ADAPTACIÓN DE LA METAHEURÍSTICA CUCKOO SEARCH PARA LA SOLUCIÓN DE PROBLEMAS DE RUTEO DE VEHÍCULOS CON DIFERENTES RESTRICCIONES

Jorge Luis Anaya Pertuz[1]
Nubia Milena Velasco Rodriguez[2]
Departamento de Ingeniería Industrial, Universidad de los Andes
Bogotá, Colombia

[1]jl.anaya84@uniandes.edu.co
[2]nvelasco@uniandes.edu.co

RESUMEN

Para dar solución a problemas de ruteo de vehículos con diferentes restricciones, en este trabajo se propone una adaptación de la metaheurística Cuckoo Search desarrollada por Yang y Deb en 2009. La propuesta consiste en aplicar la estrategia *Route First- Cluster Second* en los problemas de ruteo con restricciones de capacidad (CVRP), máxima distancia a recorrer (DCVRP) y de retornos (VRPB). Combinando los principios del Cuckoo Search, el *Path Relinking*, con movimientos *2-opt* e inserción aleatoria se soluciona un problema del agente viajero-TSP, para posteriormente cortar las rutas del TSP con el método *Split* de Prins (2004) y una modificación realizada para el VRPB. Con cada problema, el algoritmo fue probado en instancias conocidas de la literatura, obteniendo resultados que demuestran que nuestro método permite hallar soluciones de calidad en poco tiempo computacional y que es de fácil adaptación a distintos problemas del VRP.

ABSTRACT: To solve vehicle routing problems with different constraints, in this paper an adaptation of Cuckoo Search metaheuristic developed by Yang and Deb in 2009 is proposed. Proposal is to implement the Route First-Cluster Second strategy, to routing problems with constraints of capacity (CVRP), maximum distance traveled (DCVRP) and returns (VRPB). Combining the principles of Cuckoo Search, the Path Relinking with 2-opt movements and random insertion, solved the travelling sales problem-TSP, and then split the TSP routes with Prins method (2004) and its modified version to be solved the VRPB. With each problem, the algorithm was tested on known instances of literature, obtaining results that show that our method is able to find quality solutions in a short computational time and is easily adaptable to different VRP problems.

Keywords: Cuckoo Search, Lévy flight, Path relinking, Travelling salesman problem, Capacitated routing problem, Capacitated routing problem with restriction of distance, Vehicle routing problem with backhaul.

1. INTRODUCCIÓN

La familia de problemas de ruteo de vehículos (*Vehicule Routing Problem – VRP*), están dentro del conjunto de problemas más estudiados en investigación de operaciones, debido a su aplicación en un gran número de situaciones de la vida real, como lo son entre otros, la configuración de la cadena de suministros (Schmid, Doerner y Laporte, et, al. 2013) , diseño de circuitos electrónicos (Tokutaka y K. Fujimura, et, al. 1999), diseño de las rutas o movimientos

que debe realizar un robot para recoger una pieza en un sistema de producción (Sipahioglua, Yazicib, Parlaktunab y Gurelc, et, al. 2008) problemas de programación de tareas (Yu, Lin y Chou et, al. 2010) y Chernykh, Kononov y Sevastyanov et, al.2013) entre otros.

Dada la importancia de este problema tanto para la academia como para la industria, se han desarrollado distintos métodos de solución con el objeto de contribuir a la generación de nuevo conocimiento y de construir herramientas que sirvan como fuente de información para la toma de decisiones dentro de las organizaciones. Entre esos métodos podemos encontrar la programación entera mixta, los planos de corte, branch and Bound (Padherg & Rinald, 1987), algoritmos heurísticos de búsqueda local como 2-opt y 3-opt (Lin & Kernighan, 1973), cadenas de Markov (Martin, Otto, & E., 1991) y metaheurísticas como el recocido simulado (Osman, 1993), la búsqueda tabú (Du y He, 2012), algoritmos genéticos (Bae, Hwang, Cho y Goan , 2007), colonia de hormigas (Zhang y Zhang, 2012) entre otros.

De esta manera este estudio pretende aportar a la generación de una nueva herramienta para la solución de este tipo de problemas, mediante la adaptación de la metaheurística Cuckoo Search.

A pesar de que originalmente el Cuckoo Search no fue diseñado para problemas de optimización discreta, en los últimos años se han realizado adaptaciones al mismo para la solución de este tipo de estudios como el de la programación de la producción. Marichelvama, Prabaharanb &. Yang (et, al. 2014) adaptaron el algoritmo para la solución de problemas de flow shop con el objetivo de minimizar el makespan. Lim, Kanagaraj & Ponnambalam (et, al. 2014) implementaron un híbrido entre el Cuckoo Search y un algoritmo genético, para la solución de un problema de secuenciación de operaciones en procesos de mecanizado y perforación.

Raju, Babukarthik y Dhavachelvan (et, al.2013) implementaron un híbrido entre las metaheurísticas colonia de hormigas y el Cuckoo Search para la programación de trabajos, estos autores utilizan el Cuckoo Search para la búsqueda local ya que al tener menos parámetros que la colonia de hormigas hace más eficiente la búsqueda. El objetivo en esta investigación fue minimizar el makespan.

Burnwal y Deb (et, al. 2013) propusieron una adaptación del Cuckoo Search para la optimización del problema de programación de un sistema de manufactura flexible, los autores plantearon una función multiobjetivo que consistió en minimizar el costo de penalización por demoras en el proceso y maximizar el tiempo de utilización de las máquinas. En esta investigación se realizó una leve modificación al operador de búsqueda local basado en la distribución de *Lévy*, para permitir movimientos entre soluciones de tipo discreto a lo largo del espacio de búsqueda. Los resultados obtenidos mostraron mejores resultados que los algoritmos genéticos-GA y enjambre de partículas (PSO) implementados para las instancias probadas.

Por otro lado recientemente, Sur and Shukla (2014) adaptaron la metaheurística para la solución del problema de optimización combinatoria de ruteo de vehículos en grafos basados en redes viales (carreteras), alcanzando excelentes resultados en comparación al algoritmo de colonia de hormigas- ACO y el *Intelligent Water Drop –IWD*.

Del mismo modo Senthilnatha, Dasb, Omkara &. Mani (et, al. 2013) implementaron con éxito el Cuckoo Search para la solución del problema de *clustering*, obteniendo mejores resultados en comparación al GA y el PSO.

Dada la importancia del VRP y los buenos resultados obtenidos por el Cuckoo Search en comparación con los obtenidos GA y PSO en problemas de programación de la producción (Burnwal et, al., 2012) y en general en problemas de optimización (Bakhshpoori, et, al. 2011) citado por Yang y Deb et, al. 2013), se propone implementar el Cuckoo Search para la solución del VRP con distintas restricciones.

El resto de este documento está organizado de la siguiente forma: en la sección dos se describe de manera general las características de los problemas a resolver, TSP, CVRP, DCVRP y VRPB. En la sección tres se introduce al Cuckoo Search y se presenta la propuesta de solución. En la sección cuatro se muestran los resultados computacionales luego de implementar la metaheurística a un conjunto de instancias conocidas de la literatura. Finalmente en la sección cinco se presentan las discusiones y conclusiones.

2. REVISIÓN BIBLIOGRÁFICA

El problema de ruteo de vehículos *(Vehicule routing problem* VRP), consiste en determinar un conjunto de *m* vehículos que deben visitar un conjunto de clientes a un costo mínimo total, tal que cada vehículo programado debe iniciar y finalizar su ruta en el deposito y cada cliente debe ser visitado exactamente una vez y la demanda total debe ser cubierta por un único vehículo que no puede exceder su capacidad (Prins, et. al 2004). Así mismo cada ruta diseñada debe cumplir otro tipo de restricciones dependiendo el problema como son, ventanas de tiempo, precedencia entre nodo, recolección previa a la entrega entre otras.

El problema del Agente viajero (*Travelling Sales Problem – TSP)* es un problema de optimización combinatoria cuyo objetivo es establecer un recorrido al menor costo posible recorriendo una sola vez todos los nodos a través de un grafo. Este es un tipo especial del VRP donde no hay restricción como alguna de las mencionadas anteriormente.

El TSP se define como una red o grafo no dirigido G= (V; E) con un conjunto de nodos V = {0,1….n} y un conjunto de arcos E. Donde el nodo 0 es el deposito y cada uno de los otros nodos i > 0 representan un cliente con una demanda no negativa d_i y cada arco (i; j) tiene un costo C_{ij}=C_{ji} (normalmente asociado a la distancia) conocidos. La Figura 1 representa un tour de una instancia de cuatro nodos (1-4) y un deposito (0) donde cada nodo representa un cliente o una ciudad y cada arista que une cada nodo representa la parte del tour que pasa por dichos nodos.

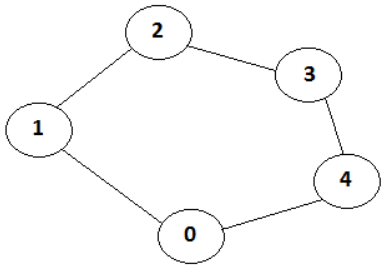


Figura 1. Tour en un TSP de 4 nodos “clientes” y un “deposito”

El problema del agente viajero, se puede clasificar de acuerdo a la distancia calculada entre los nodos (Davendra, et al 2010) en sTSP - problema del agente viajero con distancias entre nodos

simétricas, aTSP - problema del agente viajero con distancia entre nodos asimétricas y el mTSP Múltiple problema del agente viajero, donde m agentes deben visitar n ciudades, una sola vez.

Cuando existe una restricción de capacidad de los vehículos el problema se conoce como el *problema de ruteo de vehículos con restricción de capacidad – CVRP*. Cuando existe la restricción en el tiempo de viaje o de máxima distancia permitida a recorrer en una ruta el problema se conoce como *problema de ruteo de vehículos con restricción de distancia -DVRP*. La combinación entre le CVRP y el DVRP, se conoce como *problema de ruteo de vehículos capacitado con restricción de distancia - DCVRP*. Si los clientes solo pueden ser atendidos en unos intervalos de tiempo establecidos, el problema se conoce como *problema de ruteo de vehículos con ventanas de* tiempo - *VRPTW*.

Cuando en un nodo (predecesor) solo se puede recoger mercancía para luego ser entregada en otro nodo (sucesor) el problema se conoce como *problema de ruteo de vehículos con recolección y entrega –VRPPD*. Cuando existen dos conjuntos de clientes, unos de entrega (*linehaul*) y otros de recolección (*backhaul*) de mercancía, y se debe cumplir que en cada ruta primero se deben realizar todas las entregas de los *linehaul*, y de regreso al deposito se deben realizar todas las recolecciones de los clientes *backhaul* asignados a una ruta, el problema se conoce como *problema de ruteo de vehículos con retorno – VRPB.*

Además de los tipos de problema anteriormente mencionados, existe otras variaciones del VRP (Anbuudayasankar, Ganesh, & Mohapatra, et, al. 2014) que incluyen combinaciones de los mismos, adicional a otras restricciones referentes a situaciones de la vida real, como el diseño de rutas escolares, recolección y distribución de productos alimenticios con una flota de vehículos con capacidad heterogénea, diseño de rutas de distribución con múltiples depositos, diseño de rutas para entregas periódicas, entre otros.

Cada una de las variantes del VRP han sido ampliamente estudiadas por ser consideradas un conjunto de problemas NP-hard (Lawler, Lenstra, Rinnooy, & Shmoys, 1985), es decir que dependiendo del tamaño de las instancias no es posible encontrar una solución óptima en tiempo polinomial, en otras palabras se dice que cualquier implementación computacional emplea un tiempo que crece exponencialmente con el aumento del número de nodos (Mercado et, al, 2000), ya que por ejemplo, en el caso de que se tengan diez ciudades se deberían evaluar 362880 veces el costo. De ahí resulta interesante el desarrollo de algoritmos que reduzcan el número de iteraciones pero que encuentren una muy buena solución.

Adicionalmente a los métodos mencionados en la sección 1, para la solución de estos problemas se han diseñado procedimientos como branch and cut (Baldaccci et al. 2004), branch and cut and price (Fukasawa et al. 2006), programación dinámica, Métodos heurísticos (Laporte & Semet 2002), la heurística de vecino más cercano, la heurística de Clarke and Wright, el método de barrido, el algoritmo Split (Prins, et. Al, 2004), el algoritmo de aprendizaje autónomo (Mohammad et al 2012).

En 2013, Ouaarab, Ahiod & Yang, implementaron un método que denominaron Cuckoo Search discreto mejorado, el cual consiste básicamente en una adaptación del Cuckoo Search básico para la solución del TSP. La diferencia radica en que en esta nueva propuesta, no solo se selecciona una porción de peores soluciones (mayor función objetivo), sino que también se escoge un porcentaje de buenas soluciones (menor función objetivo), con las cuales darán origen a nuevos

nidos a través de movimientos de Lévy, movimientos que se hacen inteligentemente y secuencialmente a partir de un número obtenido con la de la distribución de Lévy adaptada a un problema discreto. Dichos movimientos se hacen a través de intercambios *2-opt* y *doublé bridge*, de acuerdo a unos rangos establecidos para el resultado de la cifra.

2.1. Problema de ruteo de vehículos con restricción de capacidad (CVRP) y Problema de ruteo de vehículos con restricción de capacidad y restricción de máxima distancia a recorrer (DCVRP)

El CVRP se define similar al TSP, solo que en este caso la demanda de cada cliente debe ser atendida por un flota de K vehículos de capacidad Q.

Una solución del CVRP se puede representar como se muestra en la Figura 2, donde se observa un problema en el cual se deben atender ocho clientes con demanda de dos unidades cada uno y vehículos de capacidad ocho unidades. En esta solución cada ruta debe ser atendida por un único vehículo.

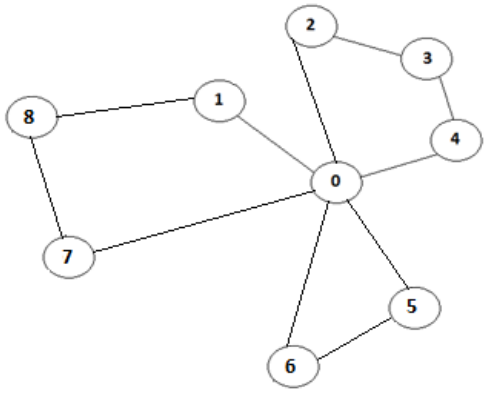


Figura 2. Esquema de rutas para la solución de un CVRP de ocho nodos "clientes" y un "deposito"

Por su parte el DCVRP se puede describir como una extensión del CVRP donde hay una restricción adicional, que consiste en que cada ruta no puede exceder una longitud máxima de recorrido permitida, o el equivalente a un máximo tiempo de duración del recorrido. A la representación del CVRP, se le debe incluir el parámetro *maxL*, que corresponde a la máxima distancia a recorrer en una ruta y S_i que se refiere al tiempo de servicio empleado para atender al cliente ($i \neq 0$). Cuando $S_i \neq 0$, el costo de recorrer un arco (distancia/tiempo) es $C_{ij} = t_{ij} + S_i/2 + S_j/2$, donde t_{ij} es el tiempo o longitud original del arco (i, j).

Los métodos empleados para la solución de estos problemas, entre otros corresponden a los mencionados en las secciones uno y dos, basados en las estrategias de *Cluster First - Route Second (CFRS)* o *Route First – Cluster Second (RFCS).* La primera estrategia consiste en agrupar clientes en cada ruta que será servida por un vehículo y luego para cada ruta se resuelve el TSP. Contrario a la anterior estrategia, el *RF-SC*, se refiere a que primero se soluciona un TSP, para finalmente cortar óptimamente dicho tour en rutas factibles para el CVRP o DCVRP, empleando procedimientos como el Split (Prins, et, al. 2004).

2.2. Problema de ruteo de vehículos con retornos (VRPB)

El *Problema de ruteo de vehículos con retornos* VRPB es una extensión del CVRP, en la cual hay dos tipos de clientes a servir. El primer conjunto *(linehaul)* está conformado por aquellos clientes a los que hay que entregar una cantidad q_i *(*d_i*)* de producto desde el deposito. El segundo conjunto *(backhaul)* corresponde a los clientes a los cuales se debe recoger una cantidad p_i *(*d_i*)* y ser llevada hasta el deposito. En este problema se deben cumplir las siguientes condiciones (Toth & Vigo, et, al. 1999): (1) cada ruta debe salir y regresar a un único deposito y debe ser servida por un único vehículo; (2) para cada ruta la carga total asociada a los clientes *linehaul* y *backhaul* no debe exceder por separado la capacidad del vehículo; (3) no pueden haber rutas donde solo se visiten clientes *backhaul*, y si en alguna ruta se visita un cliente *backhaul* este no puede ser visitado antes que se hagan todas las entregas a los clientes *linehaul* de la ruta; (4) el objetivo es minimizar la distancia total recorrida.

El VRPB se puede representar (Toth, et al 1999) con un grafo no dirigido G = (V, E) donde V = {0} ∪ *L* ∪ *B*, el deposito es {0}, *L*= {1, 2, 3……, n} corresponde al conjunto de clientes *linehaul*, y *B*= {n+1……, n+m}, es el conjunto de clientes *backhaul*. Cada uno de los clientes del conjunto *L*, tiene una demanda de entrega de q_i *(*d_i*)* unidades. Cada uno de los clientes del conjunto *B*, tiene una demanda de recolección p_i *(*d_i*)* unidades. Todos los clientes deben ser atendidos por una flota de K vehículos de capacidad Q, teniendo en cuenta que en una ruta, $\sum_{i\in L} d_i \leq Q$ y $\sum_{i\in B} d_i \leq Q$, por separado en el caso de que se visiten ambos tipos de cliente en la misma ruta. Por su parte E = {(i, j): i ≠ j}, corresponde al conjunto de arcos. Asociado a cada arco existe un costo C_{ij}, que corresponde a la distancia de ir desde i hasta j. En la Figura 3, se presenta un conjunto de rutas factibles para un problema de VRPB con ocho clientes *linehaul* y cuatro clientes *backhaul*. Cada uno de los clientes sin importar el tipo, tiene demanda de dos unidades y el vehículo tiene capacidad de seis unidades.

Entre los métodos más conocidos para la solución de este problema se encuentra el denotado como DB propuesto por Deif & Bodin (1984), que es una variante del algoritmo de Clarke & Wright. Dicho algoritmo consiste en ir consolidando rutas de acuerdo a los mayores ahorros entre arcos en términos de la distancia (costo) entre cada par de nodos (excepto con el deposito). Para evitar la formación de rutas con clientes *backhaul*, Deif y& Bodin propusieron establecer un costo de penalización asociado al máximo ahorro estimado, para aquellos arcos de clientes de diferentes tipos. Otro de los métodos es el denominado SF propuesto por Goetschalckx & Blecha (1989). Este método es fundamentado en el concepto de *spacefilling curves*, este método consiste en formar rutas por separado de clientes *linehaul y backhaul*, para luego fusionar cada ruta de clientes *linehaul* con una ruta de *backhaul* de acuerdo al mapeo del *spacefilling*.

En el año 1993, Goetschalckx & Blecha desarrollaron la heurística LHBH para el caso de distancias simétricas del VRPB. El algoritmo primero resuelve un problema de asignación y localización capacitado. Posteriormente se conforman rutas con respecto a la menor distancia entre los clientes *linehaul* al deposito y para los clientes *backhaul* de los más lejanos a los más cercanos. Seguidamente se forman grupos de clientes resolviendo un problema de asignación generalizado, de acuerdo a un costo de inserción de cada cliente a una ruta. Finalmente se implementa una heurística de inserción que tiene en cuenta la realización de movimientos factibles en la ruta.

Toth, et, al 1999, propuso una heurística basada en la estrategia *Cluster First – Route Second*. El algoritmo consiste en formar grupos (clúster) solucionando la relajación lagrangiana del problema, sin importar el caso de infactibilidad. Posteriormente para cada clúster se soluciona un

TSP, en este caso teniendo en cuenta solo movimientos factibles (relación de precedencia). Para hacer mejoras en cada ruta se procede a realizar movimientos factibles *2-opt* y *3-opt*. Finalmente entre cada una de las rutas se evalúan intercambios factibles de arcos, que respeten las relaciones de precedencia entre el conjunto de clientes.

Palhazi, Goos, Sörensen & Arráiz et, al. 2014, proponen una búsqueda local iterativa (*ILS)* fundamentado en dos características principales. La primera consiste en explorar un amplio vecindario, garantizando la eficiencia de la exploración a través del almacenamiento de información del conjunto de soluciones a lo largo de las iteraciones. La segunda consiste en el movimiento entre soluciones factibles e infactibles, regulando estos movimientos con un costo de penalización asociado a las soluciones infactibles.

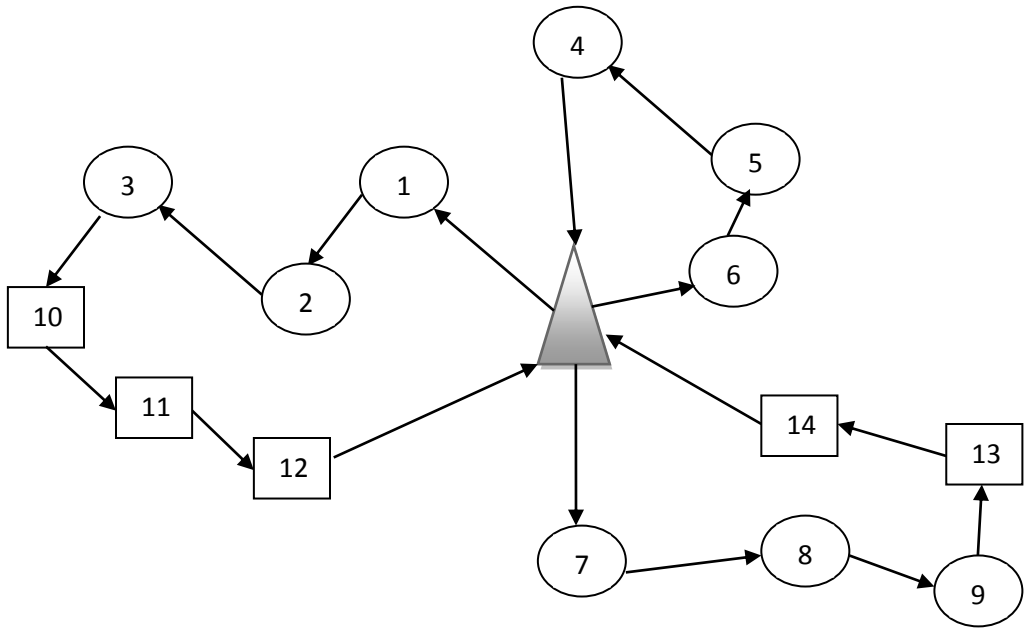


Figura 3. Esquema de rutas para la solución de un VRPB de ocho nodos *linehaul* ◯, cuatro nodos *backhaul* □ y un "deposito"

Zachariadis &. Kiranoudis, et, al. 2012, implementaron una búsqueda local intensiva que es regulada a través de sucesiones de Fibonacci, que permiten dar prioridad especial a algunas soluciones, aumentando la velocidad en la búsqueda de mejores soluciones. Así mismo, con el fin de evitar caer en ciclos y aumentar la diversificación del espacio de soluciones, los autores usan un parámetro similar al de una lista Tabú que evita evaluar soluciones ya exploradas o con atributos similares.

3. DESCRIPCIÓN DEL CUCKOO SEARCH

En este documento se propone la implementación del algoritmo Cuckoo Search desarrollado por Yang y Deb en 2009, quienes lo diseñaron basándose en tres reglas inspiradas en el comportamiento reproductivo de las aves Cuckoo:

1. Cada Cuckoo pone un huevo a la vez, y lo deja en un nido (solución) escogido aleatoriamente.

2. El mejor nido, es decir el que posee los huevos con mejor calidad (soluciones) será el que de paso a las nuevas generaciones de Cuckoo.

3. El número de nidos disponibles es fijo, y un pájaro anfitrión puede descubrir un huevo alíen con una probabilidad Ps ∈ [0, 1]. En este caso el ave puede arrojar el huevo extraño fuera del nido, o abandonar el nido.

Por simplicidad, este último supuesto corresponde a reemplazar una proporción Ps de los peores nidos por unos nuevos (nuevas soluciones aleatorias).

Basados en estas tres reglas básicas, los pasos lógicos del CS son resumidos en el seudocódigo presentado en la Figura 4, para un problema de optimización continua de maximización.

CUCKOO SEARCH BÁSICO

Función objetivo $f(x),\ x = (x_1, \ldots\ldots, x_d)^T$;
Generar población inicial de *n* nidos x_i $(i = 1,2, \ldots\ldots, n)$;
Mientras (*t* < MaxGeneraciones) ó (criterio de parada);
Obtener un Cuckoo aleatoriamente con Lévy flight *(denotarlo como i)*;
Calcular la función objetivo del cuckoo F_i;
Seleccionar un nido aleatoriamente de los *n* nidos *(denotarlo como j)*;
Si $(F_i > F_j)$,
Reemplazar *j* con la nueva solución *i*;

Fin si
Abandonar una fracción (Ps) de los peores nidos
[y construir nuevas soluciones con Lévy flight];
Mantener las mejores soluciones
Organizar de mejor a peor solución y almacenar la mejor solución actual;
Fin mientras
Reportar mejor nido (solución) encontrado

Figura 4. Seudocódigo del Cuckoo Search (Ouaarab et.al, 2013).

Cuando se están generando nuevas soluciones, o Cuckoo se lleva a cabo un vuelo de Lévy. Estos vuelos se realizan por medio de la distribución de Lévy descrita en la Ecuación 1, la cual genera una caminata aleatoria, que es definida en la ecuación 2.

$$\text{Lévy } \sim s^{-(1+\gamma)},\ (1 < \gamma \leq 2) \quad \textbf{(1)}$$

$$x_i^{t+1} = x_i^t + \alpha L(s, \gamma) \quad \textbf{(2)}$$

Dónde:

$$L(s, \gamma) = \frac{\gamma\Gamma(\gamma) sen\left(\frac{\pi\gamma}{2}\right)}{\pi} \frac{1}{s^{1+\gamma}}, \quad (s \gg s_0 > 0) \quad \textbf{(3)}$$

Adicionalmente el parámetro α o tamaño de paso debe estar enlazado con los límites o cotas (cota superior U_b y cota inferior L_b) de la variable así:

$$\alpha = 0.01(U_b - L_b) \quad \textbf{(4)}$$

3.1. Propuesta de Implementación Cuckoo Search (CS) para el TSP

A continuación se describe la estructura desarrollada para la implementación del CS para la solución del TSP:

1. Un nido y un Cuckoo son equivalentes (una solución), y se representa por un vector de tamaño $(n+1)$, en donde n se refiere al número de nodos del problema y la posición adicional indica que se debe terminar en el depósito.

2. Generar aleatoriamente la población de los *m* nidos iniciales. A partir de este conjunto de nidos realizar movimientos *2-opt*. en busca de mejores soluciones (disminución de la FO).

En el caso del VRPB dada sus características, se debe generar una solución inicial aleatoria de un TSP (Tour gigante), seleccionando aleatoriamente un cliente *linehaul*, para ser el primero en visitar en la ruta. Los demás nodos se seleccionan aleatoriamente de todo el conjunto *V* de clientes. Del mismo modo cuando se efectúen intercambios *2-opt* no se puede conectar justo después del deposito un nodo *backhaul.*

3. Seleccionar una porción Pa de los mejores nidos (soluciones) (Ouaarab, et, al. 2013) y dar paso a la generación de nuevas soluciones, implementado un *Path Relinking* como estrategia de intensificación y diversificación.

4. Generar aleatoriamente un nuevo nido. Perturbar el nuevo nido con Lévy Flight. Seleccionar aleatoriamente un nido dentro de la población y comparar su solución con la del nuevo nido. Si el valor de la función objetivo del nuevo nido es menor que la del nido de la población, este último es reemplazado.

5. Reemplazar la proporción (*m**Ps) de nidos con peores función objetivo, por nuevos nidos generados aleatoriamente y perturbados a través del método de Lévy Flight.

6. Organizar los nidos de menor a mayor función objetivo.

7. Repetir los pasos 3 al 6, hasta que el criterio de parada sea alcanzado (Número máximo de iteraciones). Reportar mejor solución.

Nota: En el VRPB cuando se efectúen los pasos 3 y 4, solo se podrán insertar clientes linehaul justo después del deposito.

3.1.2 Path relinking como estrategia de intensificación y diversificación (paso 3)

Como estrategia de intensificación y diversificación se implementó el *Path relinking*, de acuerdo a lo propuesto por Laguna, Martí et. al, 1999. Este método consiste en mantener un conjunto de soluciones élites- ***ES*** (mejores soluciones). Durante cada iteración con el propósito de obtener mejoras locales en las soluciones, se escoge un nido aleatoriamente entre la proporción Pa de los mejores nidos, y se selecciona aleatoriamente una solución dentro del conjunto ***ES*.** Luego con el par de soluciones se procede a comparar cada una con respecto a la otra y se calcula la diferencia (distancia) que hay entre los arcos de cada una. Finalmente se generan nuevas soluciones y realizan movimientos (reparaciones) para generar nuevas soluciones, a partir de los intercambios de arco en una solución, con el fin de llevar a que una solución sea muy similar a la otra, reduciendo la diferencia entre ambas.

La solución que se va a transformar con respecto a la otra, se denomina *solución inicial* - $S_{0,}$ y la solución que sirve de "referencia" u objetivo, para los movimientos de $S_{0,}$ se conoce como *solución guía*- $S_{f.}$ Los criterios que se describen a continuación, determinan cuál de las soluciones será la inicial y cuál la guía (Resende & Ribeiro, et. al., 2003):

- forward: la peor solución entre ambas, es la inicial y la otra la guía ;
- backward: la mejor entre ambas, es el origen y la otra es la guía.
- back and forward: en este caso se evalúan dos trayectorias. En la primera la mejor entre las dos es la solución inicial y en la segunda, la peor es la solución inicial.
- mixed: se exploran simultáneamente dos pasos, el primero inicia en la mejor solución y el segundo inicia en la peor entre ambas, esto se hace que se encuentra una solución equidistante a $S_{0\,y}\,S_f$.

La idea fundamental es que las mejores soluciones comparten entre sí, algunas características en la estructura de su vecindario (orden en que se visitan los nodos) por lo cual al moverse de una solución a la otra se pueden encontrar mejores soluciones que las actuales. Dado que el Cuckoo Search permite la generación de un considerable número de soluciones es idóneo para la aplicación del PR iterativamente.

En esta investigación se implementó la estrategia *back and forward*, para la búsqueda de mejores soluciones. La Figura 5, ilustra en qué consiste cada uno de estos movimientos. La primera figura corresponde a los movimientos forward. En la primera fila se presenta la solución inicial S_0 y en la última se evidencia la solución guía S_f. Las rutas T_1 a T_4 corresponden a cada uno de los tures generados luego de cada movimiento realizado para transformar S_0 *en* $S_{f..}$

La última columna especifica el número de arcos rotos (diferencias) de S_0 con respecto a $S_{f.}$ Dichas diferencias corresponden a seis ya que los arcos (0,1), (2,3), (4,5), (5,6), (6,7) y (7,0) no están presentes en $S_{0.}$

La solución T_1 de la fila dos de la tabla se obtiene, al reparar el arco (1,2), el cual se inserta en las posiciones uno y dos de la ruta, desplazando dos posiciones a cada uno de los nodos subsecuentes de la ruta. Este mismo procedimiento se realiza para obtener la solución T_2, ya que a partir de T_1 se retira de la posición actual el arco (3,4) y se inserta en las posiciones tres y cuatro, desplazando dos posiciones a cada uno de los nodos subsecuentes de la ruta.

A pesar de que en la figura se ilustran todos los movimientos que se llevan a cabo para llevar S_0 a S_f. el procedimiento se debe repetir hasta que la diferencia entre las rutas sea mayor o igual a uno, con el objeto de no duplicar soluciones.

Para el caso de los movimientos *backward* se realizan los mismos pasos anteriormente descritos, la única diferencia es que ahora se reparará S_f con respecto a $S_{0.}$

Movimientos Forward

	Tour gigante	Distancia a T(S_f)
T(S_0)	0 7 6 5 3 4 1 2 0	6
T_1	0 1 2 7 6 5 3 4 0	5
T_2	0 1 2 3 4 7 6 5 0	4
T_3	0 1 2 3 4 5 7 6 0	3
T_4	0 1 2 3 4 5 6 7 0	0
T(S_f)	0 1 2 3 4 5 6 7 0	

Movimientos Backward

	Tour gigante	Distancia a T(S_0)
T(S_f)	0 1 2 3 4 5 6 7 0	6
T_1	0 7 1 2 3 4 5 6 0	5
T_2	0 7 6 1 2 3 4 5 0	4
T_3	0 7 6 5 1 2 3 4 0	3
T_4	0 7 6 5 3 4 1 2 0	0
T(S_0)	0 7 6 5 3 4 1 2 0	

Figura 5. Ejemplo de funcionamiento del Path relinking (Villegas et.al, 2011).

Posteriormente cada una de las soluciones generadas es mejorada con alguna estrategia de búsqueda local. En el caso de esta investigación cada una de las soluciones fue mejorada a través de intercambios *2-opt*.

Seguidamente, una a una de las soluciones anteriores, es evaluada como candidata a ingresar al conjunto ***ES,*** de acuerdo a las siguientes condiciones (Villegas et.al, 2011):

- Si la función objetivo de la solución del Path relinking (*S*) es menor que la peor solución del conjunto y la distancia *d(**ES,** S)* entre *S* y cada solución del conjunto debe ser mayor a un parámetro Δ (umbral). Para esta investigación el valor de este parámetro se calculó como una proporción del número de nodos.

  Finalmente *S* reemplaza a la solución *S'* del conjunto tal que: min {*d(**S',** S)*}, es decir por la solución con la que se tenga menor distancia (diferencias).

- Cuando la función objetivo de la nueva solución *(S)* es menor que la mejor del conjunto ***ES***, no se evalúa la condición de cumplimiento del umbral y automáticamente entra al conjunto, pero reemplazando a la solución *S'* del conjunto tal que: min {*d(**S',** S)* }, es decir por la solución con la que se tenga menor distancia (diferencias).

3.1.3 Implementación de Lévy Flight (pasos 4 y 5)

El propósito de la implementación de *Lévy* Flight, es también favorecer la diversificación y la intensificación de las soluciones, mediante el movimiento de una región a otra dentro del espacio de solución, para evitar quedar atrapados en un óptimo local (Ouaarab, et, al. 2013). Para realizar los movimientos en el espacio de solución, se debe determinar la longitud de paso de acuerdo a un número generado con la distribución de *Lévy*. Para nuestro problema, nos basaremos en la propuesta hecha por Burnwal et al. 2013. A continuación se describe con un ejemplo:

I. Generar un número aleatorio a través de la distribución de Lévy, así:

$$Número\ de\ Levy = \propto * S^{-(\gamma+1)} \quad \textbf{(5)}$$

Considere: $\propto = 10^9; S = 15;\ \gamma = 1$

$$Número\ de\ Levy = \propto * S^{-(\gamma+1)} = 4{,}444{,}444$$

Donde, α → *Factor de escalada de la longitud de paso*

γ → *Varianza de la distribución de Lévy*

S → *Tamaño de paso. Que corresponde al total de generaciones (iteraciones) realizadas hasta el momento.*

II. Descomponer la parte entera del número generado en cifras de acuerdo a las necesidades del problema y estimar el número de cifras del número (largo). Este paso determinará el número de intercambios a realizar.

Cifra de Lévy → 4 4 4

Largo de Lévy (Lv) = 3

III. A partir de una solución actual o una generada aleatoriamente, realizar perturbaciones (movimientos) a la solución de acuerdo a la cifra de *Lévy*, generada en el paso I.

Considere la siguiente permutación[1]:

0 – 1 – 2 – 3 – 4 – 5 – 6 – 7 – 8 – 9 – 10 – 0

IV. Aleatoriamente decidir, si se realiza la caminata hacía adelante o atrás (desde la primera posición a la última o viceversa). De acuerdo a lo anterior seleccionar aleatoriamente una posición *k* de la permutación [2, n – Lv] ó [n - Lv, n].

*0 – **1** – 2 – 3 – 4 – 5 – 6 – 7 – 8 – 9 – 10 – 0*

. ↑

V. Sumar la cifra de Lévy con el ID del trabajo en la posición k+1, k+2,…k+Lv o hacia k-1, k-2,…k - Lv:

4 4 4 ← *Cifras de Levy*

*0 – **1 – 2 – 3** – 4 – 5 – 6 – 7 – 8 – 9 – 10 – 0* ← *Permutación*

*0 – **5 – 6 – 7** – 4 – 5 – 6 – 7 – 8 – 9 – 10 – 0* ← *Resultado de la adición*

Lv=3

VI. Hacer los cambios de posición de los nodos, teniendo en cuenta que los resultados de la adición del número de Lévy no sea superior al número total de nodos, así:

*0 – **5 – 6 – 7** – 4 – **1 – 2 – 3** – 8 – 9 – 10 – 0* ← *Nueva secuencia*

En este estudio se propone también, realizar inserción de subsecuencia seleccionadas aleatoriamente, a una posición escogida aleatoriamente. El tamaño de la subsecuencia y el máximo número de inserciones está limitado al largo del número de *Lévy* generado. El propósito

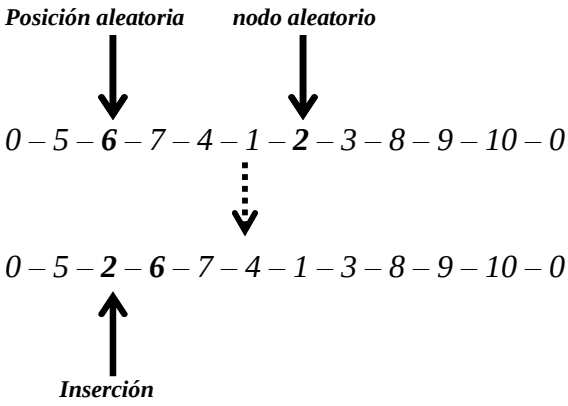
[1] Cero corresponde al deposito

de estos movimientos es generar saltos entre un espacio de solución y otro, en busca de mejores soluciones y evitar quedar atrapados en óptimos locales.

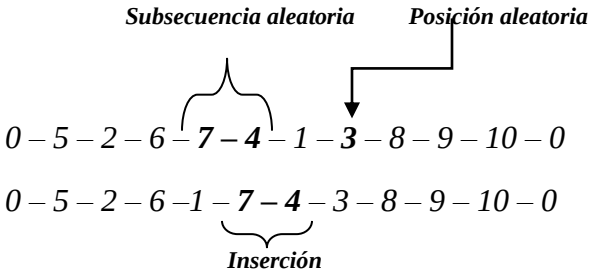
El procedimiento planteado se describe a continuación:

Para nuestro ejemplo como *Lv = 3, luego* 3 será el máximo tamaño de la subsecuencia y el máximo número de inserciones.

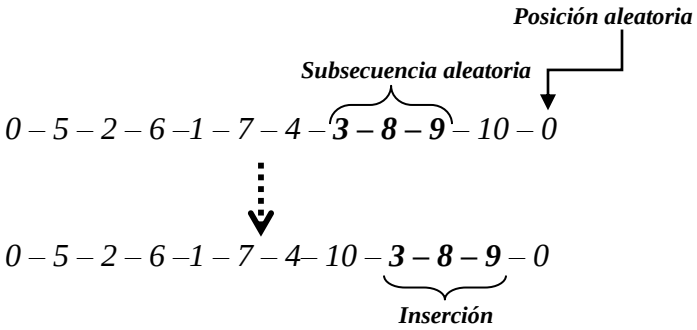
VII. Se selecciona aleatoriamente un nodo y se inserta aleatoriamente en una posición de la permutación:



VIII. Se selecciona aleatoriamente otra subsecuencia (2 nodos) y se inserta aleatoriamente en una posición:



IX. Se selecciona aleatoriamente otra subsecuencia (3 nodos) y se inserta aleatoriamente en una posición:



X. Finalmente se realizan movimientos *2-opt* a la permutación.

3.2. Implementación del Cuckoo Search para el CVRP, DCVRP y el VRPB

Para estos tres problemas se utilizará la estrategia *Route First – Cluster Second.* El objetivo es crear un conjunto de soluciones iniciales (nidos), que serán mejoradas conforme a movimientos que den solución a un TSP. Cada uno de estos nidos se cortan óptimamente con el método Split propuesto por Prins et. al, 2004 y una variante del mismo que se adaptó a las restricciones del VRPB. Se seleccionó esta estrategia, teniendo en cuenta el buen desempeño que tuvo la metaheurística en la solución del TSP (Ouaarab et, al. 2013) y de los excelentes resultados y amplia aplicación que se ha demostrado, del uso del método Split para este tipo de problemas y sus variantes (Prins, Lacomme & Prodhon, et, al. 2014).

Cabe anotar que luego de hacer el Split es posible realizar perturbaciones a cada una de las rutas obtenidas, si el costo de la solución mejora con los movimientos, se debe formar nuevamente una ruta para el TSP y posteriormente aplicar el Split. En consecuencia el costo de la solución para el CVRP, DCVRP y el VRPB debe ser menor o igual al costo de la solución obtenida luego de las perturbaciones intra-rutas. Pero dadas las características del VRPB cuando se realicen las perturbaciones dentro de cada ruta de la solución, solo se permitirán los siguientes intercambios:

- Entre arcos de clientes *linehaul.*
- Entre arcos de clientes *backhaul.*
- Entre un arco de clientes *linehaul* y el arco entre el último cliente *linehaul* y el primer *backhaul.*

La Figura 6 muestra el seudocódigo de nuestra propuesta de adaptación del Cuckoo Search, al que denominaremos *Cuckoo Search Intensivo.* Dicho seudocódigo incluye en el paso 37 la realización del *Split* en dado caso se requiera resolver un CVRP, DCVRP o un VRPB como se mencionó anteriormente. Para el caso del TSP se omite este paso.

3.2.1 Método Split para el VRPB[2]

Basados en el algoritmo desarrollado por Prins, et, al. 2004, se propone una modificación al mismo para adaptarlo a las características del VRPB. Este método opera sobre un grafo auxiliar dirigido y acíclico *G´ (N´, A´)*, con n+1 nodos en el cual se encuentra la ruta más corta para recorrer un arco:

- Cada nodo *k* representa el cliente que se encuentra en la *k-ésima* posición de la permutación *S.*

- El arco (i, j) representa la ruta que atiende los clientes desde la posición S_{i+1} hasta S_j (en la permutación).

[2] El método a aplicar en el CVRP y DCVRP es similar al que se describe para el VRPB, la diferencia radica en el cumplimiento de las restricciones propias del problema que se describirán en detalle.

- El costo de cada arco del grafo auxiliar Z_{ij} se calcula como:

$$\boldsymbol{Z_{ij} = C_{0,S_{i+1}} + \sum_{k=i+1}^{j-1} C_{S_k,S_{k+1}} + C_{S_j 0}} \qquad (\mathbf{6})$$

Esta función de costos es posible calcular siempre y cuando el arco exista, es decir se cumpla[3]:

1. $\boldsymbol{S_{i+1} \in L}$

2. $\boldsymbol{S_k \wedge S_{k+1} \in L \quad \vee \quad S_k \in L \quad \wedge \quad S_{k+1} \in B \quad \vee \quad S_k \wedge S_{k+1} \in B}$ en otras palabras $\boldsymbol{S_k \notin B \quad \wedge \quad S_{k+1} \notin L}$

- Del mismo modo un arco también existe si cumple con las siguientes condiciones[4]:

$$\boldsymbol{\sum_{k=i+1}^{j} d_{S_k} \leq Q \quad | S_k \in L} \qquad (\mathbf{7})$$

$$\boldsymbol{\sum_{k=i+1}^{j} d_{S_k} \leq Q \quad | S_k \in B} \qquad (\mathbf{8})$$

Para ilustrar el planteamiento anterior, suponga que se tiene la permutación 01234560, donde 4 y 5 son clientes *backhaul*. Cada ruta que se genere utilizará un vehículo diferente de capacidad 10 unidades. En la Figura 7 se muestran las distancias entre cada par de puntos de acuerdo a la configuración de la ruta, y la demanda de cada cliente.

[3] En el CVRP y DCVRP, no se debe cumplir ninguna de estas condiciones. Pero en el caso especifico del DCVRP se debe cumplir que $Z_{ij} \leq maxL$

[4] En el CVRP y DCVRP solo se debe cumplir la ecuación 7, ya que en esencia en estos problemas todos los clientes son de entrega (*Linehaul)*

CUCKOO SEARCH INTENSIVO PARA VRP

IT → *Máximo número de iteraciones;*
n → *nodos*;
m → *nidos*;
Ps → *Porcentaje de selección de buenos nidos* (mejores soluciones)*;*
Pa → *Porcentaje de nidos abandonados* (peores soluciones)*;*
$\propto$ → *Factor de escalada de la longitud de paso;*
γ → *Varianza de la distribución de Lévy* $1 < \gamma \leq 2$*;*
$|ES|$ → *Tamaño del conjunto de soluciones elite;*
Δ → *Umbral para permitir que una solución ingrese al conjunto ES*

1. Generar soluciones iniciales (nidos) p_i (i = 1, 2, …..,m);
2. Calcular FO(p_i) (i = 1, 2, …..,m);
3. **para** i = 1 hasta $|\boldsymbol{ES}|$;
4. *S: = Generar una permutación*;
5. $S \in \boldsymbol{ES}$;
6. **Fin para**
7. **Mientras (**t <*IT* **)** ó (Criterio de parada) ;
8. Ordenar las soluciones (nidos) de mejor a peor FO ;
9. Seleccionar aleatoriamente un p_i *entre* los *Pa* nidos (Excepto la mejor) ;
10. Seleccionar aleatoriamente $S' \in ES$;
11. *PR: = PathRelinking* (p_i, *S'*);
12. **para todo** $\bar{S} \in PR$ ***hacer*** ;
13. $\bar{S} := \ 2opt\ (\bar{S})$;
14. Calcular FO($\bar{S}$) ;
15. **si** d($\boldsymbol{ES}, \bar{S}$) $\geq \Delta$ ***luego*** ;
16. retirar *S'* de $\boldsymbol{ES}$ | min d$\left(\boldsymbol{S'}, \bar{S}\right)$;
17. insertar $\bar{S}$ en $\boldsymbol{ES}$;
18. **si** FO($\bar{S}$) < FO(p_i) ***luego*** ;
19. p_i : = $\bar{S}$;
20. FO(p_i) = FO($\bar{S}$)
21. **fin si** ;
22. **fin si** ;
23. **fin para** ;
24. Generar una permutación S_w ;
25. Modificar S_w con *Lévy Flight*;
26. Calcular FO(S_w)
27. Seleccionar aleatoriamente un p_i *entre* los m *nidos*
28. **si** FO(S_w) < FO(p_i) ***luego***;
29. p_i : = S_w ;
30. FO(p_i) = FO(S_w) ;
31. **fin sí** ;
32. Ordenar las soluciones (nidos) de mejor a peor FO ;
33. Abandonar la fracción *Pa* de peores nidos ;
34. Reemplazar la fracción *Pa* por nuevas soluciones obtenidas con *Lévy Flight*
35. **fin Mientras**
36. **para** i = 1 hasta m
37. *Split* (p_i)
38. **fin para**
39. Reportar la mejor solución

Figura 6. Seudocódigo del Cuckoo Search Intensivo

En la Figura 8, se muestra el grafo auxiliar resultante para esta permutación:

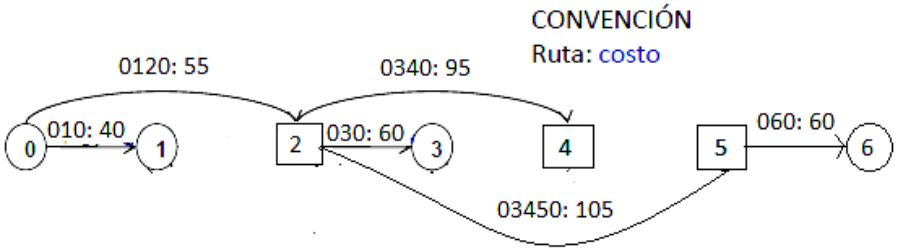


Figura 8. Grafo auxiliar para la permutación 01234560. (Basado en Prins, et, al. 2004)

Simultáneamente se calcula el costo de cada arco se evalúa su factibilidad de acuerdo a la ecuación 6 y sus condiciones (1 y 2). Siempre y cuando el costo del arco Z_{ij} se pueda calcular por cumplimiento de las condiciones, se procede a evaluar la factibilidad del arco de acuerdo a las ecuaciones 7 y 8.

Por ejemplo el arco (0,2) es factible dado que S_1 igual a 1, es un cliente *linehaul*, y S_2 igual a 2, también es un cliente *backhaul*.

El costo del arco es igual a 55, ya que:

$$\boldsymbol{Z_{02} = C_{0,S_1} + C_{S_1,S_2} + C_{S_20} = 20 + 10 + 25 = 55}$$

Por otro lado la factibilidad se cumple por que la suma de las demandas de los clientes en la ruta (en este caso solo *linehaul)* es igual a 9.

El análisis del arco (2,5) es el siguiente:

$$\boldsymbol{Z_{25} = C_{0,S_3} + C_{S_3,S_4} + C_{S_4,\,S_5} + C_{S_5,0} = 30 + 25 + 15 + 35 = 105}$$

Lo anterior es posible, dado el cumplimiento de las condiciones de factibilidad:

$$\boldsymbol{S_3 \in L \quad ^{\wedge} \quad S_4 \in B \quad y \quad S_4 \,^{\wedge}\, S_5 \in B}$$

De acuerdo a las ecuaciones 7 y 8, también se cumple la factibilidad, dado que la demanda del único cliente *linehaul* (3) en la ruta es 4 y por otro lado la suma de las demandas de los clientes *backhaul* (4 y 5) es 9. Por lo tanto en ambos casos no se supera la capacidad de 10 unidades a transportar por el camión.

Observando el grafo de la Figura 8, podemos deducir que los arcos (1,2), (1,3), (1,4), (1,5), (1,6), (2,6) (3,4), (3,5), (3,6), (4,5) (4,6) no son factibles debido a que estos implican la creación de rutas, en las cuales el primer cliente que se visitaría sería un *backhaul* o se visitaría un *lineahul* posterior a un *backhaul*.

Para extraer del grafo auxiliar, el conjunto óptimo de rutas para esa permutación, se debe primero establecer cuáles son los predecesores de cada cliente, empleando el algoritmo de *Bellman –Ford* (Prins, et, al. 2004).

La Tabla I, resume los resultados luego de aplicar el algoritmo de *Bellman – Ford* al grafo auxiliar de la Figura 8:

Tabla I. Resultados del algoritmo de *Bellman – Ford*, para el ejemplo de VRPB

nodo	V_i	P_j
0	0	-
1	40	0
2	55	0
3	115	2
4	150	2
5	160	2
6	220	5

Finalmente para construir las rutas se implementa el algoritmo de extracción descrito en Prins, et, al. 2004, el cual inicializa con una lista vacía de viajes, y conforma cada ruta almacenando en un vector el orden en que se visita cada cliente, de acuerdo a sus predecesores. Por lo cual se inicia el algoritmo partiendo desde el último nodo de la permutación hasta que se llega al deposito (nodo 0).

Aplicando el algoritmo a los resultados presentados en la Tabla I, se obtiene el conjunto de rutas que se muestra en la Figura 9, que cumplen con las restricciones de precedencia y demanda del VRPB.

El costo total de la solución es de 220, que corresponde a la suma de la distancia total recorrida en cada ruta.

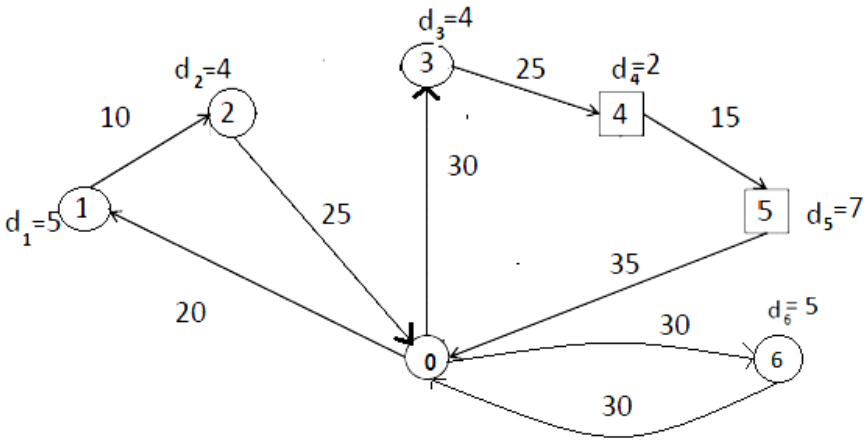


Figura 9. Solución del ejemplo de VRPB, para una permutación cortada óptimamente con el método *Split*

4. EXPERIMENTOS COMPUTACIONALES

Para hacer las corridas de prueba del algoritmo este se codificó en Java/Eclipse SDK 3.5.2. Los experimentos se corrieron en una computadora DELL INSPIRON 1420 INTEL(R) CORE™ 2Duo T5750 de 2.00 GHz y 2.00 GB de RAM y validado con

instancias conocidas de la literatura disponibles en *TSPLIB, http://www.or.deis.unibo.it/reSearch_pages/ORinstances/VRPLIB/VRPLIB.html y en* http://www.branchandcut.org/.

Con el propósito de evaluar la combinación de parámetros que arroja los mejores resultados al problema estudiado con la metaheurística propuesta, se realizó un diseño factorial 2^k donde los parámetros a comparar fueron el número *m* de nidos, el número de generaciones (iteraciones), la proporción Pa de mejores nidos, la proporción Ps de peores nidos, el tamaño del conjunto *ES*, el porcentaje (delta) para calcular el umbral del PR, y el valor del parámetro γ de la distribución de *Lévy*.

Las variables de respuesta a analizar bajo esta combinación de parámetros fueron el Gap (%) de la función objetivo y el tiempo computacional empleado para hallar una solución. La determinación de los valores mínimos y máximos para cada parámetro se estableció de la siguiente manera:

1. Para *m*, Pa y Ps, se tuvo en cuenta las recomendaciones de Ouaarab, et, al. 2013, que sugiere que se debe escoger *m* en el rango [25,40], la proporción Pa debe estar en el rango [0.2, 0.6] y la proporción Ps en el rango [0.25, 0.5]. El rango de γ se estableció entre [1.5, 2], como se plantea en la descripción de la distribución de *Lévy*.

2. Para el número de generaciones, se estableció el rango entre [100, 500].

3. El tamaño mínimo del conjunto *ES* se estableció de acuerdo a los resultados obtenidos por Laguna et. al, 1999, que establecen como bueno, un conjunto de 3 soluciones. Como valor máximo se decidió establecer un conjunto de tamaño 5. El porcentaje para calcular el umbral de aceptación en el PR se estableció entre [0.2, 0.6]. Se realizaron 10 réplicas de simulación para las 2^7combinaciones evaluando el GAP (%) y el tiempo computacional empleado para cada una de las instancias. El diseño factorial se efectuó en la herramienta estadística SAS 9.1.3 ®. Las Figuras 10 y 11 representan respectivamente el efecto estimado de los parámetros para el GAP (%) y tiempo computacional empelado para hallar una solución en la instancia E-n101-k8.

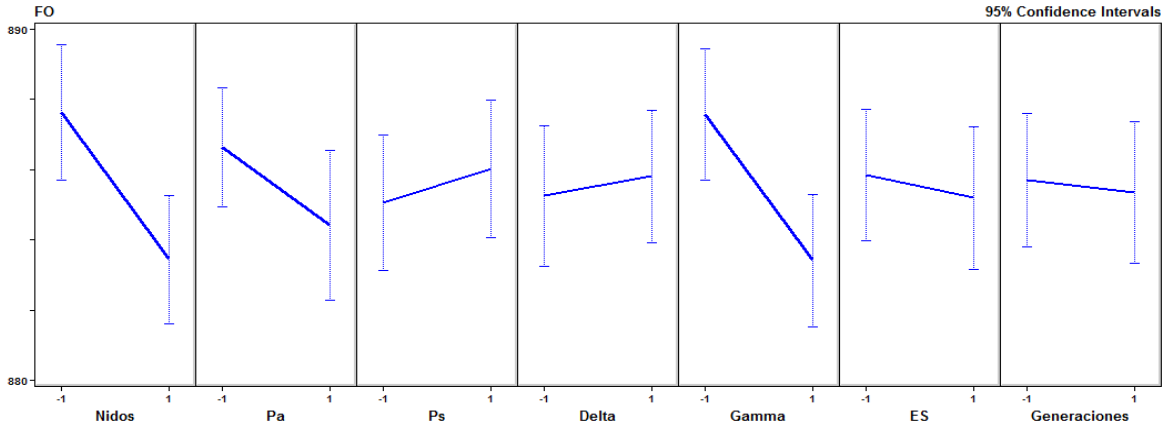


Figura 10. Efectos estimados de los Parámetros en el GAP (%)

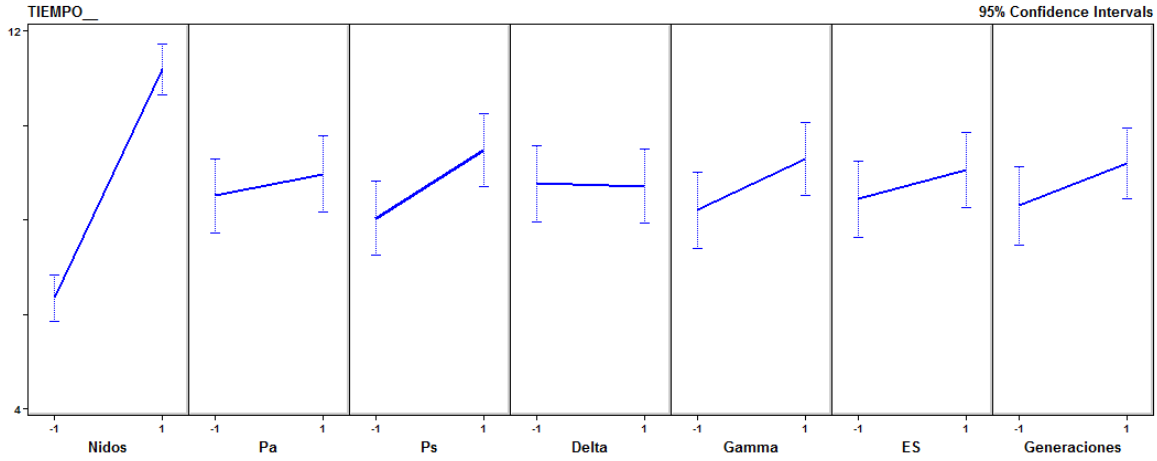


Figura 11. Efectos estimados de los Parámetros en el tiempo de corrida

Como se puede ver en la Figura 10, γ y el número nidos, Pa tienen efecto significativo en el valor de la función objetivo y por ende en la disminución del GAP (%), pero con un costo (tiempo) computacional alto como se ve en la Figura 11. Por su parte Ps y delta en sus valores máximos empeoran la FO y por ende el GAP (%), y además aumentan el tiempo computacional como se evidencia en la Figura 11. Por otro lado el conjunto *ES* y el número de generaciones, tienen un pequeño efecto en el mejoramiento del GAP (%), pero este se alcanza sacrificando el tiempo computacional.

Así mismo con los resultados obtenidos para cada una de las combinaciones de parámetros, se realizó en la herramienta estadística IBM SPSS Statistics 20 ® la prueba no paramétrica de Friedman, que evalúa si hay diferencias significativas entre los rangos de las medias de los tratamientos. Para esto se tomó independientemente de cada instancia probada, el menor GAP (%) y el menor tiempo computacional. El resultado de esta prueba mostró que había diferencias significativas en al menos tres combinaciones de parámetros ($P_{Value} < 0.05$). Dado los resultados de las pruebas estadísticas realizadas, se seleccionó la combinación de parámetros que generara el mayor equilibrio entre el menor GAP (%) y el tiempo computacional empleado. De acuerdo a los resultados anteriores, se establecieron los parámetros como sigue: *m=25; Pa = 0.6; Ps = 0.25; γ = 1.5; ES =3; delta = 0.5 e Iteraciones = 100.* Con el fin de realizar pequeños y

grandes saltos durante cada iteración se estableció que α estuviera en el rango de $[10^4, 10^9]$, lo que llevaría que se hicieran en ocasiones alrededor de mínimo tres movimientos y máximo nueve.

Los resultados se resumen en la Tabla II, la Tabla III y la Tabla IV. En la Tabla II se muestran los resultados obtenidos para el TSP, CVRP y el DCVRP. La primera columna corresponde al nombre de la instancia, la cual lleva asociado el número de nodos del problema y en el caso del CVRP, DCVRP y VRPB, adicionalmente el número de vehículos (rutas) disponibles. La segunda columna identifica el tipo de problema. La tercera muestra la mejor solución conocida (BKS) del problema. El resto de columnas corresponden respectivamente a los resultados obtenidos luego de evaluar la metaheurística de acuerdo a: (1) su estado más básico, sin PR y movimientos de *Lévy*, solo utilizando intercambios *2-opt* para mejorar las soluciones; (2) con PR pero sin movimientos de *Lévy* , los peores nidos solo se perturbaran con intercambios *2-opt*; (3) con movimientos de *Lévy* pero sin PR, no se obtendrán nuevas soluciones a través de los mejores nidos; (4) la propuesta de Cuckoo Search intensivo. Para cada una de estas pruebas se reporta la solución hallada, el tiempo computacional en segundos y el GAP (%).

En la Tabla III se reporta un comparativo entre los resultados del Cuckoo Search intensivo y otras metaheurísticas utilizadas para resolver el TSP, CVRP y el DCVRP. Para el CVRP y el DCVRP nuestra propuesta es comparada con la heurística de mejoramiento de soluciones basada en un programa lineal entero de Franceschi, Fischetti & Toth, et, al. 2006[5], probado en un computador AMD Athlon XP 2400+ PC con 1 GByte RAM. En cuanto al TSP, el Cuckoo Search intensivo es comparado con el Cuckoo Search discreto mejorado -*DCS* (Ouaarab, et. al, 2013), que fue probado en una computadora Intel(R) Core™ 2 Duo 2.00 GHz CPU, y RAM de 3 GB. Este algoritmo comparte los mismos principios de nuestra propuesta, a diferencia de la forma en que se realizan los movimientos de *Lévy*, y que en este método los movimientos tanto de las peores y mejores soluciones son hechos con *Lévy Flight.*

[5] A través de este método fueron halladas las soluciones óptimas para un amplio conjunto las instancias del CVRP y DCVRP. Dado un tour del TSP, el método remueve los nodos del tour y los reasignan óptimamente en clusters. Para esto a cada nodo se le da un peso, y se procede a solucionar el problema de la minima suma de las asignaciones. Para evaluar este método Franceschi, et, al. 2006, parten de soluciones inteligentes del TSP, generadas por los métodos conocidos de barrido, Fisher and Jaikumar o metaheuristicas que encuentren muy buenas soluciones. Así mismo para formar las rutas de cada cluster utilizan la heurística de vecino más cercano y mejoradas posteriormente con intercambios *3-opt*.

Tabla II. Resultados de la implementación del Cuckoo Search Intensivo para la solución del TSP, el CVRP y el DCVRP.

			Cuckoo search sin PR y Levy			Cuckoo search con PR y sin Levy			Cuckoo search con Levy y sin PR			Cuckoo search intensivo		
Instancia	**Problema**	**BKS**	**Solución**	**Tiempo (Seg)**	**GAP**	**Solución**	**Tiempo (Seg)**	**GAP**	**Solución**	**Tiempo (Seg)**	**GAP**	**Solución**	**Tiempo (Seg)**	**GAP**
br17	ATSP	39	39	0,004	0,00%	39	0,004	0,00%	39	0,0044	0,00%	39	0,004	0,00%
p43	ATSP	5620	5620	0,08	0,00%	5620	1,15	0,00%	5620	0,09	0,00%	5620	0,07	0,00%
ft53	ATSP	6905	9031	0,9	30,79%	7500	1	8,62%	7432	0,97	7,63%	6905	1,12	0,00%
eil51	STSP	426	432	0,77	1,41%	430	0,95	0,94%	430	0,79	0,94%	426	1,15	0,00%
berlin52	STSP	7542	7758	0,03	2,86%	7542	1	0,00%	7654	0,71	1,49%	7542	0,09	0,00%
st70	STSP	675	679	1,5	0,59%	675	1,85	0,00%	675	1,71	0,00%	675	1,8	0,00%
pr76	STSP	108159	109375	4,95	1,12%	109730	5,5	1,45%	109250	4,8	1,01%	108159	5,1	0,00%
eil76	STSP	538	562	5,5	4,46%	558	7,53	3,72%	553	7,25	2,79%	538	6,58	0,00%
KroA100	STSP	21282	21454	3,5	0,81%	21383	4,5	0,47%	21410	4,77	0,60%	21282	5,55	0,00%
KroB100	STSP	22141	22279	6,99	0,62%	22184	8,9	0,19%	22372	7,88	1,04%	22141	9,1	0,00%
A-n32-k5	CVRP	784	792	1,12	1,02%	798	2,4	1,79%	798	1,33	1,79%	784	1,7	0,00%
A-n33-k5	CVRP	661	704	1,13	6,51%	680	2,21	2,87%	703	1,1	6,35%	661	2,4	0,00%
A-n44-k6	CVRP	937	967	4,5	3,20%	969	8,78	3,42%	969	4,78	3,42%	937	9,21	0,00%
A-n53-k7	CVRP	1010	1069	7,69	5,84%	1053	16,23	4,26%	1055	7,62	4,46%	1042	16,42	3,17%
A-n55-k9	CVRP	1073	1119	7,82	4,29%	1095	23,71	2,05%	1090	8,23	1,58%	1073	30,21	0,00%
A-n60-k9	CVRP	1354	1415	16,16	4,51%	1411	34,26	4,21%	1410	15,5	4,14%	1393	31	2,88%
A-n63-k9	CVRP	1616	1688	20,16	4,46%	1684	39	4,21%	1684	20	4,21%	1672	150,23	3,47%
A-n80-k10	CVRP	1763	1839	55,26	4,31%	1828	150,43	3,69%	1831	56,7	3,86%	1824	147,79	3,46%
B-n31-k5	CVRP	672	672	1,09	0,00%	675	1,9	0,45%	672	1,13	0,00%	672	1,7	0,00%
B-n34-k5	CVRP	788	795	1,32	0,89%	789	1,59	0,13%	789	1,36	0,13%	788	2,23	0,00%
B-n50-k7	CVRP	741	763	6,66	2,97%	754	6,96	1,75%	755	6,93	1,89%	741	12,49	0,00%
B-n50-k8	CVRP	1312	1348	8,69	2,74%	1363	16,87	3,89%	1365	9,19	4,04%	1321	20,14	0,69%
B-n56-k7	CVRP	707	730	9.92	3,25%	728	23,05	2,97%	729	9,89	3,11%	717	30,5	1,41%
B-n57-k9	CVRP	1598	1626	32,13	1,75%	1625	63,25	1,69%	1626	42,07	1,75%	1625	125,3	1,69%
B-n63-k10	CVRP	1496	1578	46,87	5,48%	1578	99,62	5,48%	1579	47,65	5,55%	1570	113,53	4,95%
B-n64-k9	CVRP	861	911	30,25	5,81%	911	84,15	5,81%	906	29,13	5,23%	902	100,45	4,76%
B-n66-k9	CVRP	1316	1363	23,9	3,57%	1358	43,02	3,19%	1358	20,09	3,19%	1355	55,33	2,96%
B-n78-k10	CVRP	1221	1276	41,77	4,50%	1274	113,91	4,34%	1274	113,91	4,34%	1270	125,48	4,01%
E-n22-k4	CVRP	375	375	0,27	0,00%	375	0,48	0,00%	375	0,28	0,00%	375	0,47	0,00%
E-n33-k4	CVRP	835	854	1,4	2,28%	886	2,45	6,11%	886	1,49	6,11%	835	3,17	0,00%
E-n51-k5	CVRP	521	552	0,5	5,95%	521	1,1	0,00%	521	0,99	0,00%	521	0,99	0,00%
E-n76-k7	CVRP	682	728	24,63	6,74%	728	81,16	6,74%	728	24,13	6,74%	724	83,78	6,16%
E-n101-k8	CVRP	815	873	97,58	7,12%	869	122,15	6,63%	888	113,226	8,96%	828	148,25	1,60%
D022-04g	DCVRP	375	375	0,02	0,00%	375	0,29	0,00%	375	0,15	0,00%	375	0,22	0,00%
D023-03g	DCVRP	660	664	0,3	0,61%	664	0,55	0,61%	660	0,33	0,00%	660	0,5	0,00%
D030-03g	DCVRP	503	504	1,08	0,20%	505	1,64	0,40%	510	1,01	1,39%	503	1,8	0,00%
D033-04g	DCVRP	1106	1329	1,25	20,16%	1305	1,3	17,99%	1210	1,5	9,40%	1106	2,1	0,00%
D051-06c	DCVRP	548	598	5	9,12%	588	14,86	7,30%	588	4,93	7,30%	548	14,97	0,00%
D076-11c	DCVRP	907	998	24,84	10,03%	992	93,62	9,37%	1015	24,6	11,91%	979	87,41	7,94%
D101-11c	DCVRP	866	912	81,74	5,31%	887	302,41	2,42%	894	82,51	3,23%	872	320,66	0,69%

Tabla III. Comparativo entre el Cuckoo Search Intensivo y otros métodos de solución del TSP, el CVRP y el DCVRP.

			ILP - FJ (De Franceschi et, al.2006)			ILP - Barrido (De Franceschi et, al.2006)			IDCS (Ouaarab et, al.2013)			Cuckoo search intensivo		
Instancia	**Problema**	**BKS**	**Solución**	**Tiempo (Seg)**	**GAP**	**Solución**	**Tiempo (Seg)**	**GAP**	**Solución**	**Tiempo (Seg)**	**GAP**	**Solución**	**Tiempo (Seg)**	**GAP**
eil51	STSP	426	-	-	-	-	-	-	426	1,16	0,00%	426	1,15	0,00%
berlin52	STSP	7542	-	-	-	-	-	-	7542	0,09	0,00%	7542	0,09	0,00%
st70	STSP	675	-	-	-	-	-	-	675	1,56	0,00%	675	1,8	0,00%
pr76	STSP	108159	-	-	-	-	-	-	108159	4,73	0,00%	108159	5,1	0,00%
eil76	STSP	538	-	-	-	-	-	-	538	6,54	0,00%	538	6,58	0,00%
KroA100	STSP	21282	-	-	-	-	-	-	21282	2,7	0,00%	21282	5,55	0,00%
KroB100	STSP	22141	-	-	-	-	-	-	22141	8,74	0,00%	22141	9,1	0,00%
A-n53-k7	CVRP	1010	1010	89,7	0,00%	1010	156	0,00%	-	-	-	1042	16,42	3,17%
A-n55-k9	CVRP	1073	1073	121,7	0,00%	1073	69,3	0,00%	-	-	-	1073	30,21	0,00%
A-n60-k9	CVRP	1354	1354	287,1	0,00%	1354	204	0,00%	-	-	-	1393	31	2,88%
A-n63-k9	CVRP	1616	1616	44,5	0,00%	1616	758,4	0,00%	-	-	-	1672	150,23	3,47%
A-n80-k10	CVRP	1763	1763	1201,9	0,00%	1763	1251,6	0,00%	-	-	-	1824	147,79	3,46%
B-n50-k7	CVRP	741	741	120,8	0,00%	741	13,5	0,00%	-	-	-	741	12,49	0,00%
B-n50-k8	CVRP	1312	1312	722,7	0,00%	1312	952,6	0,00%	-	-	-	1321	20,14	0,69%
B-n56-k7	CVRP	707	707	191,9	0,00%	707	366,5	0,00%	-	-	-	717	30,5	1,41%
B-n57-k9	CVRP	1598	1598	781,4	0,00%	1598	781,4	0,00%	-	-	-	1625	125,3	1,69%
B-n63-k10	CVRP	1496	1496	97,3	0,00%	1496	2532,6	0,00%	-	-	-	1570	113,53	4,95%
B-n64-k9	CVRP	861	861	532,7	0,00%	861	322,2	0,00%	-	-	-	902	100,45	4,76%
B-n66-k9	CVRP	1316	1316	2748,8	0,00%	1316	3934,3	0,00%	-	-	-	1355	55,33	2,96%
B-n78-k10	CVRP	1221	1221	276,7	0,00%	1221	2627,7	0,00%	-	-	-	1270	125,48	4,01%
E-n51-k5	CVRP	521	521	0	0,00%	521	27,3	0,00%	-	-	-	521	0,99	0,00%
E-n76-k7	CVRP	682	682	95,3	0,00%	682	122,2	0,00%	-	-	-	724	83,78	6,16%
E-n101-k8	CVRP	815	815	5544	0,00%	815	1442	0,00%	-	-	-	828	148,25	1,60%
D051-06c	DCVRP	548	548	10,9	0,00%	548	48,2	0,00%	-	-	-	548	14,97	0,00%
D076-11c	DCVRP	907	907	2253,7	0,00%	907	2109,9	0,00%	-	-	-	979	87,41	7,94%
D101-11c	DCVRP	866	866	7736,4	0,00%	866	7736,4	0,00%	-	-	-	872	320,66	0,69%

Como se puede observar en las Tablas II y III, nuestro algoritmo resulta muy eficiente ya que en todas las instancias en un tiempo computacional relativamente corto, halla la solución óptima del problema o una buena solución muy cercana al óptimo con valores del GAP no mayores al 5%. Teniendo en cuenta que nuestro algoritmo parte de soluciones generadas aleatoriamente, a diferencia del método desarrollado por Franceschi, et, al. 2006, que parte de soluciones generadas con métodos inteligentes, se demuestra también lo eficiente y práctico que resulta la aplicación del método *Split* para el corte óptimo de una permutación en la solución del CVRP y el DCVRP.

Así mismo en cuanto el Cuckoo Search intensivo y el Cuckoo Search discreto mejorado, los resultados no difieren tanto en solución hallada y tiempo computacional, ya que en ambos casos se encuentran las soluciones optimas de los problemas y en un tiempo computacional que no varía en gran medida uno respecto al otro.

Finalmente en la Tabla IV se reportan los resultados de la aplicación del Cuckoo Search intensivo al VRPB y la comparación con los métodos SF, DB y LHBH.

Tabla IV. Resultados de la implementación del Cuckoo Search Intensivo para la solución del VRPB.

Instancia	BKS (SF)	BKS (DB)	BKS (LHBH)	Cuckoo search intensivo				
				Solución	GAP % (SF)	GAP % (DB)	GAP % (LHBH)	Tiempo (segundos)
A1	269045	230730	230568	230568	0,00%	0,00%	0,00%	0,75
A2	215461	185228	181279	181279	0,00%	0,00%	0,00%	1,5
A3	182533	169216	169975	177331	0,00%	4,80%	4,33%	2,5
B1	254098	276862	239782	232244	0,00%	0,00%	0,00%	1,8
B2	211496	246142	198493	225425	6,59%	0,00%	13,57%	1,69
B3	172297	205907	169372	211481	22,74%	2,71%	24,86%	1,64
C1	265453	315624	256120	256120	0,00%	0,00%	0,00%	4,91
C2	215020	251349	221848	251349	16,90%	0,00%	13,30%	3,9
C3	203988	-	203771	223447	9,54%	-	9,66%	3,2
D1	336599	357301	324502	363572	8,01%	1,76%	12,04%	2,78
D3	242393	265143	239801	262253	8,19%	0,00%	9,36%	5,5
D4	208622	225442	208120	235389	12,83%	4,41%	13,10%	6,74
E1	244488	295874	244377	263591	7,81%	0,00%	7,86%	7,1
E2	218499	280469	223667	243287	11,34%	0,00%	8,77%	11,64
E3	216209	269571	214609	279551	29,30%	3,70%	30,26%	9,25

De acuerdo a los resultados obtenidos se pudo comprobar la eficiencia en el funcionamiento del diseño hecho para el método Split y el corte óptimo de las rutas generadas para solucionar un *VRPB* y de las estrategias de búsqueda planteadas con el Cuckoo Search intensivo, lo cual se refleja en el poco tiempo computacional que se requiere para encontrar una muy buena solución que cumpla con las restricciones del problema. Cómo se puede observar nuestra propuesta encuentra la gran mayoría de las veces una mejor solución que el algoritmo DB, lo que da muestra del potencial de nuestro planteamiento.

5. CONCLUSIONES

En este trabajo hemos propuesto una adaptación a la metaheurística Cuckoo Search para resolver los problemas de ruteo *TSP*, *CVRP*, *DCVRP y VRPB*. Nuestro planteamiento se basa en la metaheurística Cuckoo Search, propuesta por Yang, et, al. 2009, que se diseñó para resolver problemas de tipo continuo y no de tipo discreto como el VRP.

Los aportes hechos a la metaheurística consistieron en cuanto la exploración e intensificación en el espacio de solución, mediante la generación de nuevas soluciones a partir de los mejores nidos en cada iteración con la implementación del *Path Relinking*. Esta estrategia se complementa con los movimientos realizados con *Lévy Flight,* que favorecen ampliamente la diversificación e intensificación del espacio, evitando quedar atrapados en óptimos locales.

Lo anterior se ve reflejado en los resultados mostrados en la Tabla II, donde se comparó este algoritmo con sus versiones primitivas (sin PR y sin *Lévy Flight)* lo que demostró la complementariedad de los dos métodos propuestos, ya que al construir soluciones a partir de las mejores en cada iteración y a la vez saltar de un vecindario a otro gracias a los movimientos de *Lévy,* favorecía la búsqueda soluciones de mejor calidad. Así mismo se evidencia que la propuesta de perturbación de la solución con PR y posterior mejoramiento con intercambios *2-opt*, conlleva a que con pocas iteraciones del problema se llegue rápidamente a buenas soluciones, en comparación a algunos métodos empleados para la solución de este tipo de problemas.

La calidad de las soluciones, se ve complementada con los tiempos computacionales relativamente bajos, que dan bases sólidas para sustentar el potencial de nuestra propuesta y especial de la metaheurística, para la solución de problemas de tipo discreto dada la facilidad con que se generan buenas soluciones, se realizan movimientos a lo largo de los vecindarios del espacio de solución, sin necesidad de sacrificar tiempo computacional, esto sin necesidad de que el vecindario del que parta el algoritmo sea de buena calidad en cuanto la función objetivo del problema.

De igual forma con este trabajo se presenta un valioso aporte con la adaptación del método *Split* para el problema de VRPB, en cuanto al abarcamiento de este, con una estrategia de solución *Route First – Cluster Second*, estrategia poco implementada en este tipo de problema. Se pudo demostrar la sencillez de aplicación del método propuesto y la fácil implementación computacional del mismo, lo que se refleja en las buenas soluciones obtenidas y el bajo tiempo computacional empleado.

En general el método desarrollado demuestra ser muy versátil y flexible, ya que puede ser fácilmente adaptado y aplicado a distintos problemas de optimización discreta y en el caso del VRP a las restricciones propias del problema, tal como se hizo para solucionar el TSP, el CVRP, el DCVRP y el VRPB. Para la estrategia *Route First –Cluster Second* implementada podría fácilmente adaptarse el Cuckoo Search para solucionar el *problema de ruteo de vehículos con recogida y entrega*- VRPPD, teniendo en cuenta que el VRPB es un tipo particular del primero, simplemente se tendrían que restringir los movimientos *2-opt,* Lévy y Path relinking para que no haya intercambios en los que un nodo de entrega se visite antes que su nodo de recogida asociado. Posteriormente se podría aplicar una de las

adaptaciones hechas al método Split para el VRPPD. Así mismo es posible su adaptación e implementación en el *mTSP*, donde solo habría que incluir al modelo la característica de que distintos agentes deben visitar a los diferentes clientes, lo cual es similar a si se tuviese un problema de CVRP.

Bibliografía

Anbuudayasankar, S. P., Ganesh, K., & Mohapatra, S. (2014). *Models for Practical Routing Problems in Logistics* (pp. 11–42). Cham: Springer International Publishing.

Bae, S.-T., Hwang, H. S., Cho, G.-S., & Goan, M.-J. (2007). Integrated GA-VRP solver for multi-depot system. *Computers & Industrial Engineering*, *53*(2), 233–240.

Burnwal, S., & Deb, S. (2012). Scheduling optimization of flexible manufacturing system using Cuckoo Search-based approach. *The International Journal of Advanced Manufacturing Technology*, *64*(5-8), 951–959.

Chernykh, I., Kononov, A., & Sevastyanov, S. (2013). Efficient approximation algorithms for the routing open shop problem. *Computers & Operations ReSearch*, *40*(3), 841–847.

Davendra, D. (2010). *Traveling Salesman Problem theory and applications. In- Tech Publishing.*

Du, L., & He, R. (2012). Combining Nearest Neighbor Search with Tabu Search for Large-Scale Vehicle Routing Problem. *Physics Procedia*, *25*, 1536–1546.

Franceschi, R. De, Fischetti, M., & Toth, P. (2006). A new ILP-based refinement heuristic for Vehicle Routing Problems. *Mathematical Programming*, *105*(2-3), 471–499.

Gajpal, Y., & Abad, P. L. (2009). Multi-ant colony system (MACS) for a vehicle routing problem with backhauls. *European Journal of Operational ReSearch*, *196*(1), 102–117. doi:10.1016/j.ejor.2008.02.025

Goetschalckx, M., Jacobs-Blecha, C., 1989. The vehicle routing problem with Backhauls. European Journal of Operational ReSearch 42, 39-51.

Goetschalckx, M., Jacobs-Blecha, C., 1993. The vehicle routing problem with Backhauls: properties and solution algorithms. Technical Report MHRC-TR-88-13, Georgia Institute of Technology.

Kirkpatrik, S. (1984). Optimization by simulated annealing: quantitative studies. *Journal of Statistical Physics*, 975-986.

Laguna, M. (1998). GRASP and Path Relinking for 2-Layer Straight Line Crossing Minimization.

Lawler, E., Lenstra, J., Rinnooy, A., & Shmoys, D. (1985). *The Traveling Salesman Problem.* Chichester: Wiley.

Lin, S., & Kernighan, B. (1973). An effective heuristic algorithm for the traveling-salesman. *Operations ReSearch*, 498-516.

Martí, R. (2010). *Procedimientos Metaheurísticos en Optimización Combinatoria.* Valencia: Departament d'Estadística i Investigació Operativa.

Martin, O., Otto, S., & E., F. (1991). Large-step markov chains for the traveling salesman. *Complex Systems*, 399-326.

Mercado, R. Z. R., Luis, J., & Velarde, G. (2000). Investigación de operaciones en acción: III(9), 15–20.

Osman, I. H. (1993). Metastrategy simulated annealing and tabu Search algorithms for the vehicle routing problem. *Annals of Operations ReSearch*, *41*(4), 421–451.

Ouaarab, A., Ahiod, B., & Yang, X. (2014). Cuckoo Search and Firefly Algorithm, *516*.

Padherg, M., & Rinald, i. R. (1987). Optimization of a 532-city symmetric travelling salesman. *Operations ReSearch Letters*, 1-7.

Palhazi Cuervo, D., Goos, P., Sörensen, K., & Arráiz, E. (2014). An iterated local Search algorithm for the vehicle routing problem with backhauls. *European Journal of Operational ReSearch*, *237*(2), 454–464.

Prins, C., Lacomme, P., & Prodhon, C. (2014). Order-first Split-second methods for vehicle routing problems: A review. *Transportation ReSearch Part C: Emerging Technologies*, *40*, 179–200.

Prins, C. (2004). A simple and effective evolutionary algorithm for the vehicle routing problem. *Computers & Operations ReSearch*, *31*(12), 1985–2002.

Raju, R., Babukarthik, R. G., & Dhavachelvan, P. (n.d.). Hybrid Ant Colony Optimization and Cuckoo Search Algorithm for Job Scheduling, 491–501.

Resende, M. G. C., & Ribeiro, C. C. (n.d.). Chapter 2 Grasp with Path-Relinking : Recent advances and applications.

Schmid, V., Doerner, K. F., & Laporte, G. (2013). Rich routing problems arising in supply chain management. *European Journal of Operational ReSearch*, *224*(3), 435–448.

Sipahioglu, A., Yazici, A., Parlaktuna, O., & Gurel, U. (2008). Real-time tour construction for a mobile robot in a dynamic environment. *Robotics and Autonomous Systems*, *56*(4), 289–295.

Sur, C., & Shukla, A. (2014). Proceedings of the Third International Conference on Soft Computing for Problem Solving, *258*, 307–320.

Toth, P., & Vigo, D. (1999). A heuristic algorithm for the symmetric and asymmetric vehicle routing problems with backhauls. *European Journal of Operational ReSearch*, *113*(3), 528–543.

Villegas, J. G., Prins, C., Prodhon, C., Medaglia, A. L., & Velasco, N. (2011). A GRASP with evolutionary path relinking for the truck and trailer routing problem. *Computers & Operations ReSearch*, *38*(9), 1319–1334

Yang, X.-S., & Deb, S (2013). Engineering Optimisation by Cuckoo Search. Mathematical Modelling, 1(4), 17.

Yang, X.-S., & Deb, S. (2009). Cuckoo Search via Lévy flight. , World Congress on Nature Biologically Inspired Computing NaBIC 210–214 (2009). IEEE.

Zachariadis, E. E., & Kiranoudis, C. T. (2012). An effective local Search approach for the Vehicle Routing Problem with Backhauls. *Expert Systems with Applications*, *39*(3), 3174–3184.

Zhang, J., & Zhang, Z. (n.d.). Ant Colony Algorithms and Logistics Distribution, 734–740.