ESTRATEGIA GRASP PARA EL PROBLEMA DE ENRUTAMIENTO DE VEHÍCULOS CON RECOGIDA Y ENTREGA SIMULTÁNEA

Alina Fernández Arias, Sira Allende Alonso
Facultad de Matemática y Computación, Universidad de la Habana, Cuba

ABSTRACT
Vehicle Routing Problem with Simultaneous Pickup and Delivery (VRPSPD) is defined as follows: given a set of customers with pickup and delivery demands, a set of vehicles and a depot, the goal is to design least – cost routes originating and terminating at the depot such that all customers are served in single visits and subject to vehicle capacity constraints. The critical feature of the problem is the fluctuating load on the vehicle, thus a route may be not feasible even if the total delivery and the total pickup loads are both below the vehicle's maximum capacity. In this paper, a penalization related to the overload in the routes is included in the objective function and a set of GRASP based methods are proposed. At each iteration, local search is performed using Variable Neighborhood Descent, Random Variable Neighborhood Descent and Simulated Annealing. All strategies were tested using Dethloff benchmark. It was obtained competitive results, it has reached 18 best solutions over 40 intances, in the remaining cases the average gap was arround 0.5 percent.

KEYWORDS: Vehicle Routing Problem, Simultaneous Pickup and Delivery, Local Search, Multi-start, Shaking.

MSC: 90C70.

RESUMEN
El problema de enrutamiento de vehículos con recogida y entrega simultánea se define como sigue: dado un conjunto de clientes, una flota de vehículos y un depósito, el objetivo es diseñar rutas de costo mínimo que comiencen y terminen en el depósito tales que todos los clientes sean atendidos en una única visita y no se exceda la capacidad de cada vehículo. La característica fundamental de este problema es la fluctuación de carga durante el recorrido, luego una ruta puede no ser factible incluso si el total de mercancías por entregar y el total de mercancías por recoger son inferiores a la capacidad del vehículo. En este trabajo se considera la penalización de las rutas por exceso de carga y se propone un conjunto de estrategias de solución basadas en GRASP para abordar el problema. Para realizar la búsqueda local en cada iteración se emplean las metaheurísticas Búsqueda por Entornos Variables Descendentes, Búsqueda Aleatoria por Entornos Variables Descendentes y Recocido Simulado. Todos los Algoritmos propuestos fueron validados con el conjunto de prueba propuesto por Dethloff. Se obtuvieron resultados competitivos con los reportados en la literatura, igualándose la mejor solución reportada en 18 de los 40 casos, en los restantes promedio del error relativo fue inferior al 0.5 por ciento.

1. INTRODUCCIÓN

En los últimos años son cada vez más frecuentes los temas de discusión en torno a los procesos de reciclaje y reutilización de productos. En varios países existen regulaciones que obligan a la industria a responsabilizarse con el ciclo de vida completo de sus productos. Como resultado ha surgido la logística inversa como una rama más de la optimización de las cadenas de suministro.
El objetivo de la logística inversa es la integración óptima del flujo de bienes desde los productores hacia los consumidores con el flujo de desechos o material reciclable desde los consumidores hasta las depósitos. Existen numerosas situaciones en las cuales es necesario manejar el transporte de mercancía en ambos sentidos: industria de bebidas embotelladas, distribución de oxigeno a domicilio, empresas de mensajería, etc. Si la decisión es atender de forma independiente la recogida y

entrega de mercancía, entonces, para cada una de estas actividades es necesario resolver un problema de enrutamiento de vehículos con restricciones de capacidad. Sin embargo, en muchos sistemas de distribución y recolección de mercancía, atender de forma separada estos servicios implica un mal aprovechamiento de la flota. Por esta razón, resulta interesante diseñar e implementar métodos en los que se combinen los procesos de entrega y recogida de forma óptima, particularmente en los escenarios en los cuales los clientes requieran ambos servicios.

El **Problema de Enrutamiento de Vehículos con Recogida y Entrega Simultánea** (VRPSPD por sus siglas en inglés) pertenece a la familia de problemas de enrutamiento de vehículos con restricciones de capacidad. Dado un depósito central y un conjunto de clientes con demandas conocidas de recogida y entrega de mercancía, se desea encontrar en sistema de rutas que permita satisfacer la demanda de todos los clientes al menor costo posible. Todos los recorridos se inician y terminan en el depósito. Se dispone de una flota de vehículos de diferentes capacidades. Al diseñar las rutas es necesario tener en cuenta que la carga del vehículo es una mezcla entre la mercancía recogida en los clientes ya visitados y la que todavía falta por entregar. En ningún punto del recorrido la carga puede exceder la capacidad del vehículo.

El VRPSPD fue presentado por Min [10] en 1989 relacionado con la distribución y recolección de libros entre una biblioteca central y 22 bibliotecas secundarias en la ciudad de Ohio. Min propuso un esquema en dos fases en la cual primero se agrupaban los clientes según sus demandas y la capacidad del vehículo y en la segunda etapa se organizaban las rutas a partir de un recorrido óptimo del problema del viajante.

Aunque existen algunos enfoques exactos para el VRPSPD, estos solo son aplicables a problemas de pequeñas dimensiones: Anbuudayasankar *et al.* [1] proponen un modelo de programación matemática que es resuelto usando CEPLEX para problemas de hasta 15 clientes y Dell'Amico *et al.* [4] desarrollaron un Algoritmo de ramificación y precio para instancias de hasta 40 clientes.

Dada la complejidad de los modelos de programación matemática para el VRPSPD, la mayor parte de las estrategias de solución se basan en métodos heurísticos tanto de búsqueda local como poblacionales.

Una de los enfoques más simples para este problema es la heurística por inserción presentada por Dethloff [5] en el 2001. La idea general es manejar un conjunto de rutas parciales a las que se le van sucesivamente adicionando clientes. En cada iteración se determinan todas las posibles inserciones y se selecciona la de menor costo que sea factible. Este procedimiento se repite mientras queden clientes por atender. Se proponen diferentes criterios de inserción en los que se manejan, de forma separada y/o combinada, los conceptos de distancia y la capacidad del vehículo.

Entre los enfoques basados en búsqueda por vecindades se puede citar el trabajo de Chen *et al.* [3] que tomando como punto de partida una solución factible obtenida por un método de inserción aplican un Algoritmo híbrido en el que se combina la búsqueda tabú con un esquema de aceptación por umbral (*record - to - record travel*). Tang - Montané *et al.* [14] proponen una búsqueda tabú con diferentes estrategias de vecindad. Zachariadis *et al.* [16] combinan una búsqueda tabú con un esquema inspirado en la búsqueda local guiada en el que se penalizan los arcos más costosos, para la solución inicial siguen un enfoque constructivo: crean una ruta para cada cliente y las van uniendo siempre y cuando se mantenga la factibilidad y se logre una reducción en los costos, esto se conoce como estrategia de ahorros. Subramannian *et al.* [13] desarrollan una búsqueda local iterada en la que emplean varias estrategias de vecindad. Polat [11] propone un método de múltiples reinicios basado en búsqueda por entornos variables en el que incluye un mecanismo de perturbación que permite utilizar los óptimos locales previamente obtenidos en las iteraciones sucesivas.

También se reportan varios métodos poblacionales: Zachariadis *et al.* [17] proponen un Algoritmo de memoria adaptativa que incorpora mecanismos para diversificar la búsqueda. Tasan *et al.* [15] desarrollan un Algoritmo genético que codifica las soluciones como una permutación de todos los clientes que luego es dividida en rutas factibles. Goksal *et al.* [8] describen el empleo del método de enjambre de partículas combinado con una búsqueda por entornos variables descendentes.

En este trabajo se presentan un conjunto de estrategias basadas en el procedimiento *Greedy Ran-*

domized Adaptive Search Procedure (GRASP por sus siglas en inglés). El resto de este documento se encuentra organizado como sigue: en 2. se formaliza el VRPSPD. Las estrategias propuestas, las funciones de vecindad utilizadas y las metaheurísticas Búsqueda por Entornos Variables Descendentes (VND por sus siglas en inglés), Búsqueda Aleatoria por Entornos Variables Descendentes (RVND por sus siglas en inglés) y Recocido Simulado (SA por sus siglas en inglés), empleadas para la búsqueda local en cada iteración, se describen en el epígrafe 3. Los procedimientos propuestos se validaron empleando las instancias de prueba de Dethloff [5]. Este conjunto consta de 40 problemas que corresponden con dos tipos diferentes de escenarios geográficos. Los experimentos realizados y los resultados obtenidos se resumen en 4. Finalmente aparecen las conclusiones y referencias bibliográficas.

2. PROBLEMA DE ENRUTAMIENTO DE VEHÍCULOS CON RECOGIDA Y ENTREGA SIMULTÁNEA

El VRPSPD, al igual que el resto de los problemas de planificación de recorridos, pertenece a la clase NP-duro [5]. Formalmente se caracteriza por los siguientes datos: depósito central 0, conjunto de clientes $I = \{1, 2 \ldots, n\}, I^+ = I \cup \{0\}$, para cada cliente $i \in I$ se conoce la demanda de entrega d_i y la demanda de recogida p_i (se asume que la demanda del depósito es cero), costo de viaje c_{ij} entre cada par de cliente $i, j \in I^+$, flota de vehículos, $K = \{1, 2 \ldots, m\}$, para cada vehículo $k \in K$ se conoce su capacidad Q^k.
En el VRPSPD los recorridos comienzan y terminan en el depósito central, donde se carga toda la mercancía para entregar en los clientes y una vez finalizado el trayecto, se descarga toda la mercancía recogida. Cada vehículo puede realizar a lo sumo un viaje. Una ruta es una secuencia ordenada de clientes asociadas a un vehículo en específico, se denota por $[R, k] = [(r_0 = 0, r_1, \ldots, r_l, r_{l+1} = 0), k]$. Sea $[R, k]$ una ruta. El costo de viaje es la suma de los costos de todos los arcos presentes en R. Una solución es un conjunto de rutas denotado por $S = \{[R_1, k_1], [R_2, k_2], \ldots, [R_m, k_m]\}$. El costo de S es la suma de los costos de todas las rutas, se denota por $C(S)$.
El objetivo del VRPSPD es:

$$\min C(S) \tag{1}$$

Una solución S debe satisfacer que cada cliente se visite exactamente una vez y que la carga no exceda a la capacidad en ningún punto del recorrido. A diferencia del problema de enrutamiento de vehículos clásico, en el VRPSPD la factibilidad de las rutas se determina no solo por el subconjunto de clientes que la integran, sino también, por el orden en que estos son visitados. Al asumir una flota limitada, encontrar una solución factible para este problema es NP – duro [2], de ahí la utilidad de diseñar métodos heurísticos de solución. Dada la complejidad asociada a la generación de soluciones factibles, en este trabajo se emplea una simplificación del concepto de factibilidad para la construcción de las soluciones iniciales.

Definición 2..1 *Una ruta* $[R, k]$ *es* ***débil factible*** *si el total de mercancía para entregar, así como el total de mercancía por recoger no exceden la capacidad del vehículo* k*, es decir, si se verifican simultáneamente las expresiones (2) y (3).*

$$\sum_{i=1}^{l} d_{r_i} \leq Q^k \tag{2}$$

$$\sum_{i=1}^{l} p_{r_i} \leq Q^k \tag{3}$$

Para toda ruta débil factible existe un ordenamiento de los clientes que garantizan la condición de factibilidad [7].

3. ESTRATEGIA DE SOLUCIÓN

En este trabajo se propone un enfoque basado en GRASP [6]. La característica fundamental de GRASP es que es una metaheurística de múltiples reinicios que en cada paso realiza una búsqueda local que toma como punto de partida una solución golosa – aleatoria.
Atendiendo a la complejidad asociada a la búsqueda de soluciones factibles se diseñó una estrategia de solución que penaliza en la función objetivo la no factibilidad asociada al exceso de carga. Sea $[R,k]$ una ruta, entonces para todo $j=0,\ldots,l$, la carga se calcula según (4). La penalidad asociada a la ruta se determina a partir del exceso de carga en cada punto, lo que se formaliza en la expresión (5).

$$L\left([R,k],j\right) = \sum_{i=0}^{j} p_{r_i} + \sum_{i=j+1}^{l} d_{r_i} \tag{4}$$

$$P\left([R,k]\right) = \sum_{j=0}^{l} \max\left\{0, L\left([R,k],j\right) - Q^k\right\} \tag{5}$$

Como función objetivo se emplea la expresión (6), donde μ es un factor de peso para penalidad. En la medida que se aumenta el valor de μ se favorece la obtención de soluciones factibles. Para fijar el valor de μ se realizaron experimentos con diferentes valores, obteniéndose los mejores resultados para $\mu = 10$.

$$\min \sum_{s\in S} C\left([R_s,k_s]\right) + \mu P\left([R_s,k_s]\right) \tag{6}$$

3.1. Solucición golosa aleatoria

La construcción de soluciones iniciales es un procedimiento iterativo que sigue el paradigma goloso – aleatorio [6]. En cada iteración se construye una lista restringida de candidatos (RCL por sus siglas en inglés) que contiene los clientes que están *relativamente cerca* del último cliente adicionado a la solución. Un cliente se incluye en la RCL si al añadirlo en la ruta actual no se afecta la factibilidad débil y tal que su costo de inserción se encuentre en el intervalo $[c_{min}, c_{min} + \alpha\left(c_{max} - c_{min}\right)]$ donde c_{min}, c_{max} son, respectivamente, el menor y el mayor costo de inserción. Si $|RCL| = 0$ se inicializa una nueva ruta y en este caso se toma como punto de referencia al depósito. La calidad de la solución golosa – aleatoria depende del valor del parámetro α seleccionado. Si $\alpha = 0$ la solución es completamente golosa y si $\alpha = 1$ la solución es completamente aleatoria. Para fijar el valor de α se realizaron experimentos con diferentes valores, obteniéndose los mejores resultados para $\alpha = 0.2$.

3.2. Búsqueda local

En cada iteración se emplearon las metaheurísticas Búsqueda por Entornos Variables Descendentes (VND) [9], Búsqueda Aleatoria por Entornos Variables Descendentes (RVND) y Recocido Simulado (SA) [6] para realizar la búsqueda local.
VND es una estrategia determinista que se basa en la exploración sucesiva de un conjunto prefijado de funciones de vecindad. El orden que se establece para las vecindades así como los criterios de exploración utilizados dan lugar a diferentes variantes de esta metaheurística. En este sentido la Búsqueda Aleatoria por Entornos Variables Descendentes (RVND) es una extensión de VND en la que se establece un orden aleatorio para la secuencia de vecindades. Tanto VND como RVND son ideales para la exploración en zonas cercanas al óptimo, dado el carácter exhaustivo con que realizan la exploración. Por el contrario, Recocido Simulado es ideal para escapar de los óptimos locales

permitiendo una mayor exploración del espacio de búsqueda.
Se implementaron seis funciones de vecindad:

- **Inter Ruta**
  1. Cruzar dos rutas diferentes
  2. Mover un cliente de una ruta hacia otra diferente
  3. Intercambiar dos clientes de dos rutas diferentes
- **Intra Ruta**
  1. Mover un cliente dentro de una misma ruta
  2. Intercambiar dos clientes de de la misma ruta
  3. Eliminar cruzamientos de 2 aristas (2-Opt)

3.3. Estrategias de solució basadas en GRASP

El procedimiento GRASP clásico (Algoritmo 1) consta de dos estapas: generación de las soluciones iniciales y búsqueda local. Se denota por O^* la mejor solución encontrada y por $C(x)$ el costo de la solución x. Se asume que $C(\emptyset) = +\infty$. Esta notación será empleada en el resto de los Algoritmos descritos en este trabajo.

Algoritmo 1 GRASP

```
1: Construir un conjunto de soluciones golosas-aleatorias → P_0, O* ← ∅
2: for all S_t ∈ P_0 do
3:    Realizar una búsqueda local a partir de S_t → O_t
4:    if C(O_t) < C(O*) then
5:       O_t → O*
6:    end if
7: end for
8: return O*
```

En este trabajo se proponen dos procedimientos basados en GRASP: estrategia BiNivel (Algoritmo 2) en el que se separa la búsqueda local en dos etapas y GRASP con Perturbación (Algoritmo 3) que incluye un mecanismo de ruptura de la solución para reiniciar la búsqueda a partir de un óptimo local.

Algoritmo BiNivel

Teniendo en cuenta la marcada diferencia entre las vecindades *Intra Ruta* y las *Inter Ruta*, en este trabajo se desarrollo una búsqueda local en dos etapas, realizando en cada una de ellas, exclusivamente movimientos Inter Ruta o Intra Ruta (Algoritmo 2). Con esta estrategia lo que se persigue es realizar primero *saltos* dentro del espacio de búsqueda, dejando para el final el proceso de refinamiento, entiéndase por esto, una exploración más exhaustiva al rededor del óptimo local encontrado como resultado de la búsqueda Inter Ruta.

Algoritmo 2 GRASP BiNivel

```
1: Construir un conjunto de soluciones golosas-aleatorias → P_0, O* ← ∅
2: for all S_t ∈ P_0 do
3:    Realizar una búsqueda local considerando solamente movimientos Inter Ruta S_t → Z_t
4:    Realizar una búsqueda local considerando solamente movimientos Intra Ruta Z_t → O_t
5:    if C(O_t) < C(O*) then
6:       O_t → O*
7:    end if
8: end for
9: return O*
```

Ordenamiento exacto de las rutas

Un elemento a resaltar en el Algoritmo 2 es que el paso 4 consiste en encontrar un orden factible de costo mínimo para cada una de las rutas presentes en la solución Z_t. Este problema puede ser resuelto de forma exacta a partir de su modelación en enteros mixtos. Vale resaltar que no se puede asegurar que la soluciones encontradas por esta vía sean óptimos globales pues la asignación de clientes a cada ruta está determinada por un procedimiento heurístico.

A continuación se presenta el modelo de programación matemática utilizado para el ordenamiento óptimo de una ruta del VRPSPD. Sea $[R, k]$ una ruta, J el conjunto de índices de los clientes de la ruta, $J^+ = J \cup \{0\}$ y Q la capacidad del vehículo.

Variables de decisión

$$x_{ij} = \begin{cases} 1 & \text{indica si el arco } \textit{(i,j)} \text{ forma} \\ & \text{parte de la solución} \\ 0 & \text{en otro caso} \end{cases}$$

D_{ij}: Cantidad de mercancía por entregar transportada por el arco (i, j)

P_{ij}: Cantidad de mercancía recogida transportada por el arco (i, j)

Función Objetivo:

$$\min \sum_{i,j \in J^+} x_{ij} c_{ij} \tag{7}$$

Restricciones:

$$\sum_{i \in J^+} x_{ij} = 1 \quad \forall j \in J \tag{8}$$

$$\sum_{i \in J^+} x_{ij} - \sum_{i \in J^+} x_{ji} = 0 \quad \forall j \in J \tag{9}$$

$$D_{0j} = \sum_{i \in J} d_i \tag{10}$$

$$P_{i0} = \sum_{i \in J} p_i \tag{11}$$

$$\sum_{i \in J^+} D_{ij} - \sum_{i \in J^+} D_{ji} = d_j \quad \forall j \in J \tag{12}$$

$$\sum_{i \in J^+} P_{ji} - \sum_{i \in J^+} P_{ij} = p_j \quad \forall j \in J \tag{13}$$

$$D_{ij} + P_{ij} \leq x_{ij} Q \quad \forall i, j \in J^+ \tag{14}$$

$$x_{ij} \in \{0, 1\} \quad \forall i, j \in J^+ \tag{15}$$

$$D_{ij} \geq 0 \quad \forall i, j \in J^+ \tag{16}$$

$$P_{ij} \geq 0 \quad \forall i, j \in J^+ \tag{17}$$

La función objetivo (8) es minimizar el costo de viaje. La restricción (8) indica que cada cliente se incluye una única vez en la solución y (9) se refiere a la continuidad del recorrido a partir de un punto. Con (10) y (11) se asegura que toda la mercancía para entregar se cargue en el depósito al cual se regresa con toda la mercancía recogida. (12) - (13) garantizan que se satisfagan las demandas de cada cliente. La restricción (14) corresponde al control de la carga en cada punto del recorrido. (15)-(17) se refieren a las cotas de las variables. El modelo de programación matemática descrito fue resuelto empleando GAMS [12] en su versión académica.

Algoritmo de Perturbación

La desventaja fundamental de los Algoritmos 1 y 2 es que no se aprovechan, en las iteraciones sucesivas, información del óptimo local previamente obtenido. En este trabajo se propone la inclusión de mecanismos de ruptura que modifiquen aleatoriamente un fragmento de la solución para reiniciar

la búsqueda local desde un punto que comparta parte de su estructura. Esta idea se ilustra en el Algoritmo 3. Se asume que la cantidad de veces que se reinicia la búsqueda local es una constante dada N. Se emplearon dos mecanismos de perturbación: intercambiar m clientes de la ruta a con n clientes de la ruta b y eliminar 20 por ciento de los clientes y reinsertarlos nuevamente en posiciones aleatorias.

Algoritmo 3 GRASP con Perturbación

1: Construir un conjunto de soluciones golosas-aleatorias $\rightarrow P_0$, $O^* \leftarrow \emptyset$
2: **for all** $S_t \in P_0$ **do**
3: $O_{t0} \leftarrow S_t$
4: **for** $i = 1$ **to** N **do**
5: Seleccionar aleatoriamente un mecanismo de ruptura u_i
6: Modificar el óptimo local previamente obtenido empleando $u_i \rightarrow S_{ti}$
7: Realizar una búsqueda local a partir de $S_{ti} \rightarrow O_{ti}$
8: **if** $C(O_{ti}) < C(O^*)$ **then**
9: $O_{ti} \rightarrow O^*$
10: **end if**
11: **end for**
12: **end for**
13: **return** O^*

Si la solución encontrada en el paso 7 del Algoritmo 3 es débil factible, entonces se puede lograr un menor costo al ordenar los clientes empleando el modelo de programación matemática descrito en el epígrafe 3.2. Teniendo en cuenta el costo computacional de encontrar la solución exacta se sugiere realizar una única ejecución del modelo en la última iteración del bloque de perturbación. Esta modificación se ilustra en el Algoritmo 4.

Algoritmo 4 GRASP con Perturbación Extendido

1: Construir un conjunto de soluciones golosas-aleatorias $\rightarrow P_0$, $O^* \leftarrow \emptyset$
2: **for all** $S_t \in P_0$ **do**
3: $O_{t0} \leftarrow S_t$
4: **for** $i = 1$ **to** N **do**
5: Seleccionar aleatoriamente un mecanismo de ruptura u_i
6: Modificar el óptimo local previamente obtenido empleando $u_i \rightarrow S_{ti}$
7: Realizar una búsqueda local a partir de $S_{ti} \rightarrow O_{ti}$
8: **if** $i = N \wedge O_{ti}$ es débil factible **then**
9: Realizar un ordenamiento exacto de las rutas de $O_{ti} \rightarrow O^e_{ti}$
10: **else**
11: $O_{ti} \rightarrow O^e_{ti}$
12: **end if**
13: **if** $C(O^e_{ti}) < C(O^*)$ **then**
14: $O^e_{ti} \rightarrow O^*$
15: **end if**
16: **end for**
17: **end for**
18: **return** O^*

4. RESULTADOS NUMÉRICOS

Los procedimientos presentados en el epígrafe 3. fueron evaluados empleando el conjunto de prueba propuesto por Dethloff [5]. Este juego de datos consta de 40 problemas de 50 clientes cada uno que se corresponden con dos tipos diferentes de escenarios geográficos. En los problemas SCA los clientes se distribuyen uniformemente en el conjunto $[0, 100] \times [0, 100]$, mientras que en CON la mitad de los clientes se distribuye como en SCA y el resto en $[\frac{100}{3}, \frac{200}{3}] \times [\frac{100}{3}, \frac{200}{3}]$, lo que se corresponde con la distribución urbana en las zonas céntricas. En todos los casos se asume una flota homogénea de vehículos. Atendiendo a las características de la distribución geográfica de los clientes y la capacidad de los vehículos, este conjunto puede separarse en cuatro grupos de 10 instancias cada uno:

SCA3x Clientes dispersos con pocas rutas. Estos son las instancias más simples del conjunto, al tener pocas rutas se reduce la dimensión del espacio de búsqueda y consecuentemente el espacio factible.

SCA8x Clientes dispersos con muchas rutas. Al aumentar el número de rutas aumenta la complejidad del problema, tanto desde el punto de vista de los costos pues se tienen un mayor número de combinaciones posibles, como desde el punto de vista de la factibilidad, pues en el VRPSPD esta depende de los clientes que conforman la ruta y del orden en que son visitados.

CON3x Problemas urbanos con pocas rutas. La dificultad de la distribución geográfica de clientes que responda a configuraciones urbanas radica en que la estructura de la solución óptima no responde, necesariamente, a la configuración clásica de *pétalos*, lo que aumenta la complejidad del problema desde el punto de vista de lograr soluciones de costo mínimo.

CON8x Problemas urbanos con muchas rutas. Estas son las instancias más complejas del conjunto, pues combinan la configuración urbana con un mayor número de rutas que en los problemas CON3x.

Dadas las restricciones de capacidad, en todos los problemas propuestos por Dethloff, se dificulta la obtención de soluciones factibles.

Estas instancias han sido ampliamente utilizadas en la literatura para la comparación de resultados. En la tabla 1 se resume, según el conocimiento de los autores, la mejor solución obtenida para cada una de ellas. Los métodos de solución reportados se basan en la combinación de varios enfoques heurísticos, vale resaltar que en ningún caso se usa GRASP.

Se realizaron diferentes experimentos con todos los procedimientos descritos en el epígrafe 3. Los valores de $\alpha = 0.2$ y $\mu = 10$ se obtuvieron como resultado de una experimentación previa en la que se analizó su influencia en la calidad de la solución final. En los Algoritmos 1, 3 y 4 se hicieron pruebas con las tres metaheurísticas: VND, RVND y SA. En el caso del Algoritmo 2 las combinaciones analizadas fueron VND-VND, RVND-RVND, SA-VND, SA-RVND, VND-GAMS, RVND-GAMS, SA-GAMS.

El único parámetro de la metaheurística VND es el orden en que deben ser analizadas las vecindades. En este trabajo se consideraron dos secuencias: una consiste en realizar primero los movimientos Intra Ruta seguido de los Inter Ruta y la otra es en sentido inverso. En el caso de SA los valores de temperatura inicial y final se calcularon a partir de los costos de las soluciones iniciales, expresiones (18) y (19) respectivamente, donde $C(S)$ es el costo de la solución inicial. Los valores empleados para los restantes parámetros se muestran en la tabla 2.

$$T_0 = \frac{-w * C(S)}{\log p} \tag{18}$$

$$T_f = T_0 * \beta^N \tag{19}$$

En los experimentos realizados no se observaron diferencias significativas entre los mejores resultados obtenidos por las distintas metaheurísticas. En la tabla 3 se reporta la mejor solución encontrada por cada una de los procedimientos implementados: GRASP clásico (A. 1), BiNivel (A. 2), Perturbación (A. 3) y Perturbación Extendido(A. 4). Se incluyen además las columnas A. 1_G y A. 3_G donde se

Tabla 1: Descripción de las instancias de prueba

Problema	Costo	Rutas	Problema	Costo	Rutas
SCA30	635.62	4	CON30	616.52	4
SCA31	697.84	4	CON31	554.47	4
SCA32	659.34	4	CON32	518.00	4
SCA33	680.04	4	CON33	591.19	4
SCA34	690.50	4	CON34	588.79	4
SCA35	659.90	4	CON35	563.70	4
SCA36	651.09	4	CON36	499.05	4
SCA37	659.17	4	CON37	576.48	4
SCA38	719.47	4	CON38	523.05	4
SCA39	681.00	4	CON39	578.25	4
SCA80	961.50	9	CON80	857.17	9
SCA81	1049.65	9	CON81	740.85	9
SCA82	1039.64	9	CON82	712.89	9
SCA83	983.34	9	CON83	811.07	10
SCA84	1065.49	9	CON84	772.25	9
SCA85	1027.08	9	CON85	754.88	9
SCA86	971.82	9	CON86	678.92	9
SCA87	1051.28	9	CON87	811.96	9
SCA88	1071.18	9	CON88	767.53	9
SCA89	1060.50	9	CON89	809.00	9

Tabla 2: Parámetros del Recocido Simulado

Descripción	Parámetro	Valor
Diferencia porcentual con la peor solución aceptada	w	0.2
Probabilidad de aceptar una solución	p	0.3
Factor de enfriamiento	β	0.95
Cantidad de enfriamientos	N	100
Repeticiones a la misma temperatura	γ	20
Esquema de enfriamiento	geométrico	

muestra, siempre y cuando haya ocurrido una disminución en el costo, el resultado de ordenar de forma exacta las rutas de las mejores soluciones obtenidas por los Algoritmos GRASP Clásico y Perturbación respectivamente. Se señala en negritas la mejor solución reportada para el problema y subrayado la mejor solución encontrada por las estrategias propuesta en caso de no coincidir con la de la literatura. Las diferencias inferiores a $|0.03|$ se deben a errores de redondeo en los datos. No se muestra el número de rutas porque en todos los casos coincide con el reportado en la literatura.
En la versión clásica de GRASP (Algoritmo 1) los mejores resultados se alcanzaron con la metaheurística SA y VND considerando la secuencia de movimientos Inter Ruta - Intra Ruta. En el enfoque en dos niveles (Algoritmo 2) las mejores combinaciones fueron RVND-RVND, RVND-GAMS y SA-GAMS.
En estrategia de perturbación (Algoritmo 3) y su extensión (Algoritmo 4) se realizaron experimentos manteniendo constante la cantidad de soluciones iniciales y variando el número de perturbaciones. Esto se traduce en un mayor número de iteraciones totales del Algoritmo y aunque los resultados obtenidos son muy similares, de manera general se logra un mayor rendimiento al aumentar el número de iteraciones. En el Algoritmo 3, las soluciones de mejor calidad se obtuvieron con la metaheurística VND considerando la secuencia de movimientos Inter Ruta - Intra Ruta y en el caso del Algoritmo 4 con RVND y SA.

Tabla 3: Mejores soluciones obtenidas por cada Algoritmo

Problema	Literatura	A.1	$A.1_G$	A.2	A.3	$A.3_G$	A.4
SCA30	**635.62**	636.37		640.56	636.09		636.06
SCA31	**697.84**	**697.83**		**697.83**	**697.83**		**697.83**
SCA32	**659.34**	**659.33**		664.19	**659.33**		**659.33**
SCA33	**680.04**	**680.03**		680.58	**680.03**		**680.03**
SCA34	**690.50**	**690.48**		**690.48**	**690.48**		**690.48**
SCA35	**659.90**	661.08	**659.91**	664.69	**659.91**		**659.91**
SCA36	**651.09**	652.95		653.69	**651.11**		**651.11**
SCA37	**659.17**	666.17		660.80	**659.18**		**659.18**
SCA38	**719.47**	**719.50**		**719.50**	**719.50**		**719.50**
SCA39	**681.00**	**681.02**		681.25	**681.02**		**681.02**
SCA80	**961.50**	983.94		997.47	972.97	970.67	970.67
SCA81	**1049.65**	1068.26		1090.75	1057.09	1054.93	1067.24
SCA82	**1039.64**	1052.92		1060.24	1047.57		1044.97
SCA83	**983.34**	1008.46		1021.15	1005.19	994.77	1002.83
SCA84	**1065.49**	1067.74	**1065.49**	1086.69	**1065.49**		1067.66
SCA85	**1027.08**	1058.17		1063.55	1039.87		1043.58
SCA86	**971.82**	977.89		987.69	973.26		**971.84**
SCA87	**1051.28**	1076.38		1071.55	1067.03		1067.12
SCA88	**1071.18**	1086.80	1082.91	1080.59	**1071.20**		**1071.20**
SCA89	**1060.50**	1090.03		1078.86	1065.60		1073.96
CON30	**616.52**	**616.50**		623.36	**616.50**		**616.50**
CON31	**554.47**	557.18		560.30	**554.45**		**554.45**
CON32	**518.00**	521.39		522.09	519.13		519.13
CON33	**591.19**	**591.20**		591.21	**591.20**		**591.20**
CON34	**588.79**	591.41		589.88	**588.78**		**588.78**
CON35	**563.70**	564.86		565.10	**563.67**		**563.67**
CON36	**499.05**	502.17		502.17	501.28	499.38	500.36
CON37	**576.48**	582.10		581.37	578.18		578.41
CON38	**523.05**	**523.08**		**523.08**	**523.08**		**523.08**
CON39	**578.25**	582.69		579.65	582.69		**578.25**
CON80	**857.17**	880.60		867.41	867.33	864.40	860.60
CON81	**740.85**	742.26		755.79	742.93		**740.83**
CON82	**712.89**	716.34		717.59	**712.89**		713.44
CON83	**811.07**	828.16		832.24	818.43		816.27
CON84	**772.25**	787.24		782.27	773.59	772.41	773.73
CON85	**754.88**	759.83		772.16	758.11		758.11
CON86	**678.92**	690.59		686.80	683.08		687.96
CON87	**811.96**	816.38		817.86	813.02		814.50
CON88	**767.53**	783.16		783.74	778.36	770.76	769.66
CON89	**809.00**	821.26		816.75	811.15		812.04

Al ordenar las rutas finales obtenidas por los Algoritmos GRASP clásico y Perturbación empleando el modelo de programación matemática se obtuvieron mejoras solamente en 3 y 7 instancias respectivamente, lo que apunta hacia la efectividad de las estrategias Intra Ruta y a la necesidad de mejorar las Inter Ruta.
Los mejores resultados se obtuvieron con el método de Perturbación, igualándose 18 de de las soluciones reportadas en la literatura y un gap promedio de 0.48% en el resto de los casos. La mayor efectividad se alcanzó en los problemas con menor número de rutas SCA3x (9 de 10) y CON3x (6 de 10), lo que reafirma la importancia de ampliar la búsqueda Inter Ruta. Si bien es cierto que en el enfoque de Perturbación se realiza un mayor número de iteraciones totales, en comparación con las otras dos propuestas, reiniciar la búsqueda desde un punto cercano al óptimo local previamente encontrado favorece la exploración en zonas prometedoras del espacio lo que se refleja en la calidad de la mejor solución encontrada.

5. CONCLUSIONES

En este trabajo se proponen dos procedimientos basados en GRASP para el Problema de Enrutamiento de Vehículos con Recogida y Entrega Simultánea. En el Algoritmo BiNivel (Algoritmo 2) se separa la búsqueda local en dos etapas y en el Algoritmo GRASP con Perturbación (Algoritmo 3) se incluye un mecanismo de ruptura de la solución para reiniciar la búsqueda a partir de un óptimo local.
Se desarrollaron seis funciones de vecindad: tres Intra Ruta y tres Inter Ruta y se utilizaron las metaheurísticas VND, RVND y SA para la búsqueda local. También se empleó la solución exacta del modelo de programación matemática en enteros mixtos para la exploración Intra Ruta.
Al analizar los resultados obtenidos, se puede apreciar un bajo rendimiento de los métodos desarrollados en los problemas con un mayor número de rutas (SCA8x y CON8x), lo que sugiere un aumento de la exploración Inter Ruta. Por el contrario, las discretas mejoras obtenidas al ordenar las rutas finales usando GAMS son un indicador de la efectividad de los métodos Intra Ruta diseñados.
Aunque no se observan diferencias significativas entre los resultados de los dos procedimientos, vale resaltar que el Algoritmo de Perturbación tuvo un mayor rendimiento logrando una efectividad de aproximadamente el 50% para el conjunto de prueba de Dethloff.

AGRADECIMIENTOS

Las autoras quieren agradecer al Colectivo de Optimización de la Facultad de Matemática y Computación por las sugerencias realizadas durante el desarrollo de esta investigación.

RECEIVED: SEPTIEMBRE, 2016
REVISED: MAYO, 2017

REFERENCIAS

[1] ANBUUDAYASANKAR, S. AND MOHANDAS, K. [2007]: Mixed-integer linear programming for vehicle routing problem with simultaneous delivery and pick-up with maximum route-length **The International Journal of Applied Management and Technnology**, 6, 1,:31–52.

[2] BALDACCI, R., BATARRA, M., AND VIGO, D. [2008]: **The vehicle routing problem: latest advances and new challenges**, chapter Routing heterogeneous fleet of vehicles, pages 3–28 Springer.

[3] CHEN, J.-F. AND WU, T.-H. [2006]: Vehicle routing problem with simultaneous deliveries and pickups **Journal of the Operational Research Society**, 57, 5,:579–587.

[4] DELL'AMICO, M., RIGHINI, G., AND SALANI, M. [2006]: A branch-and-price approach to the vehicle routing problem with simultaneous distribution and collection **Transportation Science**, 40, 2,:235–247.

[5] DETHLOFF, J. [2001]: Vehicle routing and reverse logistics: the vehicle routing problem with simultaneous delivery and pick-up **OR Spektrum**, 23:79–96.

[6] DRÉO, J., PÉTROWSKI, A., AND TAILLARD, E. [2006]: **Metaheuristics for Hard Optimization** Springer.

[7] FERNÁNDEZ-ARIAS, A. [2010]: Problema de enrutamiento de vehículos con recogida y entrega simultánea considerando una flota heterogénea Master's thesis, Facultad de Matemática y Computación. Universidad de La Habana.

[8] GOKSAL, F., KARAOGLAN, I., AND ALTIPARMK, F. [2013]: A hybrid discrete particle swarm optimization for vehicle routing problem with simultaneous pickup and delivery **Computers and Industrial Engineering**, 65:39–53.

[9] HANSEN, P., MLADENOVIC, N., AND MORENO-PÉREZ, J. [2003]: Búsqueda de entorno variable **Revista Iberoamericana de Inteligencia Artificial**, , 19,:72–92.

[10] MIN, H. [1989]: The multiple vehicle routing problem with simultaneous delivery and pick-up points **Transportation Reseach**, 23A, 5,:377–386.

[11] POLAT, O., C-B, K., KULAK, O., AND GUNTHER, H.-O. [2015]: A perturbation based variable neighborhood search heuristic for solving the vehicle routing problem with simultaneous pickup and delivery with time limit **European Journal of Operational Research**, pages 369–382.

[12] Rosenthal, R. **A GAMS Tutorial** www.gams.com.

[13] SUBRAMANIAN, A. AND CABRAL, L. [2008]: An ils based heuristic for the vehicle routing problem with simultaneous pickup and delivery and time limit In van Hemert, J. and Cotta, C., editors, **EvoCOP**, pages 135–146. Springer-Verlag.

[14] TANG-MONTANÉ, F.-A. AND GALVAO, R.-D. [2006]: A tabu search algorithm for the vehicle routing problem with simultaneous pick-up and delivery service **Computers and Operations Reseach**, 33:595–619.

[15] TASAN, A.-S. AND GEN, M. [2012]: A genetic based approach to vehicle routing problem with simultaneous pick-up and deliveries **Computers and Industrial Engieneering**, 62:755–761.

[16] ZACHARIADIS, E., TARANTILIS, C., AND KIRANOUDIS, C. [2009]: A hybrid metaheuristic algorithm for the vehicle routing problem with simultaneous delivery and pick-up service **Expert System with Applications**, 36:1070–1081.

[17] ZACHARIADIS, E., TARANTILIS, C., AND KIRANOUDIS, C. [2010]: An adaptive memory methodology for the vehicle routing problem with simultaneous delivery and pick-up service **Expert System with Applications**, 36:1070–1081.