritmos para resolver cinco instancias de 25 clientes adaptadas de la literatura. Los resultados permiten comprobar la funcionalidad del modelo y muestran que el nuevo algoritmo es capaz de obtener mejores resultados en comparación con el algoritmo evolutivo, lo que brinda a los tomadores de decisiones una buena herramienta para la planeación de rutas de este tipo de productos.

Palabras clave: Ruteo de vehículos, productos perecederos, metaheurísticas, algoritmo genético, algoritmo cromático.

ABSTRACT

Distribution constitutes a major challenge for perishable food companies who seek to satisfy their customers in terms of quality, which changes through the supply chain. In this context, in the following article we present a new mathematical model for the distribution of perishable products that aims to minimize the total travel time and tardiness in the delivery of transported products. In order to address this complex problem, we employ a genetic algorithm (GA) and a new hybrid metaheuristic (N2) that combines operators of three different algorithms to solve five 25-customer instances adapted from the literature. The results prove that the model is capable of providing feasible solutions for the problem and show that the new algorithm is capable of obtaining better results compared to the evolutionary algorithm, which gives decision makers a good tool for route planning for this type of product.

Key words: Vehicle routing problem, Perishable products, metaheuristics, genetic algorithm, chromatic algorithm

[1*] Ingeniero Industrial, Universidad de Córdoba, Montería, Colombia – email: jgalarcionoguera04@correo.unicordoba.edu.co - Dirección: Cra 20 No 8a-40 Santa Teresa, Cereté. Teléfono: 3008668042
[2] Ingeniero Industrial, Universidad de Córdoba, Montería, Colombia – email: mayibupa@gmail.com
[3] Ingeniero Industrial, Universidad de Córdoba, Montería, Colombia – email: paanica2795@hotmail.com
[4] MsC en Ingeniería Industrial, Universidad de Córdoba, Montería, Colombia – email: jotamlopez@correo.unicordoba.edu.co
[5] PhD en Ingeniería Industrial, Universidad de Córdoba, Montería, Colombia – hhernandez@correo.unicordoba.edu.co

1. INTRODUCCIÓN

Aquellos productos que deben ser usados o consumidos en cortos periodos de tiempo son conocidos como productos perecederos (derivados de animales y vegetales). Dentro de este grupo se encuentran productos tipo alimenticio como las frutas y hortalizas consideradas como productos altamente perecederos, ya que luego de cosechadas se estropean naturalmente (Ahmad y Siddiqui 2015). Sin embargo, existen procesos como almacenamiento y transporte, que aceleran el deterioro de estos productos, generando pérdidas (Kasso y Bekele 2016). A menudo este deterioro tiene lugar debido a largos tiempos de viaje y las frecuentes paradas que se tiene un vehículo al servir a un determinado cliente, durante el proceso de entrega (Hsu el al. 2007). Es decir que cualquier proceso relacionado con este tipo de productos resulta exitoso o no de acuerdo a la gestión de los factores que afectan la calidad de los mismos.

El problema de ruteo de vehículos (VRP) es una versión extendida de uno de los problemas de ruteo más ampliamente estudiados como lo es el problema del agente viajero (TSP), el objetivo que se persigue en este problema es minimizar a distancia total recorrida, cuando uno o varios agentes deben visitar un numero de nodos y regresar al punto inicial de partida. El VRP es un problema TSP, pero en este los agentes son reemplazados por vehículos, se incluyen las demandas de cada nodo y se especifica la capacidad de carga de cada vehículo individual (Mohapatra 2014). Los problemas de este tipo son estudiados de acuerdo a su complejidad, los recursos comúnmente estudiados en complejidad computacional son: el tiempo (aproximación al número de pasos de ejecución que un algoritmo emplea para resolver un problema), y el espacio (aproximación a la cantidad de memoria utilizada para resolver el problema). De acuerdo a su complejidad los problemas se clasifican en L (lineal), NL (no lineal), P (polinomial), P-Complete (polinomial completo), NP (no polinomial), NP-Complete (no polinomial completo), NP-Hard (no polinomial duro). Debido a la dimensión y complejidad que presenta el VRP, este es considerado un tipo de problema NP-Hard, esto es, no existe un algoritmo que en tiempo polinomial pueda resolver cualquier instancia del problema.

El VRP simétrico es definido como un grafo no dirigido $G=(V,E)$. El conjunto de vértices es $V=\{0,\ldots,n\}$. Cada vértice $i \in V \setminus \{0\}$, representa un cliente que tiene una demanda no negativa q_i, mientras el vértice 0 corresponde al depósito. Para cada eje $e \in E = \{(i,j):i,j \in V, i<j\}$ hay asociado un costo de viaje C_e o C_{ij}. Una flota fija de m vehículos idénticos, cada uno con capacidad Q, está disponible en el depósito. El VRP simétrico requiere la determinación de un conjunto de m rutas, cuyo costo de viaje total es minimizado y se cumpla que: cada cliente es visitado exactamente una vez por una ruta, cada ruta inicia y termina en el depósito, la demanda total de los clientes servidos en una ruta no excede la capacidad del vehículo Q y que el tamaño de cada ruta no debe exceder un límite pre-establecido L. (Comúnmente se asume una velocidad constante de modo que las distancias, los tiempos de viaje y los costos de viaje son sean considerados como sinónimos). Una solución puede ser vista como un conjunto de m ciclos que comparten un vértice común, el depósito. Por su parte el VRP asimétrico es similarmente definido como un grafo direccionado $G = (V,A)$, donde $A = \{(i,j):i,j \in V, i \neq j\}$ es un conjunto de arcos. En este caso un circuito (ciclo dirigido) está asociado a una ruta y un vehículo (Cordeau et al. 2007).

A lo largo de los años se han desarrollado diversas variantes o extensiones para el problema de VRP, estas incluyen en el problema diversidad de variables, como capacidad de vehículos, tipo de flota de vehículos, número de depósitos, tiempos de entrega y recogidas, entre otras.

La versión básica del VRP es conocida como CVRP, en esta versión la flota de vehículos es homogénea, existe un único depósito, las demandas son conocidas y los clientes corresponden a entregas. Este problema es representado como un grafo $G = (V,A)$ donde $V = \{0,1,\ldots,n\}$ es el conjunto de vértices y A es el conjunto de arcos. Los vértices $i = 1,\ldots,n$ corresponden a los clientes, donde el vértice 0 corresponde al depósito. Algunas veces el depósito es asociado con el vértice n + 1. Asociado a cada arco $(i,j) \in A$, tiene un costo C_{ij} no negativo, el cual representa el costo del viaje gastado en ir del vértice i al vértice j. Cada cliente está asociado a una demanda conocida d_i y el depósito tiene asociada una demanda ficticia $d_0 = 0$, un conjunto de vehículos idénticos K, cada uno con capacidad C, para garantizar la factibilidad se asume que $d_i \leq C$ para cada cliente, de tal manera que la demanda total de cada ruta no puede

exceder la capacidad del vehículo asignado. Dado un conjunto $S \subseteq V \setminus \{0\}$, $r(S)$ es el número mínimo de vehículos necesarios para servir a todos los clientes. La variable decisión binaria x_{ijk} es igual a 1 el vértice j es visitado después del vértice i por el vehículo k, de otro modo es 0. De acuerdo a la notación dada, la formulación matemática del modelo clásico del CVRP es:

$$Min \sum_{i \in V} \sum_{j \in V} C_{ij} X_{ij}$$

Sujeto a:

$$\sum_{i \in V} x_{ij} = 1 , \quad \forall j \in V \backslash \{0\},$$
$$\sum_{j \in V} x_{ij} = 1 , \quad \forall i \in V \backslash \{0\},$$
$$\sum_{i \in V} x_{i0} = K ,$$
$$\sum_{j \in V} x_{0j} = K ,$$
$$\sum_{i \notin S} \sum_{j \in S} X_{ij} \geq r(S) \qquad \forall S \subseteq V \setminus \{0\}, S \neq \emptyset$$
$$x_{ij} \in (0,1) \qquad \forall i, j \in V$$

Las dos restricciones iníciales son de entrada y de salida, indican que exactamente un arco entra y sale de cada vértice asociado a un cliente, respectivamente; las dos siguientes establecen los requerimientos del depósito; la restricción desigualdad es la restricción de capacidad de los vehículos y elimina los subtours (recorrido en el cual el vehículo parte y regresa al depósito, pero no visita todos los vértices).

El problema de ruteo de vehículos se encuentra en el corazón de la gestión de la distribución (Cordeau et al. 2007). Este es enfrentado cada día por miles de compañías encargadas de recoger y entregar bienes y servicios o personas. De esta manera el VRP tiene una relación estrecha con la distribución de los llamados productos perecederos, como frutas y vegetales. Como se mencionó anteriormente, parte del daño que se generan en este tipo de productos es debido a los largos tiempos de viajes al transportar los productos, a la vez estos largos tiempos de viaje se presentan como consecuencia de la no adecuada planeación de las rutas de entrega y distribución, lo que ha llevado a muchas organizaciones al uso de métodos eficientes para la creación de rutas de entrega más convenientes que minimicen el tiempo total gastado en transportar los productos desde un depósito hasta los clientes. Son muchas empresas que reportaron éxito al implementar algoritmos matemáticos para optimizar sus operaciones de camionaje y transporte (Fisher 1995).

En los últimos años el problema de la distribución de productos perecederos ha sido abordado por diversos autores como un VRP. Algunos autores se enfocan en la distribución de productos perecederos sin tener en cuenta la pérdida de calidad y frescura del producto tal como lo hacen (Amorim y Almada-Lobo 2014); Faulin (2003) aplicó un procedimiento en un problema de enrutamiento para optimizar la entrega de alimentos y Tarantilis y Kiranoudis, (2001, 2002) se centraron en la distribución de la leche y de la carne. Teniendo en cuenta la complejidad del problema tratado, su estudio y resolución ha originado el desarrollo e implementación de técnicas heurísticas y metaheurísticas de aplicación general, con el fin de encontrar soluciones de calidad en un tiempo computacional razonable (Dyer y Stougie 2006). Específicamente para el problema de ruteo de vehículos en el área de productos perecederos se destacan diversas investigaciones, las cuales se han enfocado en la planeación de la cadena de suministro. Tarantilis y Kiranoudis (2001) desarrollaron un algoritmo meta-heurístico para la distribución de alimentos perecederos; Prindezis et al. (2003) desarrolló un proveedor de servicios de distribución logística basado en metaheurísticas para centros que distribuyen productos alimenticios frescos; tomando la frescura como factor crítico e incluyendo restricciones de ventanas de tiempo y tiempos de viaje Osvald y Stirn (2008) construyen un algoritmo para modelar la distribución de vegetales frescos; Keizer et al. (2015) demuestra mediante el diseño de una red logística para productos perecederos, como flores y otros productos agrícolas que la pérdida de calidad debe ser tenida en cuenta en la optimización para entregar satisfactorios al cliente.

En el presente estudio se hacen dos principales contribuciones. En primer lugar, se plantea un modelo basado en el CVRP, que tiene como objetivo minimizar el tiempo en que los productos son transportados, considerando, a diferencia de los artículos revisados, las tardanzas en las rutas. En segundo lugar, dada la complejidad del problema se propone un nuevo algoritmo para solucionar distintas instancias, este algoritmo es luego comparado con una metaheurística evolutiva conocida. Así se crea una herramienta

de apoyo para la planificación de rutas de entrega de productos perecederos haciendo que estos lleguen al cliente final con la mayor calidad posible, considerando al tiempo como principal factor.

El siguiente articulo esta está dividido de la siguiente forma. En la sección 2 se describe detalladamente la metodología empleada para la realización del estudio. En la sección 3 se muestran y posteriormente analizan los resultados experimentales. Finalmente, en la sección 4 se realizan las conclusiones del estudio y plantean futuras direcciones de investigación a partir del modelo.

2. METODOLOGÍA

En esta sección se realiza una definición detallada del problema y se explica la formulación matemática utilizada para el planteamiento del modelo.

Seguidamente, se describen las metaheurísticas empleadas y se muestran los parámetros a utilizar en el diseño experimental a partir de los dos algoritmos y las cinco instancias de Solomon (1987) para la validación del modelo propuesto.

2.1. Definición y formulación del problema

El problema puede ser representado como un grafo $G = (V,A)$ donde $V = \{0,1...,n\}$ representa el conjunto de nodos y $V' = V \setminus \{0\}$ al conjunto de clientes. Se asume que una flota homogénea de vehículos de tamaño K con capacidad para transportar una cantidad C de un producto perecedero se encuentra en el depósito disponible para suplir la demanda de los clientes. A cada vértice $i \in V'$ le es asignada una demanda no negativa D_i, y un tiempo de servicio no negativo S_i. Asociada a cada trayectoria entre dos nodos i y j ($i \neq j$) hay un tiempo de viaje t_{ij} que representa el tiempo que toma recorrer dicha trayectoria. Adicionalmente se incluye el parámetro dt que representa un tiempo luego del cual el producto no debería ser entregado.

La variable binaria de decisión x_{ijk} es igual a 1 si el conductor k visita el vértice j después del i y 0 de otro modo. De forma semejante, la variable y_{ik} toma el valor 1 si el cliente i pertenece a la ruta k. Por otro lado, la variable td_k representa la tardanza en la entrega del producto en la ruta k. Finalmente, las variables Z y L_{ik} representan respectivamente el tiempo total de viaje con penalización de tardanza en las rutas y la carga del vehículo k cuando deja el nodo i.

Partiendo de la notación anterior, se formula un modelo matemático para representar la optimización del tiempo total de viaje, el cual es penalizado por la tardanza en las rutas. En general, este modelo está enfocado en el diseño de rutas con tardanza cero, permitiendo evitar la pérdida de productos durante el transporte y, a su vez, minimizando los tiempos de viaje en cada ruta. El modelo propuesto se presenta a continuación.

$$Min(Z) = \sum_{k\in R}\sum_{i\in V}\sum_{j\in V} x_{ijk} * t_{ij} + \sum_{i\in V'} s_i + \sum_{k\in R} td_k \quad (1)$$

Sujeto a:

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

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

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

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

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

$$\sum_{i\in V}\sum_{j\in V'} x_{ijk} * (t_{ij} + s_j) - dt \leq td_k, \quad \forall k \in R \quad (7)$$

$$D_j - M(1 - x_{0jk}) \leq L_{jk}, \quad \forall j \in V', k \in R \quad (8)$$

$$L_{ik} + D_j - M(1 - x_{ijk}) \leq L_{jk}, \quad \forall (i,j) \in V', k \in R \quad (9)$$

$$L_{jk} \leq C, \forall j \in V', k \in R \quad (10)$$

$$td_k \geq 0, k \in R \quad (11)$$

$$L_{ik} \geq 0, \forall i \in V', k \in R \quad (12)$$

$$x_{ijk} \in \{0,1\}, \forall (i,j) \in V, i \neq j, k \in R \quad (13)$$

$$y_{ik} \in \{0,1\}, \forall i \in V', k \in R \quad (14)$$

En el modelo planteado la función objetivo (1) minimiza la suma de los tiempos de llegada de los diferentes vehículos y la tardanza en la entrega del producto. La ecuación (2) garantiza el flujo de la red. Las ecuaciones (3) y (4), de manera respectiva, indican que cada vehículo debe

salir y regresar del depósito y que el número de vehículos empleados no puede exceder el número de la flota. Seguidamente, las ecuaciones (5) y (6) no permiten que un cliente sea servido por más de un vehículo y que cada cliente sea asignado a una ruta. La restricción (7) calcula las tardanzas en la entrega del producto en cada una de las rutas. Las ecuaciones (8) y (9) evitan la creación de subtours.

Finalmente, las ecuaciones de la (10) a la (14) establecen los dominios de cada una de las variables de decisión.

2.2. Diseño de metaheurísticas

2.2.1. Algoritmo genético

Desde que fue propuesto por Holland (1992) el algoritmo genético ha sido ampliamente utilizado para solucionar diversas variantes del VRP (Liu and Xu 2008; Vidal et al. 2012; Sivaram Kumar et al, 2014; Li et al. 2015; Saeidian, Mesgari y Ghodousi 2016). En él 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 1, se muestra un diagrama de flujo que describe cómo opera esta metaheurística.

Los pasos utilizados se describen brevemente a continuación:

- Inicialización de 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.
- Generación de la población inicial: La población inicial de individuos es generada de acuerdo al tamaño previamente asignado. Para fines de este articulo utilizaremos para representar cada individuo de la población inicial un *enconding* de tipo permutado en el que cada número representa a un cliente y la secuencia de la permutación el orden en que los clientes son atendidos y asignados a una ruta.
- Evaluación: 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. El resultado de esta evaluación será el *fitness* de cada solución.
- 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 a su fitness. En la selección se forma la matriz de "padres".
- Cruzamiento: Si se cumple con la probabilidad de cruzamiento dos padres son escogidos para generar el mismo número de hijos. Para realizar este paso se genera un punto de corte aleatorio y se intercambian los alelos de los individuos.
- Mutación: Si se cumple con la probabilidad de mutación esta se realiza a través de la selección y el intercambio de dos elementos de un individuo de forma aleatoria.

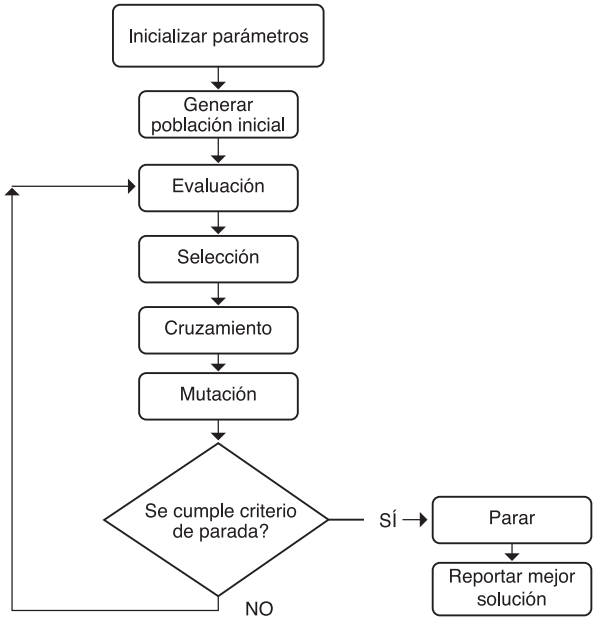

Figura 1. Flujograma del algoritmo genético

2.2.2. Diseño de la metaheurística N2

Esta nueva propuesta algorítmica encierra aspectos de distintas metaheurísticas, tales como el algoritmo cromático (AC) (Sabie y Mestra 2011), la optimización por enjambre de partículas (PSO) (Kennedy et al. 2001; Clerc 2004), y la búsqueda tabú (TS) (Glover 1989, 1990; Batista y Glover 2006).

2.2.2.1. Aportes del algoritmo cromático

Con respecto a la metaheurística cromática, esta fue diseñada inicialmente para resolver problemas de optimización en un dominio continuo de datos (Sabie y Mestra 2011). 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). 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).

Para el diseño de la metaheurística N2, se tomaron tres operadores del algoritmo cromático: (a) Utilización de la mejor melodía, (b) combinación de dos melodías seleccionadas aleatoriamente y (c) búsqueda de vecinos de rotación de notas.

a. Utilización de la mejor melodía

En la metaheurística cromática, las soluciones son llamadas “melodías” y la mejor melodía (MM) representa la mejor solución encontrada en su ejecución. Este operador consiste en tomar los valores de cada posición j de la melodía (MM) e insertarlos en una melodía i. Este operador puede entenderse mejor con el ejemplo que se muestra en la figura 2.

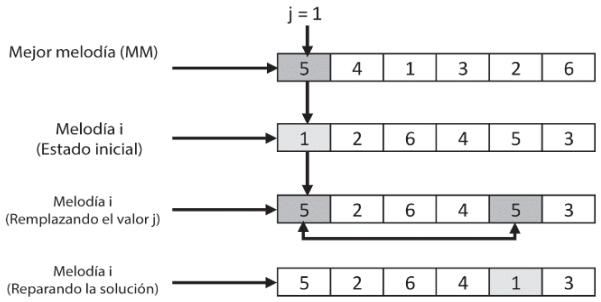

Figura 2. Utilización de la mejor melodía

a. Combinación de dos melodías seleccionadas aleatoriamente

En este operador, se escogen aleatoriamente dos melodías del conjunto de soluciones y se le asigna a la primera la letra S y el valor del uno, y a la segunda, la letra P y el valor de dos.

A continuación, se genera un vector con números aleatorios entre uno y dos y se construye una nueva melodía a partir de este vector. Este proceso, se realiza de derecha a izquierda, asignando a la nueva melodía, los valores de la posición j de las melodías S o P, según sea el caso.

Si un valor ya ha sido asignado, se procede con el siguiente valor de la misma melodía que no haya sido asignado. La figura 3 ilustra un ejemplo.

b. Búsqueda de vecinos de rotación de notas

En este operador, se hace una analogía con el proceso musical donde se tiene que los cambios melódicos más importantes y básicos para una nota, 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.

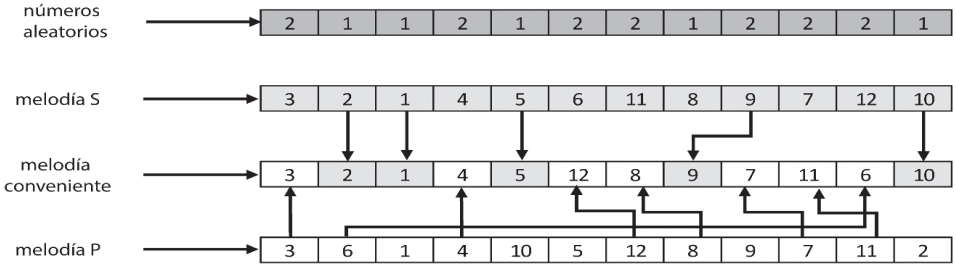

Figura 3. Ejemplo combinación de dos melodías aleatorias S y P

Los vecinos de rotación pueden ser de dos clases: descendentes y ascendentes.

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 tabla 1.

Cada nota gira hacia delante, tomando la posición clave más cercana (figura 4).

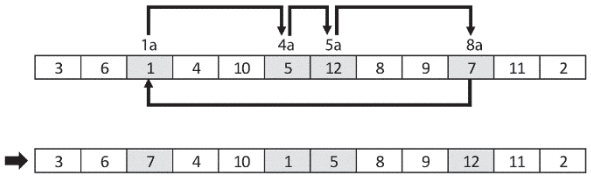

Figura 4. Vecinos de rotación descendentes

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	$8^{va}\rightarrow$ 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}	$5^{ta} \rightarrow$ se obtiene de $\rightarrow 8^{va} - 3$
3	Cuarta de la nota correspondiente	Esta encuentra al restarle 4 posiciones a la 8^{va}	$4^{ta} \rightarrow$ se obtiene de $\rightarrow 8^{va} - 4$
4	Primera de la nota correspondiente	Esta encuentra al restarle 7 posiciones a la 8^{va}	$1^{ra} \rightarrow$ se obtiene de $\rightarrow 8^{va} - 7$

II. 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 tabla 2.

Cada nota gira hacia delante, tomando la posición clave más cercana (figura 5).

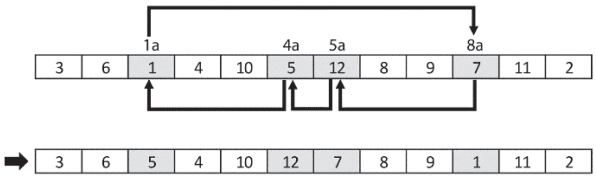

Figura 5. Vecinos de rotación ascendentes

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 $\rightarrow {}^{1ra} +7$

2.2.2.2. Aportes de la metaheurística PSO

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 y (F_g) como el valor de su evaluación.

2.2.2.3. Aportes de la Búsqueda tabú

La búsqueda tabú, o Tabú Search (Batista y Glover 2006) permitió introducir varios conceptos al algoritmo N2, como el de memoria a corto plazo. Se implementa una lista tabú para cada partícula, homologando y adaptando en capas la estructura de datos planteada en Batista y Glover (2006).

También, se adoptó de forma parcial el concepto de búsqueda tabú, teniendo en cuenta que no se generan y evaluan todos los vecinos, sino que se establece un parámetro (nran), definido como el porcentaje de vecinos a generar dentro del rango total de vecinos (ecuaciones 15 y 16).

$$\binom{Numero\ de}{vecinos\ posibles} = k\ \boldsymbol{C}\ 2 = \frac{k!}{2!(k-2)!} \qquad (15)$$

$$\binom{Número\ de\ vecinos}{a\ generar} = \left[\!\!\left[\begin{matrix} numero\ de\ vecinos \\ posibles * nran \end{matrix} \right]\!\!\right] \qquad (16)$$

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.

Para seleccionar los vecinos más susceptibles a ser visitados, se utiliza el concepto de *valor de movimiento*, ya que se calcula para cada uno de los vecinos de cada partícula X*i*, y de este valor dependerá el orden en la lista tabú. Así, se calcula el valor del movimiento (ecuación 17).

$$\binom{Valor\ del\ movimiento}{\forall m} = F(Xvecinos_{im}) - Factual(X_i), \qquad (17)$$

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).

De acuerdo al procedimiento de la búsqueda tabú, se organizan todos los vecinos de menor a mayor con respecto al *valor del movimiento* y se selecciona aquel de menor valor que *no se encuentre en la lista tabú* de la partícula X_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 potencial proporciona una mejor solución que la mejor solución encontrada (F_g).

2.2.2.4. Metaheurística N2

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. 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. 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).

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 del operador de búsqueda tabú. Si una partícula (X_i) sufre una transformación producto del uso de un operador distinto, su lista tabú, ($MInd_i$), es reiniciada.

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 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. 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.

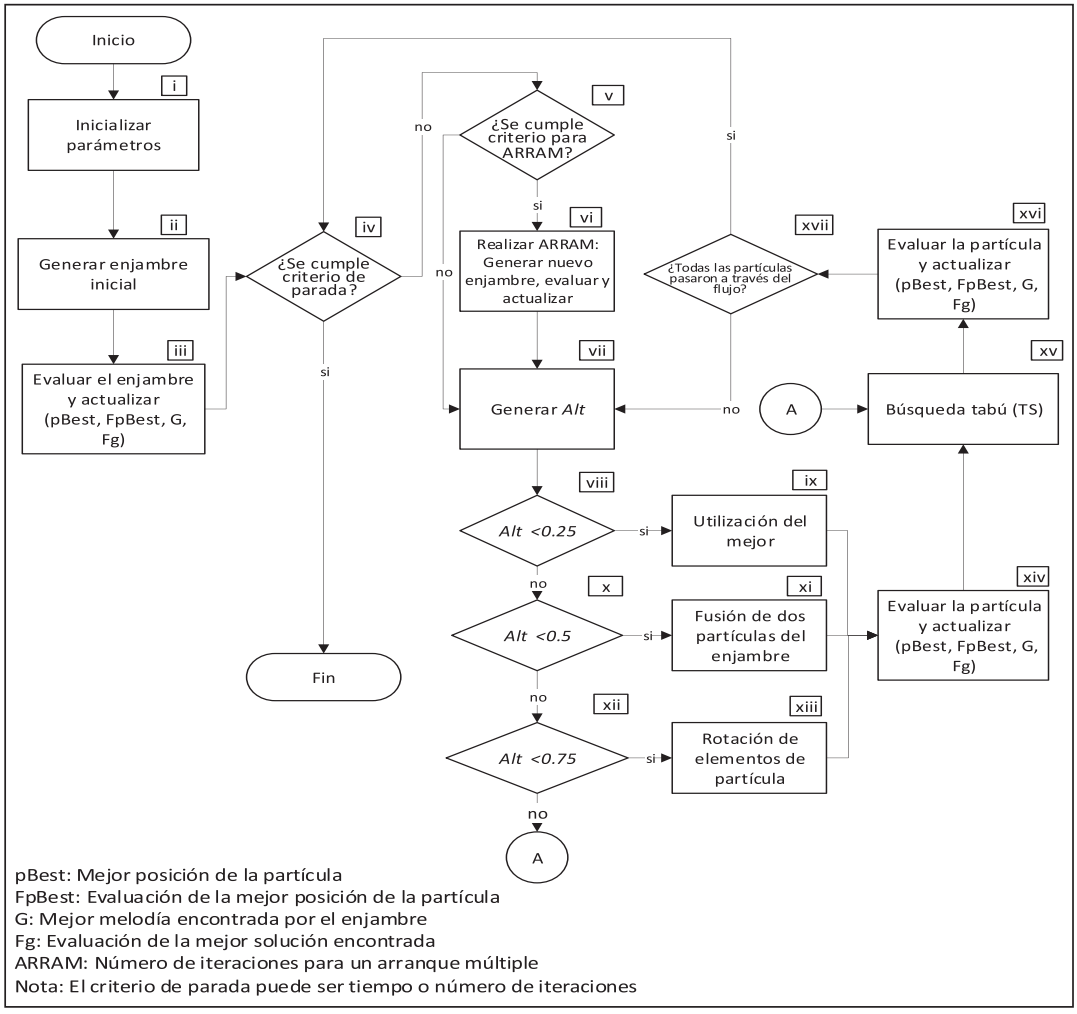

Figura 6. Diagrama de flujo de la metaheurística N2.

A continuación, se explican los pasos representados en la figura 6.

a. *Inicializar parámetros*. Se verifican los parámetros de la metaheurística (*Ngen*$_n$, *N, ARRAM, t, nran*) y los parámetros del problema.
b. *Generar enjambre inicial*. Se genera un enjambre de tamaño *N*.
c. *Evaluar el enjambre y actualizar*. Se actualiza *pBest, FpBest, G y Fg*, para las *N* partículas del enjambre.
d. ¿*Se cumple criterio de parada*? Si el criterio de parada se cumple, el algoritmo finaliza su ejecución. De lo contrario, ir al paso v.
e. ¿*Se cumple criterio para ARRAM*? Se verifica si se cumple el número de iteraciones para realizar un arranque múltiple (ARRAM). Si se cumple, ir al paso vi, de otro modo, se procede al paso *vii*.
f. *Generar nuevo enjambre, evaluar y actualizar*. Se realizan los procedimientos descritos en los pasos *ii* y *iii*, se reinicia el contador de iteraciones para un ARRAM y se avanza al paso *vii*.
g. *Generar Alt*. Se genera un número aleatorio *Alt* que definirá el camino a seguir para cada partícula en cada iteración.
h. *Alt <0.25*. Si el número *Alt* es menor que 0.25, ir al paso *ix*, de lo contrario, avanzar al paso *x*.
i. *Utilización del mejor*. Se utiliza la partícula (*G*), aplicando el operador del algoritmo cromático.
j. *Alt <0.5*. Si *Alt* es mayor o igual que 0.25 y menor que 0.5, ir al paso *xi*, de otro modo, proceder al paso *xii*.
k. *Fusión de dos partículas del enjambre*. Se rige por el principio de combinar dos melodías escogidas aleatoriamente.
l. *Alt <0.75*. Si *Alt* es mayor o igual a 0.5 y menor a 0.75, ir paso *xiii*, de otro modo, realizar paso *xv*.
m. *Rotación de elementos de partícula*. Se aplica este operador a la partícula X_i.
n. *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.
o. *Búsqueda Tabú (TS)*. Se generan los vecinos de la partícula X_i con base en el porcentaje de vecinos, (*nran*). Se calcula el *Valor del movimiento* 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.
p. *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.
q. ¿*Todas las partículas completaron el flujo*? Se comprueba que todas las partículas hayan pasado a través del flujo. Si se cumple, se avanza al paso *vi*, de lo contrario, se inicia el paso *vii* con la siguiente partícula.

2.3. Diseño de experimentos

En este apartado se proponen una serie de experimentos para probar la efectividad de las metaheurísticas anteriormente explicadas. Cada una de ellas fue codificada y ejecutada en Matlab 2015a en una computadora con 2GB de RAM y procesador Intel Celeron CPU G1610 a 2.60 GHz con Windows 7 Home Premium. Con el propósito de realizar el cálculo de réplicas se ejecutó una pre-prueba de 30 corridas, cada una por un tiempo de 6 minutos, utilizando el algoritmo genético. Así, a partir de la metodología propuesta por Montgomery (2008), se obtuvieron un total de 24 réplicas necesarias en el experimento lo que significa un total de 240 corridas – 24 (replicas) x 2 (algoritmos) x 5 (instancias) - que se ejecutaron cada una por un tiempo de 3 minutos. Para los experimentos fueron aleatoriamente seleccionadas las instancias R101, R109, RC208 y C105 creadas originalmente por Solomon (1987) para el ruteo de vehículos con ventanas de tiempo (VRPTW).

Dado que nuestro problema corresponde a una variante del CVRP, solo se tuvieron en cuenta los parámetros correspondientes a las coordenadas, demandas, capacidades, tamaño máximo de flota y tiempos de servicio de las instancias mencionadas. Los parámetros para la ejecución de cada uno de los algoritmos se muestran en la tabla 3.

Tabla 3. Parámetros para los experimentos por algoritmo

Algoritmo	Parámetro	Valor
Algoritmo Genético	Probabilidad de Mutación	0,05
	Probabilidad de Cruzamiento	0,9
	Tamaño de Población inicial	100
*Algoritmo H2**	NgenN	1500
	N	10
	ARRAM	100
	τ	30
	PRAN	0.07

**Los parámetros para el algoritmo N2 son el resultado de la experimentación de los autores.*

3. RESULTADOS Y DISCUSIÓN

Se realizó un experimento para 5 instancias de 25 clientes con 2 algoritmos y 24 réplicas. Los resultados obtenidos se resumen en la tabla 4.

Los mejores resultados para cada instancia aparecen con un asterisco, mientras que los mejores promedios están subrayados. Puede observarse que algoritmo N2, en general, encontró los mejores resultados para cada instancia y obtuvo los mejores promedios.

Con el objetivo de efectuar comparaciones del desempeño de cada algoritmo sin importar la instancia en la cual fue ejecutado, se estandarizaron los valores de la función objetivo mediante la ecuación 18.

$$Z_i = \frac{x_{ij} - \bar{x}_i}{s_i} \qquad (18)$$

Donde x_{ij} es 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.

Tabla 4. Resumen de resultados por instancia y por algoritmo

INSTANCIA i	ALGORITMO						Min(prom) para la instancia i	Min. de F para la instancia i	$\overline{x_i}$	s_i
	ALGORITMO GENÉTICO			ALGORITMO N2						
	Prom. de F	Mín. de F	Desvest de F	Prom. de F	Mín. de F	Desvest de F				
1	76,93	76,21	0,44	75,65	75,54*	0,10	75,65	75,54	76,29	0,72
2	75,69	74,81	0,56	74,62	74,54*	0,07	74,62	74,54	75,16	0,67
3	10,21	9,03	0,48	8,69	8,46*	0,18	8,69	8,46	9,45	0,85
4	10,28	9,35	0,45	8,74	8,46*	0,25	8,74	8,46	9,51	0,86
5	8,10	7,07	0,46	7,00	6,99*	0,02	7,00	6,99	7,55	0,64

Ahora bien, para indagar sobre la existencia de diferencias significativas entre los resultados obtenidos por cada algoritmo, se evaluaron los supuestos correspondientes para utilizar un análisis de varianza, sin embargo, los datos de las observaciones no cumplieron con los supuestos de normalidad. Teniendo en cuenta lo anterior, se optó por utilizar un método no paramétrico para sustentar nuestro análisis: la prueba de medianas de mood. Los resultados de esta prueba se observan en la tabla 5.

Mediante esta prueba, se evalúa la hipótesis de que las medias de ambos algoritmos son iguales. Debido a que valor-P es menor que 0,05, puede decirse con el 95% de confianza que las muestras son significativamente diferentes. Adicionalmente, se observa que los intervalos de confianza son mutuamente excluyentes, siendo el de mejor desempeño el que se encuentra más a la izquierda, en este caso, el algoritmo N2.

Tabla 5. Prueba de medinas de mood por algoritmo

ALGORITMO	*Tamaño de Muestra*	*n<=*	*n>*	*Mediana*	*LC inferior 95,0%*	*LC superior 95,0%*
GENETICO	120	4	116	0,8592	0,71051	1,061
N2	120	116	4	-0,8728	-0,8865	-0,861
Estadístico = 209,067					Valor-P = 0,0	

Con base en la prueba de medianas de mood, se infiere que la metaheurística N2 presenta un mejor desempeño en términos de calidad de soluciones que el algoritmo genético en el problema planteado.

4. CONCLUSIONES

En el presente artículo se formuló un modelo de programación entera mixta para la distribución de productos perecederos que minimiza el tiempo total de viaje y las tardanzas en la entrega de los mismos. El modelo fue resuelto con dos algoritmos aproximados: el algoritmo genético y un nuevo algoritmo híbrido (N2). El estudio de este problema cobra importancia por la naturaleza altamente perecedera de los alimentos, aspecto que causa que se deterioren con el paso del tiempo e impacta

en la calidad con la que los productos llegan a los consumidores finales. Adicionalmente, por ser un problema complejo resulta útil la utilización de metaheurísticas que puedan generar soluciones al problema en tiempos razonables para grandes instancias.

En los resultados fue posible observar que el modelo formulado puede generar soluciones factibles para el problema y que el nuevo algoritmo hibrido propuesto es capaz de generar mejores soluciones (con un 95% de confianza) que el algoritmo genético. En general el algoritmo N2 mostró mejores resultados en todas las instancias evaluadas lo que lo convierte en una buena herramienta para la planeación de rutas.

Futuras investigaciones pueden centrar sus esfuerzos en probar el modelo propuesto y el nuevo algoritmo en instancias de mayor tamaño y comparar los resultados con otras metaheurísticas de la literatura. De igual forma es posible mejorar la formulación del modelo con la inclusión de otras variables de la vida real, como la consideración de múltiples productos, ventanas de tiempo, modelos de deterioro del producto, flota heterogénea, entre otros.

REFERENCIAS

[1]. M. S. Ahmad and M. W. Siddiqui, "Postharvest Quality Assurance of Fruits," no. Kader 2002, p. 279, 2015.

[2]. M. Kasso and A. Bekele, "Post-harvest loss and quality deterioration of horticultural crops in Dire Dawa Region, Ethiopia," J. Saudi Soc. Agric. Sci., 2016.

[3]. C. I. Hsu, S. F. Hung, and H. C. Li, "Vehicle routing problem with time-windows for perishable food delivery," J. Food Eng., vol. 80, no. 2, pp. 465–475, 2007.

[4]. G. S. Mohapatra, Models for Practical Routing Problems in Logistics. 2014.

[5]. J.-F. Cordeau, G. Laporte, M. W. P. Savelsbergh, and D. Vigo, "Chapter 6 Vehicle Routing," vol. 14, no. 6, pp. 367–428, 2007.

[6]. M. Fisher, "Vehicle Routing," vol. 8, 1995.

[7]. P. Amorim and B. Almada-Lobo, "The impact of food perishability issues in the vehicle routing problem," Comput. Ind. Eng., vol. 67, no. 1, pp. 223–233, 2014.

[8]. J. Faulin, "Applying MIXALG procedure in a routing problem to optimize food product delivery," Omega, vol. 31, no. 5, pp. 387-395, 2003.

[9]. C. D. Tarantilis and C. T. Kiranoudis, "A meta-heuristic algorithm for the efficient distribution of perishable foods," J. Food Eng., vol. 50, no. 1, pp. 1-9, 2001.

[10]. C. D. Tarantilis and C. T. Kiranoudis, "Distribution of fresh meat," J. Food Eng., vol. 51, no. 1, pp. 85-91, 2002.

[11]. M. Dyer and L. Stougie, "Computational complexity of stochastic programming problems," Math. Program., vol. 106, no. 3, pp. 423-432, 2006.

[12]. N. Prindezis, C. T. Kiranoudis, and D. Marinos-Kouris, "A business-to-business fleet management service provider for central food market enterprises," J. Food Eng., vol. 60, no. 2, pp. 203–210, 2003.

[13]. A. Osvald and L. Z. Stirn, "A vehicle routing algorithm for the distribution of fresh vegetables and similar perishable food," J. Food Eng., vol. 85, no. 2, pp. 285-295, 2008.

[14]. M. de Keizer, R. Haijema, J. M. Bloemhof, and J. G. A. J. van der Vorst, "Hybrid optimization and simulation to design a logistics network for distributing perishable products," Comput. Ind. Eng., vol. 88, pp. 26–38, 2015.

[15]. M. M. Solomon, "Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints," Oper. Res., vol. 35, no. 2, pp. 254–265, 1987.

[16]. J. H. (John H. Holland and J. H., Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence. MIT Press, 1992.

[17]. P. Li, J. He, D. Zheng, Y. Huang, and C. Fan, "Vehicle Routing Problem with Soft Time Windows Based on Improved Genetic Algorithm for Fruits and Vegetables Distribution," Discret. Dyn. Nat. Soc., vol. 2015, 2015.

[18]. T. Vidal, T. G. Crainic, M. Gendreau, N. Lahnrichi, and W. Rei, "A hybrid genetic algorithm for multidepot and periodic vehicle routing problems," Oper. Res., vol. 60, no. 3, pp. 611-624, 2012.

[19]. B. Saeidian, M. S. Mesgari, and M. Ghodousi, "Evaluation and comparison of Genetic Algorithm and Bees Algorithm for location-allocation of earthquake relief centers," Int. J. Disaster Risk Reduct., vol. 15, pp. 94-107, 2016.

[20]. Q. Liu and J. Xu, "A study on vehicle routing problem in the delivery of fresh agricultural products under random fuzzy environment," Int. J. Inf. Manag. Sci., vol. 19, no. 4, pp. 673-690, 2008.

[21]. V. Sivaram Kumar, M. R. Thansekhar, R. Saravanan, and S. Miruna Joe Amali, "Solving multi-objective vehicle routing problem with time windows by FAGA," Procedia Eng., vol. 97, pp. 2176-2185, 2014.

[22]. D. Toth, P. & Vigo, The Vehicle Routing Problem. 2002.

[23]. Y. Marinakis and M. Marinaki, "A hybrid genetic–Particle Swarm Optimization Algorithm for the vehicle routing problem," Expert Syst. Appl., vol. 37, no. 2, pp. 1446-1455, 2010.

[24]. R. Sabie and A. Mestra, "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, 2011.

[25]. J. Kennedy, J. F. Kennedy, R. C. Eberhart, and Y. Shi, Swarm intelligence. Morgan Kaufmann, 2001.

[26]. M. Clerc, "Discrete particle swarm optimization, illustrated by the traveling salesman problem," in New optimization techniques in engineering, Springer, 2004, pp. 219–239.

[27]. F. Glover, "Tabu Search Part I," INFORMS J. Comput., vol. 1, no. 5, pp. 190–206, 1989.

[28]. F. Glover, "Tabu search, Part II," ORSA J. Comput., vol. 2, no. 1, pp. 4–32, 1990.

[29]. B. M. Batista and F. Glover, "Introducción a la Búsqueda Tabú," vol. 3, pp. 1–36, 2006.

[30]. R. Sabie and J. Lopez, "Implementation of the new algorithm chromatic metaheuristic a nonlinear regression model in time series forecasts," 2011, vol. IX Congres.

[31]. F. Hernández and J. Poveda, "Aplicación de la metaheurística cromática al problema de secueciación de proyectos con recursos limitados (RCPSP)," Universidad de Córdoba, Montería, Colombia, 2014.

[32]. M. Arrieta and E. López, "Diseño de un algoritmo de optimización para el problema de asignación de horarios en la Universidad de Córdoba," Universidad de Córdoba, Montería, Colombia, 2013.

[33]. D. Montgomery, Diseño y análisis de experimentos, 2a. ed. México: Limusa Wiley, 2008.