Aplicación de la Metaheurística Búsqueda de la Armonía para Resolver el Problema de Ruteo de Vehículos con Inventarios

Ing. Ramiro Saltos Atiencia, Dr. Ricardo Aceves García

Universidad Nacional Autónoma de México
División de Estudios de Posgrado
Facultad de Ingeniería
Sección de Investigación de Operaciones
ramirojsaltos@hotmail.com
aceves@unam.mx

Resumen. *En el presente trabajo se presenta la teoría básica de la metaheurística Búsqueda de la Armonía, la cual basa su filosofía de optimización en el proceso de aprendizaje utilizado los músicos al desarrollar una nueva melodía musical. Así mismo, esta mantiene su simplicidad de interpretación e implementación de tal forma que el investigador pueda dedicar más tiempo al desarrollo de mejores criterios de búsqueda que a la implementación computacional del algoritmo. Finalmente, se aplican los conceptos de esta metaheurística para resolver el bien conocido Problema de Ruteo de Vehículos con Inventarios (IRP por sus siglas en inglés) considerando que el modelo matemático que describe al problema es de programación entera mixta.*

Palabras Clave: *Metaheurística Búsqueda de la Armonía, IRP.*

1. Introducción

La metaheurística Búsqueda de la Armonía fue introducida por primera vez por Zong Woo Geem [**1**] en el año 2001 como respuesta a la creciente necesidad de desarrollar algoritmos de optimización para problemas de programación no lineal que no requieran información substancial sobre el gradiente de la función de objetivo. La filosofía de esta metaheurística se basa en el proceso musical para buscar un estado perfecto de la armonía. La armonía en la música es análoga con el vector de soluciones del problema de optimización, y el proceso de mejora de los músicos es comparable con los esquemas de búsqueda local y global de las técnicas de optimización [**2**].

El algoritmo utiliza un procedimiento de búsqueda estocástica aleatorizado basado en la información que se tiene en la memoria junto con otros parámetros que se definirán más adelante, de tal manera que la información del gradiente y el uso de las derivadas de la función objetivo del problema ya no es necesaria [**2**].

2. Descripción del Proceso de Optimización

Las actuaciones musicales tratan de encontrar una armonía agradable guiados por un patrón estético, tal y como, los procesos de optimización buscan encontrar la solución global a un problema guiados por la función objetivo. El tono de cada instrumento musical determina una calidad estética, de igual forma que el valor de la función objetivo es determinado por el conjunto de valores asignados a cada variable de decisión [**2**].

En las prácticas de los grupos musicales, cada miembro del grupo toca una nota seleccionada al azar de un rango posible de opciones, haciendo todos juntos una armonía (*Harmony Vector*). Si la armonía entonada es buena, la experiencia es recordada en la memoria de cada miembro (*Harmony Memory*) y la posibilidad de entonar una mejor armonía se incrementa para la próxima vez. De manera similar, en optimización, cada variable de decisión toma un valor seleccionado al azar dentro un intervalo posible, haciendo todas juntas un vector solución. Si todas las variables de decisión forman una buena solución del

problema, el resultado es almacenado en la memoria de cada variable y la posibilidad de formar una mejor solución la próxima iteración aumenta [**2**].

Cuando un músico mejora un tono, usualmente sigue una de las siguientes tres reglas:

- Toca cualquier tono que se encuentre almacenado en su memoria.
- Toca un tono adyacente al tono actual basado en su memoria y en un ajuste de tono.
- Toca un nuevo tono al azar.

De manera similar, cuando una variable de decisión toma un valor dentro de la metaheurística Búsqueda de la Armonía, sigue uno de estos tres pasos:

- Toma un valor seleccionado al azar de entre los que se encuentran almacenados en la memoria de la variable.
- Toma un valor adyacente a uno de los valores que se encuentre en la memoria de la variable, determinado por el ancho de banda.
- Toma un valor seleccionado al azar dentro de un intervalo posible.

Las tres reglas en el algoritmo son dirigidas de manera efectiva por medio de la utilización de los siguientes parámetros:

- La probabilidad de consideración de memoria $(HMCR)$.
- La probabilidad de ajuste de tono (PAR).
- El ajuste de ancho de banda (Bw).

Utilizando la harmony memory y las tres reglas mencionadas se puede crear una nueva armonía, la cual es análoga con encontrar una nueva solución del problema de optimización. Si la nueva armonía es mejor que alguna de las que se encuentran almacenadas en la harmony memory, ésta reemplaza a la peor de todas las armonías almacenadas. Caso contrario, se crea una nueva armonía siguiendo el procedimiento descrito. Se repiten los pasos anteriores hasta que el criterio de parada sea cumplido.

En resumen, la metaheurística Búsqueda de la Armonía se sintetiza en los siguientes pasos:

1. Inicializar los parámetros del algoritmo $(HMCR, PAR, Bw)$.
2. Establecer el tamaño de la memoria (HMS) e inicializar la harmony memory (HM).
3. Crear una nueva armonía a partir de la harmony memory.
4. Actualizar la harmony memory.
5. Repetir los pasos 3 y 4 hasta que el criterio de parada sea cumplido.

3. Pseudocódigo de la Metaheurística Búsqueda de la Armonía

Sea $x = (x_1, x_2, \dots, x_n)$ el vector de variables de decisión del problema de optimización:

$$Max\ o\ Min\ z = f(x)$$
$$S.t:$$
$$LB_i \leq x_i \leq UB_i \qquad \forall i = 1,2, \dots, n$$

Donde LB_i y UB_i son las cotas inferior y superior asociadas a la variable de decisión x_i.

Paso 1

Se inicializan los parámetros del algoritmo:

- $HMCR = 0.9, HMS = 10$
- $PAR = 0.5, Bw = 100$

Paso 2

Inicializar la harmony memory creando armonías aleatorias utilizando el siguiente procedimiento:

$$Para\ k = 1\ hasta\ k = HMS$$
$$Para\ i = 1\ hasta\ i = n$$

$$x_i^k = y \sim U(LB_i, UB_i)$$
$$Fin\ Para$$
$$Fin\ Para$$

Paso 3

Crear una nueva armonía considerando los parámetros del algoritmo y la harmony memory. Para ello se utiliza el siguiente procedimiento:

$$Para\ i = 1\ hasta\ i = n$$
$$Si\ r \sim U(0,1) \leq HMCR$$
$$x'_i = x_i^k\ donde\ k \sim U(1, HMS)$$
$$Si\ u \sim U(0,1) \leq PAR$$
$$x'_i = x'_i + r \cdot Bw\ donde\ r \sim U(-1,1)$$
$$Fin\ Si$$
$$Caso\ contrario$$
$$x'_i = y \sim U(LB_i, UB_i)$$
$$Fin\ Si$$
$$Fin\ Para$$

Paso 4

Comparar la nueva armonía x' con la peor armonía almacenada en la harmony memory. Si x' es mejor que la que se tiene almacenada, reemplazar la peor armonía por x'.

Paso 5

Repetir los pasos 3 y 4 hasta que se alcance el criterio de parada establecido. La mejor solución almacenada en la harmony memory será la solución final encontrada para el problema de optimización que se esté resolviendo.

4. Variantes de la Metaheurística Búsqueda de la Armonía

Desde su reaparición en el año 2005, algunas variantes han sido propuestas para mejorar el rendimiento del algoritmo original desarrollado por Geem. Las variantes más conocidas son:

4.1. Improved Harmony Search [3]

La cual propone un método adaptativo para ir actualizando algunos de los parámetros del algoritmo. El modelizador sólo necesita inicializar los parámetros $PAR_{min}, PAR_{max}, Bw_{min}$ y Bw_{max} y, a través del método propuesto, se irán calculando iteración tras iteración los valores reales de PAR y Bw. La principal desventaja de este método radica en que se necesita inicializar los valores de Bw_{min} y Bw_{max} los cuales son muy dependientes del problema y difíciles de ajustar.

4.2. Global Best Harmony Search [4]

La cual propone, en lugar de actualizar iterativamente el valor de Bw, eliminarlo del algoritmo y realizar el ajuste de tono asignándole a la variable de decisión que se está explorando, el valor que toma una de las variables de decisión (seleccionada de manera aleatoria) que forma parte de la mejor armonía almacenada hasta esa iteración en la harmony memory. Su principal desventaja radica en que las variables de decisión del problema pueden representar cosas completamente ajenas a lo que representa la variable que se explora en esa iteración.

5. Problema de Ruteo de Vehículos con Inventarios (IRP)

5.1. Origen del IRP

El Problema de Ruteo de Vehículos con Inventarios (IRP por sus siglas en inglés), nace en torno del ámbito logístico a raíz de la implantación, en medianas y grandes empresas, de los esquemas de Inventario Manejado por el Proveedor (VMI por sus siglas en inglés). Esta filosofía de trabajo consiste en

que el proveedor deberá monitorear constantemente los niveles de inventario de sus clientes, determinar cuándo reabastecerlos y en qué cantidad, así como elaborar la hoja de ruta que deberán recorrer sus camiones al momento de realizarse la distribución.

Este esquema logístico plantea la necesidad al proveedor, de integrar en un sólo modelo las decisiones asociadas con el ruteo de los vehículos, la cantidad de producto a enviar a cada cliente cada día y el nivel de inventario que se deberá mantener en las bodegas de los mismos, de tal forma que se minimicen todos los costos operativos asociados. Dada la creciente necesidad de desarrollar políticas óptimas de operación para este esquema logístico, en el año de 1984, Awi Federgruen y Paul Zipkin publican un artículo titulado *"A Combined Vehicle Routing and Inventory Allocation Problem,"* donde se propone un modelo integrado de ruteo de vehículos con localización de inventarios, dando origen a lo que hoy se conoce como *Inventory Routing Problem*.

5.2. Descripción Matemática del IRP

El problema en su versión básica consiste en, dado un conjunto de clientes y un almacén central desde el cual se los atenderá, determinar cuál es el conjunto de rutas de distribución, la cantidad de producto a enviar a cada cliente junto con los niveles de inventario que se deben mantener, para cada día o instante de tiempo dentro del horizonte de planificación, de tal forma que, se minimicen los costos de inventario y distribución. Nótese que, a diferencia del problema clásico de ruteo de vehículos, la cantidad a enviar a cada cliente es ahora una variable de decisión por lo cual no necesariamente se debe visitar a todos los clientes todos los días, considerando que los niveles de inventario en las bodegas de los mismos, deben ser suficientes para cubrir la demanda para los días que no son atendidos.

En la Figura 1 se puede apreciar cómo suelen ser diseñadas las rutas en un modelo de IRP. En el día 1 todos los clientes son visitados estableciéndose dos rutas, mientras que en el día 2, sólo algunos de ellos son visitados estableciéndose una única ruta para atenderlos. Este plan de visitas implica que los clientes que no son visitados en el día 2, tienen el suficiente nivel de stock en sus bodegas para cubrir la demanda de ese día.

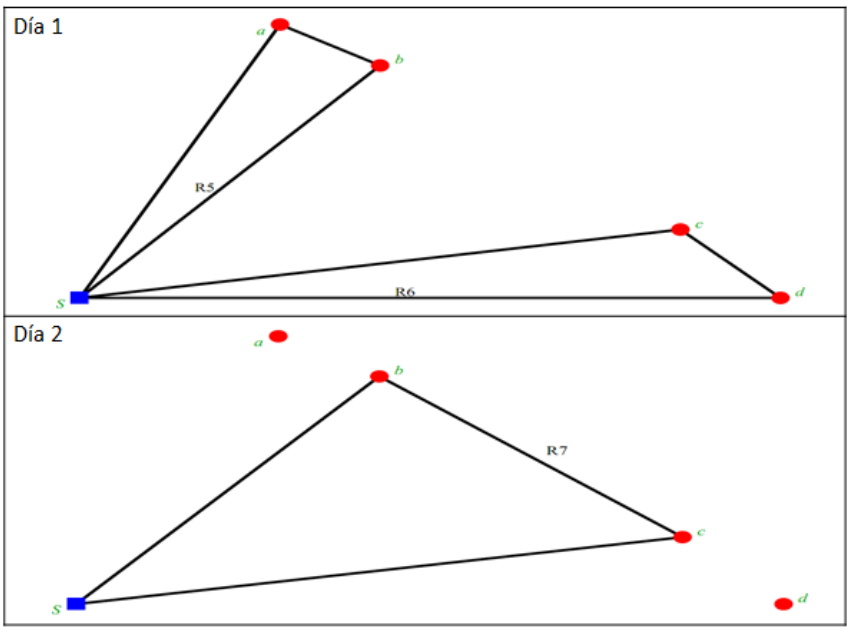

Figura 1: Plan de Ruteo del IRP

El modelo de programación matemática propuesto por Federgruen y Zipkin [**5**] considera un horizonte de tiempo continuo y una función objetivo no lineal, lo cual hace al problema muy difícil de tratar en periodos razonables de tiempo. Una formulación de programación lineal entera mixta se menciona en [**6**], sin embargo, ésta considera que se conocen todas las posibles rutas con anticipación y plantea un esquema de selección de ruta para el día que el cliente deba ser visitado, lo cual es difícil de manejar en

problemas de gran escala, debido a que generar todas las rutas posibles exige demasiado tiempo computacional y no nos ayuda a resolver el problema de interés.

Los modelos de programación entera mixta considerando un horizonte de tiempo finito y discreto, facilitan en cierto grado el manejo de problema, debido a que se puede descomponer con base en dos decisiones; determinar qué clientes van a ser visitados cada día junto con la cantidad de producto que van a recibir, para luego decidir cómo rutear los vehículos para visitar dichos clientes. Este enfoque plantea la resolución de un problema de ruteo de vehículos (VRP) para cada instante de tiempo, el cual es un problema de la clase NP-Hard [**7**], por lo cual se puede deducir inmediatamente que el IRP también pertenece a la clase de problemas NP-Hard.

El Problema de Ruteo de Vehículos con Inventarios (IRP), ha sido ampliamente estudiado desde dos principales enfoques, el determinístico y el probabilístico. La principal diferencia entre estos radica en que la demanda de los clientes en el primer caso es conocida con certeza, mientras que en el segundo se ajusta a una distribución de probabilidad. De igual forma, se han propuesto varias estrategias de solución, que van desde el uso de métodos exactos como los algoritmos de *Branch and Bound*, hasta la implementación de métodos metaheurísticos como la Búsqueda Tabú. El lector podrá revisar los trabajos relacionados con estos temas en la referencias [**6**], [**8**], [**9**] y [**10**].

5.3. Modelo Matemático del IRP

Para la formulación del modelo matemático se tomó como base el modelo de tres índices para el Problema de Ruteo de Vehículos (VRP por sus siglas en inglés) presentado en [**7**], y la consideración de que el IRP puede ser visto como un VRP multiperiodo.

Datos

- $V = \{0,1,2,\dots,n\}$ es el conjunto de nodos, clientes. El nodo $i = 0$ representa el depósito central.
- $T = \{1,2,\dots,H\}$ es el conjunto finito y discreto de los instantes de tiempo pertenecientes al horizonte de planificación considerado.
- $K = \{1,2,\dots,m\}$ es el conjunto de vehículos disponibles con capacidad de transporte homogénea.
- $CapV$ es la capacidad de carga de los vehículos.
- $CapC$ es la capacidad de almacenamiento de los clientes.
- r es la tasa de mantenimiento del inventario.
- $c_{i,j}$ es el costo de transporte desde el nodo i al nodo j.
- Dem_i^t es la demanda del cliente i en el instante de tiempo t.

Variables de Decisión

- $Q_{i,k}^t$ es la cantidad de producto enviado al cliente i usando el vehículo k en el tiempo t.
- s_i^t es el nivel de inventario almacenado en el cliente i en el tiempo t.
- $y_{i,k}^t = \begin{cases} 1, & si\ el\ cajero\ i\ es\ visitado\ por\ el\ vehículo\ k\ en\ el\ tiempo\ t \\ 0, & si\ no \end{cases}$
- $x_{i,j}^{k,t} = \begin{cases} 1, & si\ el\ arco\ (i,j)\ es\ usado\ por\ el\ vehículo\ k\ en\ el\ tiempo\ t \\ 0, & si\ no \end{cases}$

Modelo Matemático

$$Min\ z = \sum_i \sum_j \sum_k \sum_t c_{i,j} x_{i,j}^{k,t} + \sum_i \sum_t r s_i^t \qquad (5.1)$$

$S.t$:

$$\sum_k y_{i,k}^t \leq 1 \qquad \forall i \in V\backslash\{0\}\ \forall t \in T \qquad (5.2)$$

$$\sum_k y_{0,k}^t \leq m \qquad \forall t \in T \qquad (5.3)$$

$$\sum_j x_{i,j}^{k,t} = y_{i,k}^t \qquad \forall i \in V\ \forall k \in K\ \forall t \in T \qquad (5.4)$$

$$\sum_j x_{i,j}^{k,t} = \sum_j x_{j,i}^{k,t} \qquad \forall i \in V\ \forall k \in K\ \forall t \in T \qquad (5.5)$$

$$\sum_{i\neq 0} Q_{i,k}^t \leq CapV \qquad \forall k \in K\ \forall t \in T \qquad (5.6)$$

$$Q_{i,k}^t \leq CapC - s_i^{t-1} + M\left(1 - y_{i,k}^t\right) \qquad \forall i \in V\backslash\{0\}\ \forall k \in K\ \forall t \in T \qquad (5.7)$$

$$Q_{i,k}^t \leq CapC \cdot y_{i,k}^t \qquad \forall i \in V\backslash\{0\}\ \forall k \in K\ \forall t \in T \qquad (5.8)$$

$$s_i^t = s_i^{t-1} + \sum_k Q_{i,k}^t - Dem_i^t \qquad \forall i \in V\backslash\{0\}\ \forall t \in T \qquad (5.9)$$

$$\sum_{i\in S}\sum_{j\in S} x_{i,j}^{k,t} \leq |S| - 1 \qquad \forall S \subseteq V\ |S| \geq 2\ \forall k \in K\ \forall t \in T \qquad (5.10)$$

$$Q_{i,k}^t \geq 0 \qquad \forall i \in V\backslash\{0\}\ \forall k \in K\ \forall t \in T \qquad (5.11)$$

$$s_i^t \geq 0 \qquad \forall i \in V\backslash\{0\}\ \forall t \in T \qquad (5.12)$$

$$y_{i,k}^t \in \{0,1\} \qquad \forall i \in V\ \forall k \in K\ \forall t \in T \qquad (5.13)$$

$$x_{i,j}^{k,t} \in \{0,1\} \qquad \forall i,j \in V\ \forall k \in K\ \forall t \in T \qquad (5.14)$$

La ecuación (5.1) busca minimizar los costos de ruteo y de inventario, la (5.2) asegura que en cada instante de tiempo un cliente sea visitado máximo una vez, la (5.3) impone que del depósito no salgan más vehículos de los que se tienen disponibles en cada instante de tiempo, la (5.4) y la (5.5) son las restricciones de grado de los nodos de la red, la (5.6) garantiza que la capacidad de los vehículos no sea excedida, la (5.7) y la (5.8) aseguran que un cliente que no es visitado en un determinado instante de tiempo, no reciba dinero mientras que el que sí es visitado no reciba un monto superior a su capacidad menos el inventario que ya poseía almacenado, la (5.9) garantiza la continuidad en el flujo de inventario a través del tiempo, la (5.10) es la restricciones de eliminación de subciclos similares a las referenciadas en [**7**]. Finalmente, las ecuaciones de la (5.11) a la (5.14) imponen las condiciones de signo y lógica que deben cumplir las variables de decisión.

Un modelo similar al presentado en esta sección fue propuesto en [**11**], sin embargo, sus principales diferencias radican en que no se consideran en la función objetivo los costos de mantenimiento del inventario y que se utiliza una función de demanda acumulada para garantizar el abastecimiento de cada cliente y así no exceder la capacidad del mismo.

6. Algoritmos para Ruteo de Vehículos

En la presente sección se describirán brevemente dos de los algoritmos más conocidos y fáciles de implementar para resolver el problema del agente viajero (TSP por sus siglas en inglés). El primero de ellos es un algoritmo constructivo, el cual, iteración tras iteración, va construyendo una solución para el problema que se desea resolver. El segundo, es un algoritmo de mejora, el cual, está basado en procedimientos de búsqueda local que exploran una porción del espacio de soluciones que se encuentra en los "alrededores" de una solución inicial dada. Las soluciones aportadas por los algoritmos de búsqueda local no necesariamente son soluciones óptimas.

Aunque estos algoritmos fueron diseñados exclusivamente para encontrar buenas soluciones al problema del agente viajero, se pueden adaptar para resolver rápidamente el problema de ruteo de vehículos (VRP por sus siglas en inglés), dado que el TSP no es más que un caso particular del VRP suponiendo que la capacidad de los vehículos es infinita.

6.1. Algoritmo de Inserción

Es uno de los algoritmos constructivos mayormente utilizados para el TSP, y que en general, suele reportar mejores soluciones que la heurística del vecino más cercano. Se ha demostrado a través de experimentos empíricos con instancias de prueba de dificultad teórica controlada, que el algoritmo de inserción, en cualquiera de sus tres variantes, reporta soluciones con una desviación promedio por debajo del 15%, mientras que para la heurística del vecino más cercano es del 20% [**12**].

Este algoritmo empieza con un ciclo que contenga al menos 3 elementos (escogidos o no al azar) del conjunto de ciudades a visitar. Luego, la siguiente ciudad en agregarse al ciclo es seleccionada entre aquellas que no forman parte aún de la solución construida hasta el momento utilizando cualquiera de los siguientes criterios:

- ***Inserción Ciudad Más Cercana:*** Consiste en seleccionar, de entre todas las ciudades candidatas a entrar, a la ciudad que se encuentra más cercana al ciclo parcialmente construido, es decir, la que se encuentre más cerca de cualquiera de los nodos que forman parte de la solución parcial actual [**13**].
- ***Inserción Ciudad Más Lejana:*** Consiste en seleccionar, de entre todas las ciudades candidatas a entrar, a la ciudad que se encuentra más lejos del ciclo parcialmente construido, es decir, la que se encuentre más lejos de cualquiera de los nodos que forman parte de la solución parcial actual [**13**].
- ***Aleatoria:*** Consiste en seleccionar al azar cualquiera de las ciudades que son candidatas a entrar al ciclo parcialmente construido [**13**].

Una vez determinada la ciudad que se va a insertar en el ciclo, la pregunta es en dónde hay que insertarla. El algoritmo de inserción se fundamenta principalmente en el hecho de que la inserción debe hacerse entre las ciudades donde el costo marginal sea el menor de entre todas las opciones posibles. El costo marginal se calcula a partir de la expresión:

$$s_{i,j} = c_{i,k} + c_{k,j} - c_{i,j}$$

Donde i, j pertenecen al ciclo construido hasta el momento y k representa la ciudad que se va a agregar a la solución actual. El pseudocódigo de este algoritmo es:

$Inicio$
  $Genere\ un\ ciclo\ C = \{v_1, v_2, v_3\}\ escogiendo\ v_1, v_2, v_3\ al\ azar$
  $Mientras\ (|C| \neq n)$
    $Escoja\ un\ nodo\ k \in V/C\ \ usando\ cualquier\ criterio\ mencionado$
    $Para\ cada\ posible\ inserción\ i, j \in C\ con\ i \neq j\ calcule\ s_{i,j} = c_{i,k} + c_{k,j} - c_{i,j}$
    $Inserte\ el\ nodo\ k\ entre\ los\ nodos\ i, j\ con\ menor\ valor\ de\ s_{i,j}$
  $Fin\ Mientras$
  $C = C \cup \{v_1\}$
  $Return\ C$
Fin

6.2. Algoritmo de Mejora 2-opt

El algoritmo de mejora 2-Opt, a partir de una solución inicial creada con cualquier método constructivo, toma dos aristas no adyacentes de la misma y las intercambia para así obtener una nueva solución la cual se dice que es "vecina" a la actual. Si la nueva solución es mejor que la actual, se la reemplaza y se prosigue a realizar otro intercambio. El algoritmo termina cuando todos los posibles intercambios han

sido explorados. El porcentaje de desviación respecto a la solución óptima de acuerdo a los experimentos realizados en [**12**], considerando como solución inicial la aportada por el vecino más cercano, es menor al 10%. El pseudocódigo de esta heurística, tomado de [**12**], es:

```
Inicio
    opt = 1
    Mientras(opt = 1)
        opt = 0
        Etiquetar todos los vértices como no explorados
        Mientras(Queden vértices por explorar)
            Seleccionar un vértice i no explorado
            Examinar todos los intercambios 2 − opt que puedan realizarse
            entre la arista de i a su sucesor en el ciclo
            Si alguno de los intercambios mejora la solución actual
                Realizar el mejor de todos los intercambios
                opt = 1
            Caso contrario
                Etiquetar el vértice i como explorado
            Fin Si
        Fin Mientras
    Fin Mientras
Fin
```

7. Desarrollo del Algoritmo de Solución

En este capítulo se describirá de forma detallada cómo está compuesto el algoritmo que se desarrolló para resolver el problema abordado en esta investigación, el cual se encuentra basado en la metaheurística Búsqueda de la Armonía, cuya filosofía guiará el proceso de optimización. Se utilizarán los algoritmos de inserción y mejora 2-opt para encontrar las soluciones a los problemas de ruteo generados en cada iteración. Luego se validará el algoritmo utilizando dos instancias diseñadas para ello y finalmente, se presentarán los resultados obtenidos al aplicarse la metodología desarrollada al caso concreto que se aborda.

El algoritmo consta de dos etapas, una donde se construyen una o varias soluciones del problema, y otra, donde a partir de dichas soluciones, se construyen soluciones mejoradas. Cada solución generada recibe el nombre de armonía, y está compuesta por:

- Política de envío de dinero a los clientes.
- Nivel de inventario guardado en los clientes.
- Rutas de distribución para cada día.
- Costo de ruteo.
- Costo de inventario.
- Costo total.

7.1. Etapa 1: Creación de la Solución Inicial

Esta etapa consta de dos fases: la primera se encarga de establecer la política de inventario de los clientes de la red, mientras que la segunda toma las decisiones asociadas al ruteo tomando en cuenta la política de inventario generada en la fase 1.

7.1.1. Fase 1: Política de Inventario

La fase 1 del algoritmo se encarga de tomar las decisiones relacionadas con la política de inventario a manejar en los clientes, es decir, determina qué día deben ser reabastecidos y cuál es la cantidad de producto a enviar a cada uno de ellos, de tal manera que se cumplan las restricciones de capacidad de almacenamiento de los clientes y se cubra toda la demanda estimada por el administrador de la red, o el gerente de la empresa.

El algoritmo desarrollado utiliza el procedimiento que propuesto en la metaheurística mencionada con anterioridad para crear una armonía, el cual consiste en:

Paso 0
Hacer $i = 1$

Paso 1
Hacer $t = 1$

Paso 2
Si $s_i^{t-1} \geq Dem_i^t$, asignar el valor de cero a las variables de decisión Q_i^t y y_i^t. Caso contrario, asignar a la variable de decisión Q_i^t un número aleatorio uniformemente distribuido entre cero y la capacidad máxima del cliente menos el inventario guardado al final del día anterior, y asignar el valor de 1 a la variable binaria y_i^t.

Paso 3
Actualizar el valor del inventario guardado en el cliente i al final del día t utilizando la expresión $s_i^t = s_i^{t-1} + Q_i^t - Dem_i^t$ e incrementar el valor del índice t haciendo $t = t + 1$.

Paso 4
Si $t > H$, aumentar el valor del índice i haciendo $i = i + 1$ y volver al paso 1. Caso contrario, volver al paso 2.

Una vez que se han explorado todos los clientes de la red, se tiene almacenado en las variables de decisión Q_i^t y y_i^t las cantidades de producto a enviar y cuáles clientes son atendidos en cada día, respectivamente. Estas variables de decisión son enviadas a la segunda fase como datos de entrada para resolver el problema de ruteo asociado a la política de inventario generada en la fase 1.

7.1.2. Fase 2: Política de Distribución o Ruteo

Al haberse determinado en la fase anterior las cantidades de producto a enviar, y los clientes que deben ser visitados, el problema se simplifica al tener que resolver un problema de ruteo de vehículos por cada instante de tiempo. Para esta fase, primero se construye una solución inicial utilizando el algoritmo de inserción adaptado, considerando únicamente los clientes que cuya variable de decisión y_i^t toma el valor de uno, debido a que esto nos indica cuáles se deben atender ese día, y la demanda de los mismos es la asociada a la variable Q_i^t.

Una vez obtenida esta solución inicial, las rutas que la conforman son mejoradas utilizando el algoritmo 2-opt, con lo cual se determina el costo total de ruteo asociado a la política de inventario determinada en la fase 1. El procedimiento para crear las rutas iniciales consiste en:

Paso 0

- Hacer $t = 1$

Paso 1

- Inicializar el vector de demandas dem_i^t con los valores Q_i^t correspondientes.
- Inicializar el conjunto V' con todos los clientes cuya variable de decisión y_i^t sea igual a 1.
- Inicializar el conjunto de nodos explorados $nodExp = \emptyset$.
- Inicializar el conjunto de rutas $setR_t = \emptyset$.

Paso 2

- Inicializar la capacidad del vehículo $CapV$.
- Hacer $W = V' \backslash nodExp$.
- Hacer $D = \{i \in W / \, dem_i^t > CapV\}$.
- Hacer $C = W \backslash D$.

- Inicializar el conjunto $ruta = \{0,0\}$.

Paso 3

Mientras $C \neq \emptyset$, hacer:

- Si $Card(ruta) \leq 3$, entonces:
  - Seleccionar al azar un cliente del conjunto C.
  - Agregarlo al conjunto de nodos explorados.
  - Agregarlo a la ruta actual.
  - Restar la demanda de ese cliente de la capacidad del vehículo.
  - Hacer $W = V' \backslash nodExp$.
  - Hacer $D = \{i \in W / \, dem_i^t > CapV\}$.
  - Hacer $C = W \backslash D$.
- Caso contrario:
  - Seleccionar al azar un cliente del conjunto C.
  - Agregarlo al conjunto de nodos explorados.
  - Calcular los costos marginales $s_{i,j}$ asociados con la inserción del nodo seleccionado entre cada par de nodos $i, j \in ruta$.
  - Insertar en la ruta el nodo seleccionado entre el par de nodos con menor valor de $s_{i,j}$.
  - Restar la demanda de ese cliente de la capacidad del vehículo.
  - Hacer $W = V' \backslash nodExp$.
  - Hacer $D = \{i \in W / \, dem_i^t > CapV\}$.
  - Hacer $C = W \backslash D$.

Paso 4

- Agregar la ruta encontrada al conjunto de rutas $setR_t$.
- Si $nodExp \neq V'$, volver al paso 2, caso contrario, guardar el conjunto de rutas hallado e incrementar el valor del índice t haciendo $t = t + 1$.

Paso 5

- Si $t \leq H$, volver al paso 1, caso contrario finalizar y devolver las rutas generadas durante todo el proceso.

Al finalizar esta sub-fase, se tienen las rutas asociadas a la política de inventario determinada en la etapa 1 considerando únicamente aquellos clientes que deben ser visitados. La siguiente sub-fase de esta etapa, consiste en aplicar el algoritmo de mejora 2-opt a estas rutas, utilizando el siguiente procedimiento:

Paso 0

- Hacer $t = 1$

Paso 1

- Extraer el conjunto de rutas $setR_t$ determinadas para el instante de tiempo t.
- Aplicar el algoritmo de mejora 2-opt a cada ruta en el conjunto $setR_t$.

Paso 2

- Actualizar el conjunto de rutas $setR_t$ con las rutas mejoradas e incrementar el valor del índice t haciendo $t = t + 1$.

Paso 3

- Si $t \leq H$, volver al paso 1, caso contrario, finalizar y devolver las rutas mejoradas.

Al finalizar la etapa 1 se cuenta con una solución inicial factible para el problema de ruteo de vehículos con inventarios (IRP por sus siglas en inglés). Hay que notar que esta etapa propone una heurística sencilla y rápida para crear soluciones al problema tratado, sin embargo, es difícil determinar la calidad

de las soluciones generadas, dado que no existen problemas de prueba reportados en la literatura cuyas soluciones óptimas sean conocidas y la topología de red se asemeje a la tratada en el presente trabajo.

7.2. Etapa 2: Proceso de Optimización

En esta etapa se generaran nuevas y mejores soluciones para el problema tratado utilizando como guía el algoritmo principal que se describe en la metaheurística Búsqueda de la Armonía. Las soluciones generadas en la etapa 1 se almacenarán en la estructura computacional de la metaheurística conocida como *Harmony Memory*. La harmony memory puede almacenar tantas soluciones factibles del problema como se desee, sin embargo, la literatura recomienda, casi para todos los problemas de optimización, se almacenen 10 soluciones. Esta cantidad de soluciones no depende del tamaño del problema, teniendo empíricamente casi la misma calidad en los resultados si el problema es de gran escala o no lo es.

Otra de las cuestiones a resolver en el algoritmo tiene que ver con la calibración de los parámetros que utiliza para guiar el proceso de aprendizaje y mejora de las soluciones del problema. Los artículos referenciados en [**1**], [**2**], [**3**] y [**4**] recomiendan calibrarlos con los siguientes valores:

- $HMCR = 0.9$
- $PAR_{min} = 0.01$
- $PAR_{max} = 0.99$

Hay que tener en cuenta que el algoritmo desarrollado en la presente investigación, utilizará el mecanismo auto-adaptativo propuesto en [**3**] para ir calculando el valor del parámetro PAR a través de la expresión:

$$PAR = PAR_{min} + \frac{PAR_{max} - PAR_{min}}{NI} it$$

Donde NI es el número total de iteraciones a ejecutarse e it es el número de iteraciones ejecutadas hasta ese instante. Dado que este parámetro no depende de la naturaleza del problema, es más conveniente usar este método que fijarlo en un valor dado.

Otra de las consideraciones implementadas en el desarrollo del algoritmo es la eliminación del parámetro Bw, a través de la utilización del método propuesto en [**4**] y mejorado en [**14**], el cual consiste en asignar a la variable de decisión que se esté explorando, el valor que toma en la mejor solución almacenada hasta el momento en la harmony memory.

El proceso de optimización general del algoritmo desarrollado, incluyendo todos los procedimientos anteriormente descritos, consiste en:

Paso 0

- Inicializar el número de iteraciones $NI = 150$
- Inicializar $PAR_{min} = 0.01$, $PAR_{max} = 0.99$ y $HMCR = 0.9$
- Inicializar el tamaño de la harmony memory $HMS = 10$
- Inicializar la harmony memory (HM) ejecutando el procedimiento descrito en las fases 1 y 2 hasta completar la cantidad de soluciones indicadas por el parámetro HMS
- Ordenar las soluciones en HM de menor a mayor según el costo total
- Hacer $it = 1$

Paso 1

- Calcular el valor de $PAR = PAR_{min} + \frac{PAR_{max} - PAR_{min}}{NI} it$
- Inicializar los conjuntos $Qty = \emptyset$ y $Stk = \emptyset$
- Hacer $i = 1$

Paso 2

- Generar un número aleatorio $u \sim U(0,1)$
  - Si $u \leq HMCR$, entonces:
    - Generar un número aleatorio $rn \sim U(1, HMS)$
    - Asignar a los conjuntos Q_i y S_i los valores que tiene la política de envío y de inventario del cliente i almacenada en la solución que se encuentra en la posición rn de la harmony memory.
    - Generar un número aleatorio $v \sim U(0,1)$ y si $v \leq PAR$
      - ✓ Asignar a los conjuntos Q_i y S_i los valores que tiene la política de envío y de inventario del cliente i almacenada en la mejor solución de la harmony memory.
  - Caso contrario:
    - Crear una nueva política de envío e inventario utilizando el procedimiento descrito en la fase 1 de la etapa 1, únicamente para el cliente i.

Paso 3

- Agregar los conjuntos Q_i y S_i a los conjuntos Qty y Stk, respectivamente.
- Incrementar el valor del índice i haciendo $i = i + 1$.

Paso 4

- Si $i \leq n$, entonces:
  - Volver al paso 2.
- Caso contrario:
  - Crear la política de distribución utilizando el procedimiento descrito en la fase 2 de la etapa 1, enviando como datos de entrada la política de envío de los cajeros almacenada en el conjunto Qty.

Paso 5

- Si la nueva solución es mejor que alguna de las que se encuentran almacenadas en la harmony memory, entonces:
  - Introducir la nueva solución a la harmony memory.
  - Eliminar la peor solución almacenada en la harmony memory.
  - Ordenar nuevamente las soluciones en la harmony memory de menor a mayor según el costo total.
  - Incrementar el valor de it haciendo $it = it + 1$
- Caso contrario:
  - Descartar la nueva solución.
  - Incrementar el valor de it haciendo $it = it + 1$

Paso 6

- Si $it \leq NI$, volver al paso 1, caso contrario, finalizar y devolver la harmony memory resultante del todo el proceso de optimización. La solución que se encuentre en la primera ubicación es la mejor solución hallada para el problema.

Hay que recalcar que la solución encontrada a través de la ejecución del presente algoritmo, no es necesariamente la solución óptima del problema, dado que este algoritmo es un procedimiento metaheurístico, por lo cual reporta buenas soluciones al problema en un tiempo de cómputo razonable.

8. Resultados Computacionales

Dada la topología del problema estudiado, no se encuentran disponibles instancias del modelo cuya solución óptima sea conocida, por ello, para tener una aproximación de la eficiencia del algoritmo desarrollado en este trabajo, se diseñaron 20 instancias aleatorias que tienen las siguientes características:

- ***Naturaleza de la Demanda:*** Determinística y dinámica (cambia con el tiempo).
- ***Función de Distancia:*** Métrica de Manhattan.

- ***Número de Clientes en la Red:*** entre 5 y 8 clientes con sus coordenadas cartesianas.
- ***Capacidad de Almacenamiento:*** 350000 unidades monetarias.
- ***Capacidad del Vehículo:*** 3 veces la capacidad de un cliente.
- ***Horizonte de Planificación:*** 7 días.
- ***Tasa de Inventario:*** 0.10 anual.

Las instancias se implementaron en el modelizador GAMS y resolvieron utilizando el solver CPLEX de IBM, cuya versión actual es 12.3.0.0. Los parámetros del solver fueron:

- ***Tiempo Máximo de Ejecución:*** 3600 segundos para las instancias de 5 a 7 clientes y 5400 segundos para las de 8 clientes.
- ***Número Máximo de Iteraciones:*** Las permitidas por el nivel de precisión de la máquina.
- ***Consumo de Memoria:*** Se almacenó el árbol en el disco duro de la computadora.

Las mismas instancias fueron resueltas por el algoritmo propuesto considerando los siguientes parámetros para el mismo:

- ***Tiempo Máximo de Ejecución:*** 3600 segundos para todas las instancias.
- ***Número Máximo de Iteraciones:*** 2000
- ***Tamaño de la Harmony Memory:*** 100
- ***HMCR:*** 0.90
- ***PARmin:*** 0.01
- ***PARmax:*** 0.99

En ambos casos, los programas se ejecutaron en una laptop HP Pavilion g4-1085la con 4gb de RAM y un procesador Intel Core i5 de 2.3 GHz. Los resultados obtenidos se muestran en las siguientes tablas, donde el tiempo está dado en segundos.

Tabla 1: Resultados para las Instancias de 5 Clientes

Nombre Instancia	GAMS/CPLEX				Metaheurística HS				GAP
	TOTAL	RUTEO	INV	TIEMPO	TOTAL	RUTEO	INV	TIEMPO	
IRP01N05	2519.14	1894	625.14	318.83	2519.14	1894	625.14	3.49	0.00%
IRP02N05	2209.75	1626	583.75	3493.39	2234.79	1824	410.79	3.58	1.13%
IRP03N05	1990.06	1372	618.06	293.37	1990.06	1372	618.06	6.06	0.00%
IRP04N05	2180.03	1548	632.03	3600.30	2264.12	1664	600.12	7.89	3.86%
IRP05N05	2017.53	1314	703.53	466.52	2061.16	1384	677.16	8.25	2.16%

Tabla 2: Resultados para las Instancias de 6 Clientes

Nombre Instancia	GAMS/CPLEX				Metaheurística HS				GAP
	TOTAL	RUTEO	INV	TIEMPO	TOTAL	RUTEO	INV	TIEMPO	
IRP01N06	2800.54	2198	602.54	3600.47	2858.48	2198	660.48	9.09	2.07%
IRP02N06	2402.36	1724	678.36	3600.53	2427.86	1672	755.86	9.39	1.06%
IRP03N06	2509.18	1994	515.18	594.23	2652.04	1898	754.04	14.82	5.69%
IRP04N06	1885.52	1438	447.52	3600.28	1939.07	1422	517.07	12.27	2.84%
IRP05N06	2160.42	1432	728.42	372.62	2160.42	1432	728.42	10.64	0.00%

Tabla 3: Resultados para las Instancias de 7 Clientes

Nombre Instancia	GAMS/CPLEX				Metaheurística HS				GAP
	TOTAL	RUTEO	INV	TIEMPO	TOTAL	RUTEO	INV	TIEMPO	
IRP01N07	2732.85	2258	474.85	3600.59	2787.63	2184	603.63	13.72	2.00%

IRP02N07	3070.93	2380	690.93	3601.65	3096.15	2404	692.15	15.80	0.82%
IRP03N07	2213.99	1454	759.99	3601.34	2306.48	1834	472.48	15.11	4.18%
IRP04N07	3729.15	2890	839.15	3600.31	3726.14	3106	620.14	16.91	-0.08%
IRP05N07	3235.58	2530	705.58	3600.40	3332.97	2578	754.97	22.41	3.01%

Tabla 4: Resultados para las Instancias de 8 Clientes

Nombre Instancia	GAMS/CPLEX				Metaheurística HS				GAP
	TOTAL	RUTEO	INV	TIEMPO	TOTAL	RUTEO	INV	TIEMPO	
IRP01N08	2929.13	2192	737.13	5400.72	3122.65	2298	824.65	25.97	6.61%
IRP02N08	2937.71	2258	679.71	5400.80	2976.24	1974	1002.24	37.19	1.31%
IRP03N08	3053.79	2510	543.79	5401.59	3330.72	2842	488.72	32.99	9.07%
IRP04N08	3319.10	2548	771.10	5401.92	3463.01	2484	979.01	32.76	4.34%
IRP05N08	3419.35	2214	1205.35	5400.83	3459.77	2414	1045.77	25.50	1.18%

Como se puede apreciar en las tablas anteriores, el algoritmo propuesto desarrolla políticas de reabastecimiento con costos muy cercanos a las desarrolladas por el solver y en algunos casos el costo de inventario es menor en la propuesta del algoritmo que en la propuesta del solver, lo cual puede ser interesante de considerar. Adicionalmente, el algoritmo desarrollado le gana en tiempo de cómputo al solver de manera decisiva pues mientras que el solver requirió en la mayoría de los casos el tiempo máximo permitido, el algoritmo se demoró escasos segundos.

El tiempo de cómputo es importante porque permite facilitar el análisis de escenarios propuestos por el decisor, lo cual no es factible si se desea usar el solver, debido a que requería demasiado tiempo obtener dichos resultados.

Para finalizar, cabe recalcar que el tiempo máximo de cómputo que se estableció en el solver fue con el fin de obtener la solución a estas instancias rápidamente, en consecuencia, no en todos los casos es la solución óptima de las misma, por lo que es normal observar, en algún caso particular, que la solución arrojada por el algoritmo sea menor que la solución arrojada por el solver.

9. Conclusiones

En el presente trabajo se ha descrito la teoría básica de la metaheurística Búsqueda de la Armonía comprobando que la simplicidad de sus ideas permite una fácil implementación computacional. Aunque fue diseñada para resolver problemas de optimización continua, en el presente caso se la ha adaptado para resolver de manera conjunta el Problema de Ruteo de Vehículos con Inventarios a través de un procedimiento de dos etapas obteniendo excelentes resultados en lo que se refiere a velocidad de cómputo y calidad de las soluciones.

Dado que en la literatura científica existen muchas más metaheurísticas unas muy simples de entender como el GRASP y otras muy complejas como la Búsqueda Tabú, es recomendable que tanto investigadores como alumnos de todos los niveles tengan en su repertorio el conocimiento de no sólo uno, sino de varios de estos procedimientos metaheurísticos de tal manera que estén adecuadamente preparados para poder resolver un problema de cualquier índole usando la mejor herramienta posible en cuanto a simplicidad de aplicación, eficiencia computacional y calidad de resultados.

Para trabajos futuros se puede considerar utilizar otros procedimientos heurísticos para la construcción de las soluciones iniciales así como para mejorar la calidad de las rutas generadas. La combinación con otras metaheurísticas puede ser un punto interesante a tratar.

10. Agradecimientos

Este trabajo es producto del desarrollo de la tesis de maestría titulada ***"Optimización del Reabastecimiento de una Red de Cajeros Automáticos con Estimación Difusa de la Demanda"***

presentada por el Ing. Ramiro Saltos Atiencia, becario del Consejo Nacional de Ciencia y Tecnología (CONACYT) a quien se hace llegar un agradecimiento especial por el completo financiamiento otorgado al desarrollo de la presente investigación.

11. Bibliografía

[1] Z. W. Geem, J. H. Kim, and G. V. Loganathan, "A New Heuristic Optimization Algorithm: Harmony Search," *Simulation*, vol. 2, pp. 60-68, 2001.

[2] K. S. Lee and Z. W. Geem, "A New Metaheuristic Algorithm for Continuous Engineering Optimization: Harmony Search Theory and Practice," *Computers Methods in Applied Mechanics and Engineering*, no. 194, pp. 3902-3933, 2005.

[3] M. Mahdavi, M. Fesanghary, and E. Damangir, "An Improved Harmony Search Algorithm for Solving Optimization Problems," *Applied Mathematics and Computation*, no. 188, pp. 1567-1579, 2007.

[4] M. Omran and M. Mahdavi, "Global Best Harmony Search," *Applied Mathematics and Computation*, no. 198, pp. 643-656, 2008.

[5] A. Federgruen and P. Zipkin, "A Combined Vehicle Routing and Inventory Allocation Problem," *Operation Research*, vol. 32, no. 5, pp. 1019-1037, Sep-Oct 1984.

[6] A. Campbell, L. Clarke, A. Kleywegt, and M. Savelsbergh, "The Inventory Routing Problem," Georgia Institute of Technology, Working Paper Mayo 1997.

[7] P. Toth and D. Vigo, *The Vehicle Routing Problem*, Primera ed.: SIAM, 2002.

[8] A. Kleywegt, V. Nori, and M. Savelsbergh, "The Stochastic Inventory Routing Problem with Direct Deliveries," Georgia Institute of Technology, Working Paper Junio 1999.

[9] A. Campbell and M. Savelsbergh, "A Decomposition Approach for the Inventory Routing Problem," *Transportation Science*, vol. 38, no. 4, pp. 488-502, Noviembre 2004.

[10] C. Archetti, L. Bertazzi, A. Hertz, and M. G. Speranza, "A Hybrid Heuristic for an Inventory Routing Problem," Working Paper 2009.

[11] T. Henry, "A New Solution Approach for the Inventory Routing Problem: Using Vehicle Routing Problem Constructive Heuristics," Universidad de Singapore, Singapore, Tesis de Maestría 2005.

[12] R. Martí, "Procedimientos Metaheurísticos en Optimización Combinatoria," Universidad de Valencia, Valencia, Artículo Académico.

[13] J. Gross and J. Yellen, *Handbook of Graph Theory*.: CRC Press, 2003.

[14] Q. Pan, P. N. Suganthan, M. Tasgetiren, and J. J. Liang, "A self-adaptive global best harmony search algorithm for continuous optimization problems," *Applied Mathematics and Computation*, no. 216, pp. 830-848, 2010.

[15] Solomon. (2007, Marzo) VRP Web. [Online]. http://xurl.es/op8ci