Ciencias Matemáticas, Vol. 31, No. 1, Pag. 1-8, 2017
Recibido 09-2016

Una estrategia GRASP aplicada al Problema de Enrutamiento de Vehículos con Recogida y Entrega Simultánea

A GRASP strategy for the Vehicle Routing Problem with Simultaneous Pickup and Delivery

Alina Fernández Arias[1]*, Sira Allende Alonso[2]

Resumen El Problema de Enrutamiento de Vehículos con Recogida y Entrega Simultánea es una extensión del Problema de Enrutamiento de Vehículos con Restricciones de Capacidad. El objetivo de este problema es diseñar rutas que satisfagan la demanda de recogida y entrega de los clientes en una única visita y se minimice el costo total de transportación. En este trabajo se presenta una estrategia de penalización basada en la metaheurística *Greedy Randomized Adaptive Search Procedure*. En la construcción de las soluciones iniciales se utilizó una simplificación del concepto de factibilidad y se empleó la Búsqueda por Entornos Variables Descendentes para realizar la búsqueda local en cada paso. En este trabajo se proponen diferentes variantes de GRASP. Los resultados computacionales muestran el comportamiento de los algoritmos desarrollados para el conjunto de prueba descrito por Salhi y Nagy.

Abstract The Vehicle Routing Problem with Simultaneous Pickup and Delivery is an extension of the well-known Capacitated Vehicle Routing Problem. The goal of this problem is to design routes in such a way that pickup and delivery demands of each customer must be performed with the same vehicle and the overall cost is minimized. In this paper a penalization approach based on Greedy Randomized Adaptive Search Procedure is presented. For the initial solution a relaxation of the feasibility constrain is considered and Variable Neighborhood Descendant is used to perform local search at each step. This study proposes different variants of GRASP. Experimental results show the performance of the proposed approach for the Salhi and Nagy benchmark.

Palabras Clave
enrutamiento de vehículos — recogida y entrega simultánea — GRASP — VND

[1] *Departamento de Matemática, Universidad de La Habana, La Habana, Cuba, aly@matcom.uh.cu*
[2] *Departamento de Matemática Aplicada, Universidad de La Habana, La Habana, Cuba, sira@matcom.uh.cu*
***Autor para la Correspondencia**

Introducción

El transporte de mercancía es un tema al que se dedica gran atención en las industrias. Estudios realizados muestran que el costo de transportación puede representar hasta un 20 por ciento de los costos totales de los productos [18], lo que justifica el empleo de técnicas de la inteligencia artificial y de la investigación operacional para asistir a la planificación eficiente de los sistemas de distribución.

Existen numerosas situaciones en las cuales es necesario manejar el flujo de materiales desde los centros de distribución hacia los consumidores y desde éstos hasta los depósitos o puntos de reciclaje: industria de bebidas embotelladas, empresas de mensajería, distribución de mercancía y recogida de merma en los supermercados, etc. Atender de forma separada estos servicios implica un mal aprovechamiento de la flota de vehículos, por lo que resulta interesante diseñar e implementar métodos en los que se combinen los procesos de recogida y entrega de forma óptima, particularmente en 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 con restricciones de capacidad en la flota. Formalmente se define como sigue: dado un depósito central y un conjunto de clientes con demandas conocidas de recogida y entrega de mercancía, el objetivo del VRPSPD es diseñar un sistema de rutas, con el menor costo posible, que permita satisfacer las demandas de todos los clientes. Cada cliente es visitado exactamente una vez. Todas las rutas comienzan y terminan en el depósito central, donde se carga la mercancía a distribuir en los clientes y, una vez finalizado el recorrido, se descarga la mercancía recogida. Se garantiza además que la carga del vehículo no exceda su

capacidad.

La característica esencial de este problema es que la carga del vehículo es una mezcla entre la mercancía previamente recogida y la que aún falta por entregar, lo que implica que la factibilidad debe ser verificada en cada punto del recorrido.

Dada la complejidad asociada a la búsqueda de soluciones factibles para el VRPSPD, en este trabajo se propone una estrategia que penaliza en la función objetivo la no factibilidad de las rutas por exceso de carga. Se diseñó un método basado en la metaheurística *Greedy Randomized Adaptive Search Procedure* (GRASP por sus siglas en inglés) en el que se empleó la metaheurística Búsqueda por Entornos Variables Descendentes (VND por sus siglas en inglés) para realizar la búsqueda local en cada iteración de GRASP. Para validar los procedimientos propuestos se empleó el conjunto de prueba propuesto por Salhi y Nagy en 1999 [14].

El documento ha sido organizado como sigue: en la sección 1.1 se formalizan aspectos relativos al VRPSPD y se comentan algunos de los principales resultados en torno al problema. En 2 se describe la estrategia de solución propuesta. Los resultados obtenidos son comentados en 3. Finalmente aparecen las conclusiones y referencias bibliográficas.

1. Problema de Enrutamiento de Vehículos con Recogida y Entrega Simultánea

El problema de enrutamiento de vehículos con recogida y entrega simultánea, al igual que el resto de los problemas derivados del VRP, pertenece a la clase NP-duro [5]. Su objetivo es diseñar un conjunto de rutas, con el menor costo posible, que permitan satisfacer la demanda de todos los clientes en una única visita, garantizando que la mercancía transportada no exceda la capacidad del vehículo. El VRPSPD fue presentado por Min [11] 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 organizan las rutas a partir de un recorrido óptimo del problema del viajante.

Aunque existen algunos enfoques exactos para el VRPSPD, estos sólo 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.

Una de los enfoques más simples para este problema es la heurística por inserción presentada por Dethloff [5]. La idea general es manejar un conjunto de rutas parciales en las que adicionan los clientes sucesivamente. 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 utilizan diferentes criterios de inserción en los que se manejan, de forma separada y/o combinada, la distancia y la capacidad del vehículo.

Entre los métodos basados en búsqueda por vecindades se pueden citar los trabajos de Chen et al.[3](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 *record-to-record travel*), Tang-Montané et al. [16](búsqueda tabú con diferentes estrategias de vecindad), Zachariadis et al. [19](búsqueda tabú combinada con una búsqueda local guiada, la solución inicial la construyen siguiendo un enfoque de ahorros), Subramannian et al. [15](partiendo de una solución golosa desarrollan una estrategia de búsqueda local iterada en la que emplean varias estrategias de vecindad) y Polat [12](método de múltiples reinicios basado en búsqueda por entornos variables, incluyen un mecanismo de perturbación que permite utilizar los óptimos locales previamente obtenidos en las iteraciones sucesivas)

Se reportan también varios enfoques poblacionales: Zachariadis et al. [20](algoritmo de memoria adaptativa que incorpora mecanismos para diversificar la búsqueda), Tasan et al. [17](algoritmo genético que codifica las soluciones como una permutación de todos los clientes que es dividida en rutas factibles) y Goksal et al. [8](enfoque híbrido en el que se combina la versión discreta de la optimización por enjambre de partículas con la búsqueda por entornos variables descendentes).

1.1 Formalización del VRPSPD

Una instancia del VRPSPD se caracteriza por los siguientes datos: depósito central 0, conjunto de clientes $I = \{1,\ldots,n\}$ $I^+ = I \cup \{0\}$, para cada cliente $i \in I$ se conoce la demanda de entrega d_i y la demanda de recogida p_i, costo de viaje c_{ij} entre cada par de cliente $i, j \in I^+$, flota de vehículos, $K = \{1,\ldots,m\}$, para cada vehículo $k \in K$ se conoce su capacidad Q^k y costo asociado f^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, denotada por $[R,k] = [(r_0 = 0, r_1, \ldots, r_l, r_{l+1} = 0), k]$.

Sea $[R,k]$ una ruta. El costo de la ruta es el costo de viaje más el costo asociado al vehículo (expresión (1)). El costo de viaje (TC) es la suma de los costos de todos los arcos presentes en R.

$$C([R,k]) = TC([R,k]) + f^k \tag{1}$$

Una solución es un conjunto de rutas denotado por $S = \{[R_1,k_1],\ldots,[R_m,k_m]\}$. El costo de S se calcula según la ex-

presión (2).

$$C(S) = \sum_{s \in S} C([R_s, k_s]) \tag{2}$$

El objetivo del VRPSPD es:

$$\text{mín} C(S) \tag{3}$$

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 sólo por el subconjunto de clientes que la integran, sino también, por el orden en que estos son visitados. Luego, al asumir una flota limitada, encontrar una solución factible para este problema es NP – duro [2], de ahí la necesidad de diseñar métodos heurísticos de solución.

En este trabajo se diseñó una estrategia que penaliza en la función objetivo la no factibilidad asociada al exceso de carga, atendiendo a la complejidad asociada a la búsqueda de soluciones factibles. La particularidad de este problema es que en cada punto del recorrido la carga es una mezcla entre la mercancía previamente recogida y la que aún falta por entregar. Sea $[R,k]$ una ruta, entonces para todo $j = 0, \ldots, l+1$, la carga se calcula según (4).

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

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).

$$P([R,k]) = \sum_{j=0}^{l+1} \text{máx}\left\{0, L([R,k], j) - Q^k\right\} \tag{5}$$

La función objetivo considerada en este trabajo es 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.

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

2. Estrategia de Solución

En este trabajo se propone un enfoque GRASP[6] en el que se emplea la Búsqueda por Entornos Variables Descendentes [9]-[10] para realizar la búsqueda local en cada iteración. La característica fundamental de GRASP (Algoritmo 1) es que es una metaheurística de múltiples reinicios que en cada paso realiza una búsqueda determinista que toma como punto de partida una solución golosa – aleatoria.

Algoritmo 1 GRASP

1: **for** $k = 1$ **to** K **do**
2: Construir una solución golosa-aleatoria S_k
3: Realizar una búsqueda local empleando VND a partir de $S_k \rightarrow O_k$
4: **end for**
5: **return** $\text{mín}\{O_i\}_{i=1,\ldots,K}$

2.1 Solución Golosa – Aleatoria

La construcción de soluciones iniciales es un procedimiento iterativo que sigue el paradigma goloso – aleatorio. En cada iteración se construye una lista restringida de candidatos donde se incluyen los clientes que están *relativamente cerca* del último cliente insertado en la solución. La cercanía está determinada por un parámetro $\alpha \in [0,1]$.

Como se comentó en la sección 1.1, para el VRPSPD la factibilidad de las rutas está asociada a la carga en cada punto del recorrido y, como se ilustra en la expresión (4), para calcular este valor es necesario tener en cuenta todos los clientes presentes en la misma. En este trabajo se empleó una simplificación del concepto de factibilidad para la construcción de las soluciones golosas – aleatorias.

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 satisfacen las expresiones (7) y (8).*

$$\sum_{i=1}^{l} d_{r_i} \leq Q^k \tag{7}$$

$$\sum_{i=1}^{l} p_{r_i} \leq Q^k \tag{8}$$

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

La estrategia golosa – aleatoria (Algoritmo 2) se basa en adicionar clientes a cada ruta mientras se conserve la condición de factibilidad débil. La lista restringida de candidatos (RCL por sus siglas en inglés) se construye tomando como punto de referencia el último cliente añadido. Si no es posible realizar ninguna inserción que sea débil factible entonces se inicializa una nueva ruta y en este caso se toma como punto de referencia al depósito. Un cliente r se incluye en la RCL si el costo asociado a su inserción se encuentra en el intervalo $[c_{min}, c_{min} + \alpha(c_{max} - c_{min})]$ donde c_{min}, c_{max} son, respectivamente, el menor y el mayor costo de inserción. La calidad de la solución golosa – aleatoria depende del valor del parámetro α seleccionado. Si $\alpha = 0$ la solución es completamente golosa y si $\alpha = 1$ es completamente aleatoria.

2.2 Variantes de GRASP

El éxito de GRASP depende en gran medida de la capacidad de generar soluciones diferentes que sirvan de punto de partida a la búsqueda iterativa. Dado que no es posible

Algoritmo 2 Solución Golosa – Aleatoria

Clientes por añadir a la solución $A = I$
Inicializar la solución $\rightarrow S$
Último cliente añadido $\rightarrow c = 0$
Mercancía para entregar $D = 0$
Mercancía por recoger $P = 0$
while $|A| > 0$ **do**
  Construir la lista restringida de candidatos a partir de c considerando el factor de aleatoriedad $\alpha \rightarrow RCL$
  Seleccionar aleatoriamente un cliente de $RCL \rightarrow r$
  if $D + d_r \leq Q^k \wedge P + p_r \leq Q^k$ **then**
    Adicionar r a la ruta actual
  else
    Añadir la ruta actual a la solución S
    Crear una ruta nueva, $D = 0, P = 0$
    Añadir r a la ruta nueva
  end if
  $D = D + d_r$
  $P = P + p_r$
  Eliminar r de A
end while
return S

conocer de antemano el valor ideal para el parámetro α, se realizaron algunas modificaciones al procedimiento clásico de GRASP para lograr una mayor exploración del espacio de búsqueda con el objetivo de mejorar la calidad de las soluciones obtenidas por la metaheurística.

El procedimiento 3 propone una selección más adecuada del parámetro α, de forma tal que si se obtienen soluciones iniciales muy cercanas se aumente el valor de α para lograr una mayor diversidad y por el contrario si las soluciones son muy lejanas se reduzca α para intensificar la búsqueda.

La desventaja fundamental de los procedimientos 1 y 3 es que no se aprovechan, en las iteraciones sucesivas, la información de los óptimos locales previamente obtenidos. En el procedimiento 4 se presenta un enfoque en el cual, en lugar de crear una nueva solución en cada iteración se incluye un mecanismo de ruptura que modifica aleatoriamente un fragmento de la solución con el objetivo de escapar del óptimo local.

Se consideraron tres mecanismos de perturbación: eliminar 10 por ciento de los clientes y reinsertarlos nuevamente en posiciones aleatorias, invertir una ruta y realizar un cruzamiento aleatorio entre dos rutas diferentes.

En el procedimiento desarrollado, en cada paso se seleccionó aleatoriamente uno de los métodos de perturbación propuestos.

2.3 Búsqueda Local

En cada iteración de GRASP para realizar la búsqueda local se empleó la metaheurística Búsqueda por Entornos Variables Descendentes. VND es un enfoque determinista cuyo principio fundamental es realizar una exploración exhaustiva del entorno (Algoritmo 5).

Algoritmo 3 GRASP Reactivo

Número de iteraciones para hacer el ajuste de $\alpha \rightarrow T$
for $k = 1$ **to** K **do**
  if i mód $T = 0$ **then**
    Calcular la desviación estándar $\rightarrow E_T$ y el promedio P_T de las T últimas soluciones iniciales construidas y modificar consecuentemente el valor de α donde $\lambda \in [0,1]$ y $\varepsilon \in [0,1]$ son valores prefijados
    if Si $\frac{E_T}{P_T} > \varepsilon$ **then**
      $\alpha = \alpha - \lambda$
    else
      $\alpha = \alpha + \lambda$
    end if
  end if
  Construir una solución golosa-aleatoria S_k considerando el valor de α previamente calculado
  Realizar una búsqueda local empleando VND a partir de $S_k \rightarrow O_k$
end for
return $\min\{O_i\}_{i=1,\ldots,K}$

Algoritmo 4 GRASP con Pertubación

Construir una solución golosa-aleatoria O_0
for $k = 1$ **to** K **do**
  Perturbar el óptimo local previamente obtenido $O_{k-1} \rightarrow S_k$
  Realizar una búsqueda local empleando VND a partir de $S_k \rightarrow O_k$
end for
return $\min\{O_i\}_{i=1,\ldots,K}$

Se consideraron cuatro funciones de vecindad: mover un cliente dentro de una misma ruta, intercambiar dos clientes dentro de la misma ruta, mover un cliente de una ruta hacia otra diferente e intercambiar dos clientes de dos rutas diferentes.

2.4 Reordenamiento exacto de las rutas

El elevado número de variables y restricciones presentes en los modelos de programación matemática para el VRPSPD hacen prácticamente imposible su solución mediante procedimientos exhaustivos. A pesar de que en la literatura se presentan algunas estrategias exactas solo han sido aplicadas a problemas de pequeñas dimensiones. Sin embargo, al considerar solamente una ruta, disminuye notablemente la complejidad del modelo, por lo que puede ser resuelto de forma exacta con relativamente poco esfuerzo computacional.

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 la metaheurística.

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

Algoritmo 5 VND

```
Dada una solución inicial S_0
Dadas {V_i} estructuras de entornos
while i ≤ I do
    Explorar la vecindad V_i((S_{k-1}) hasta encontrar el primer elemento que mejore la solución S_{k-1} → S_k
    if S_k = ∅ then
        i = i + 1
    else
        i = 0
        S_{k+1} = S_k
    end if
end while
return S_k
```

de los clientes de la ruta, $J^+ = J \cup \{0\}$ y Q la capacidad del vehículo.

Variables de decisión

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

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:

$$\text{mín} \sum_{i,j \in J^+} x_{ij} c_{ij} \tag{9}$$

Restricciones:

$$\sum_{i \in J^+} x_{ij} = 1 \quad \forall j \in J \tag{10}$$

$$\sum_{i \in J^+} x_{ij} - \sum_{i \in J^+} x_{ji} = 0 \quad \forall j \in J \tag{11}$$

$$D_{0j} = \sum_{i \in J} d_i \tag{12}$$

$$P_{i0} = \sum_{i \in J} p_i \tag{13}$$

$$\sum_{i \in J^+} D_{ij} - \sum_{i \in J^+} D_{ji} = d_j \quad \forall j \in J \tag{14}$$

$$\sum_{i \in J^+} P_{ji} - \sum_{i \in J^+} P_{ij} = p_j \quad \forall j \in J \tag{15}$$

$$D_{ij} + P_{ij} \leq x_{ij} Q \quad \forall i, j \in J^+ \tag{16}$$

$$x_{ij} \in \{0,1\} \quad \forall i, j \in J^+ \tag{17}$$

$$D_{ij} \geq 0 \quad \forall i, j \in J^+ \tag{18}$$

$$P_{ij} \geq 0 \quad \forall i, j \in J^+ \tag{19}$$

La función objetivo es minimizar el costo de viaje. La restricción (10) es relativa inclusión de todos los clientes y (11) se refiere a la continuidad del recorrido a partir de un punto. Con (12) y (13) se asegura que toda la mercancía para entregar se cargue en el depósito al cual se regresa con toda la mercancía recogida. (14) - (15) garantizan que se satisfagan las demandas de cada cliente. La restricción (16) corresponde al control de la carga en cada punto del recorrido. (17)-(19) se refieren a las cotas de las variables.

El modelo de programación matemática descrito fue resuelto con la versión académica de GAMS [13].

3. Experimentación y Resultados

Los procedimientos presentados en la sección 2 fueron evaluados empleando el conjunto de prueba propuesto por Salhi y Nagy en 1999 [14]. Este juego de datos consta de 26 problemas euclidianos generados a partir de instancias del problema de enrutamiento de vehículos con restricciones de capacidad. Estos problemas han sido ampliamente utilizados en la literatura para la comparación de resultados [3], [8], [12], [15], [16], [20] y [19]. En la tabla 1 se describen las dimensiones de los problemas del conjunto de prueba y la mejor solución reportada para cada uno de ellos.

Se realizaron cuatro bloques de experimentos con un subconjunto de las instancias de prueba para analizar la influencia del factor de aleatoriedad de GRASP (α) así como del factor de penalidad de la función objetivo (μ). En todos los casos, las rutas de la solución obtenida por la metaheurística se ordenaron de forma exacta utilizando el modelo de programación matemática propuesto en 2.4.

GRASP Clásico Se consideró $\alpha = 0{,}2$, $\alpha = 0{,}5$ y $\alpha = 0{,}8$. Para cada uno de ellos por separado, se analizó la influencia del factor de penalidad: sin penalidad $\mu = 0$, penalidad baja $\mu = 10$ y penalidad alta $\mu = 50$.

GRASP Reactivo Se consideraron tres valores iniciales para el factor de aleatoriedad: $\alpha = 0{,}2$, $\alpha = 0{,}5$ y $\alpha = 0{,}8$ y dos valores para el factor de penalidad: $\mu = 0$ y $\mu = 10$. Cada 10 iteraciones se reajustó el valor de α y en caso de no haberse obtenido ninguna solución factible se aumentó el valor de μ en 10 unidades.

GRASP con Perturbación 1 Se consideraron tres valores iniciales para el factor de aleatoriedad: $\alpha = 0{,}2$, $\alpha = 0{,}5$ y $\alpha = 0{,}8$ y dos valores para el factor de penalidad: $\mu = 0$ y $\mu = 10$. En cada iteración se seleccionó aleatoriamente el método de perturbación a emplear y si el óptimo local encontrado no es factible se aumentó en 10 unidades el factor de penalidad.

GRASP con Perturbación 2 La diferencia con el método *GRASP con Perturbación 1* es que no se aumentó el valor μ en caso de no encontrarse soluciones factibles. Se consideraron dos valores para el factor de penalidad: $\mu = 10$ y $\mu = 50$.

En la tabla 2 se resumen los mejores resultados obtenidos en cada uno los experimentos desarrollados. Se resalta en negritas la mejor solución para la instancia y subrayada la mejor solución obtenida en este trabajo.

Tabla 1. Descripción de las instancias de prueba

Problema	Clientes	V	Costo	R	Referencia
CMT1X	50	6	466.77	3	Subramanian et al. [15]
CMT1Y	50	5	466.77	3	Subramanian et al. [15]
CMT2X	75	10	682.39	6	Zachariadis et al. [19]
CMT2Y	75	12	679.44	6	Chen et al. [3]
CMT3X	100	10	719.06	5	Zachariadis et al. [19]
CMT3Y	100	10	719.00	5	Tang-Montané et al. [16]
CMT4X	150	15	852.46	7	Zachariadis et al. [20]
CMT4Y	150	15	852.35	7	Chen et al. [3]
CMT5X	199	19	1030.56	10	Zachariadis et al. [20]
CMT5Y	199	19	1030.56	10	Zachariadis et al. [20]
CMT6X	50	6	476.00	3	Tang-Montané et al. [16]
CMT6Y	50	6	474.00	3	Tang-Montané et al. [16]
CMT8X	100	10	720.00	5	Tang-Montané et al. [16]
CMT8Y	100	10	721.00	5	Tang-Montané et al. [16]
CMT9X	150	15	885.00	7	Tang-Montané et al. [16]
CMT9Y	150	15	900.00	8	Tang-Montané et al. [16]
CMT10X	199	20	1100.00	11	Tang-Montané et al. [16]
CMT10Y	199	21	1083.00	11	Tang-Montané et al. [16]
CMT11X	120	11	831.09	4	Zachariadis et al. [19]
CMT11Y	120	11	829.85	4	Zachariadis et al. [19]
CMT12X	100	10	658.83	5	Zachariadis et al. [19]
CMT12Y	100	11	660.47	5	Zachariadis et al. [19]
CMT13X	120	13	918.00	5	Tang-Montané et al. [16]
CMT13Y	120	12	910.00	5	Tang-Montané et al. [16]
CMT14X	100	13	675.00	5	Tang-Montané et al. [16]
CMT14Y	100	12	689.00	5	Tang-Montané et al. [16]

Los experimentos realizados muestran que los refinamientos aplicados a la estrategia GRASP se reflejan en una mejoría en la calidad de las soluciones. Al reodenar las rutas de los óptimos locales previamente obtenidos no se observó un gran decrecimiento en el valor de la función objetivo, lo que apunta hacia la efectividad de las estrategias intra - ruta, de ahí la necesidad de incluir movimientos que exploren más en el espacio inter - ruta.

En *GRASP Clásico* no se obtuvieron soluciones factibles para $\mu = 0$. Como promedio el modelo de programación matemática mejoró en un dos por ciento la solución obtenida por la metaheurística.

En *GRASP Reactivo* el mejor rendimiento se obtuvo para $\alpha = 0{,}5$. Como promedio el modelo de programación matemática mejoró en un dos por ciento la solución obtenida por la metaheurística.

En *GRASP con Perturbación 1* como promedio el modelo de programación matemática mejoró en un uno por ciento la solución obtenida por la metaheurística.

En *GRASP con Perturbación 2* como promedio el modelo de programación matemática mejoró en un 0,9 por ciento la solución obtenida por la metaheurística.

En general, las variantes de GRASP con Perturbación fueron los métodos con mejor desempeño. Para las restantes instancias del conjunto de prueba empleado se realizaron dos bloques de experimentos. El primero con las mismas características descritas para los experimentos GRASP con Perturbación 1 y 2. En el segundo se aumentó el número de iteraciones para favorecer una mayor exploración del espacio de búsqueda. En la tabla 3 aparecen los mejores resultados obtenidos empleando estos procedimientos. *GRASP 15* se refiere a 15 iteraciones de GRASP y *GRASP 30* a 30 iteraciones.

En ninguno de los problemas de grandes dimensiones fue posible mejorar la solución reportada en la literatura. Al aumentar el número de iteraciones se obtuvo, como promedio, una solución 2,30 por ciento mejor.

En la tabla 4 se resumen los mejores resultados obtenidos por la estrategia propuesta en este trabajo (GG) y se comparan con los reportados en la literatura. Se mejoraron cuatro de las soluciones reportadas: CMT6X, CMT6Y, CMT13X y CMT13Y y se igualó una CMT1Y. En general se obtuvieron soluciones con menos de un 5 por ciento de gap con la literatura para 18 problemas.

Conclusiones

En este trabajo se presentaron diferentes variantes del método GRASP aplicadas al problema de enrutamiento de vehículo con recogida y entrega simultánea. En general, los procedimientos clásico y reactivo mostraron resultados similares entre sí. Las mejores soluciones se obtuvieron por las estrategia de pertubación, en la cuales se remplaza la creación

Tabla 2. Variantes de GRASP

		Clásico		Reactivo		Perturbación 1		Perturbación 2	
Problema	**Literatura**	**GRASP**	**GAMS**	**GRASP**	**GAMS**	**GRASP**	**GAMS**	**GRASP**	**GAMS**
CMT1X	**466.77**	492.06	480.07	478.00	471.53	475.10	473.03	467.79	467.79
CMT1Y	**466.77**	479.20	478.69	474.38	467.79	472.77	**466.75**	472.87	472.36
CMT2X	**682.39**	742.42	735.71	728.47	723.35	715.48	714.89	713.97	713.97
CMT2Y	**679.44**	730.61	723.36	724.68	722.80	718.26	717.56	708.34	*708.34*
CMT3X	**719.06**	780.32	749.57	756.55	735.28	748.94	744.36	746.03	738.15
CMT3Y	**719.00**	770.87	757.78	763.83	738.15	746.38	740.76	752.40	745.29
CMT6X	476.00	485.79	473.28	480.01	472.36	473.62	**467.79**	475.71	475.71
CMT6Y	474.00	486.88	478.55	477.22	474.38	479.95	472.36	473.86	**471.53**
CMT8X	**720.00**	778.38	762.31	754.21	735.35	746.10	743.76	739.90	739.90
CMT8Y	**721.00**	770.81	752.19	756.25	745.54	730.50	730.50	736.74	736.74
CMT11X	**831.09**	999.98	938.70	977.01	920.60	917.80	901.37	933.56	908.62
CMT11Y	**829.85**	951.11	922.89	950.87	930.82	927.85	906.45	911.82	908.95
CMT12X	**658.83**	745.21	732.10	740.98	729.91	689.78	683.96	696.21	681.67
CMT12Y	**660.47**	748.84	744.00	730.20	705.88	684.19	675.19	692.76	686.53
CMT13X	918.00	965.96	953.14	973.60	911.35	917.14	904.86	897.85	**880.08**
CMT13Y	910.00	920.42	896.53	923.11	891.47	902.41	**882.15**	907.65	896.80
CMT14X	**675.00**	701.87	696.92	693.07	684.63	692.87	688.57	680.74	676.04
CMT14Y	**689.00**	714.69	708.35	736.02	721.50	706.98	693.37	698.66	695.09

Tabla 3. GRASP con Perturbación para los problemas de grandes dimensiones

		GRASP con Pertutbación 1				GRASP con Pertubación 2			
		GRASP 15		GRASP 30		GRASP 15		GRASP 30	
Problema	**Literatura**	**GRASP**	**GAMS**	**GRASP**	**GAMS**	**GRASP**	**GAMS**	**GRASP**	**GAMS**
CMY4X	**852.46**	945.83	930.46	909.98	900.21	1000.06	961.80	941.00	932.18
CMT4Y	**852.35**	953.65	939.18	950.41	924.68	936.80	931.25	925.58	922.00
CMT5X	**1030.56**	1169.21	1150.60	1167.23	1152.00	1232.48	1204.93	1168.35	1145.33
CMT5Y	**1030.56**	1185.44	1164.78	1168.77	1159.59	1191.62	1173.00	1179.91	1172.77
CMT9X	**885.00**	971.79	950.15	958.30	937.17	962.04	948.02	901.01	896.07
CMT9Y	**900.00**	970.10	961.08	926.57	903.00	975.66	957.98	960.59	940.29
CMT10X	**1100.00**	1227.65	1205.63	1190.36	1177.90	1205.61	1191.58	1222.31	1212.81
CMT10Y	**1083.00**	1211.78	1196.89	1181.63	1150.63	1172.49	1168.32	1155.34	1148.24

de nuevas soluciones por un criterio de ruptura que permite aprovechar parte de la estructura del óptimo local encontrado en el paso anterior. Las rutas de las mejores soluciones obtenidas por el procedimiento GRASP se ordenaron de forma exacta utilizando el modelo de programación matemática resuelto con la versión académica de GAMS. Con la estrategia presentada se mejoraron las soluciones de cuatro instancias del conjunto de prueba de Salhi y Nagy (CMT6X, CMT6Y, CMT13X y CMT13Y).

En general no se apreció una diferencia significativa entre las soluciones reportadas por GRASP y las obtenidas después de aplicar GAMS, lo que apunta hacia la efectividad de la metaheurístca en la ubicación de los clientes dentro de las rutas. De ahí la necesidad de incluir nuevos movimientos inter-ruta que garanticen una mayor diversidad. Otro elemento a considerar para futuras investigaciones es la sustitución de la búsqueda determinista (VND) en cada paso de GRAPS por un enfoque aleatorio que permita una mayor exploración del espacio de búsqueda y evite quedar atrapados en óptimos locales.

Agradecimientos

Las autoras quieren agradecer al Grupo de Optimización de la Facultad de Matemática y Computación por la colaboración en el desarrollo de este trabajo. A Oscar Luis por su ayuda en el diseño de los algoritmos y en la ejecución de los experimentos. A los revisores anónimos por las sugerencias realizadas.

Referencias

[1] SP Anbuudayasankar and K Mohandas. 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, 2007.

[2] R Baldacci, M Batarra, and D Vigo. *The vehicle routing problem: latest advances and new challenges*, chapter

Tabla 4. Comparación con la literatura

Problema	Literatura	GG	Problema	Literatura	GG	Problema	Literatura	GG
CMT1X	**466.77**	467.79	CMT5Y	**1030.56**	1159.59	CMT11X	**831.09**	901.37
CMT1Y	**466.77**	**466.75**	CMT6X	476.00	**467.79**	CMT11Y	**829.85**	906.45
CMT2X	**682.39**	713.97	CMT6Y	474.00	**471.53**	CMT12X	**658.83**	681.67
CMT2Y	**679.44**	708.34	CMT8X	**720.00**	735.35	CMT12Y	**660.47**	675.19
CMT3X	**719.06**	735.28	CMT8Y	**721.00**	730.50	CMT13X	918.00	**880.08**
CMT3Y	**719.00**	738.15	CMT9X	**885.00**	896.07	CMT13Y	910.00	**882.15**
CMT4X	**852.46**	900.21	CMT9Y	**900.00**	903.00	CMT14X	**675.00**	676.04
CMT4Y	**852.35**	922.00	CMT10X	**1100.00**	1177.90	CMT14Y	**689.00**	693.37
CMT5X	**1030.56**	1145.33	CMT10Y	**1083.00**	1148.24			

Routing heterogeneous fleet of vehicles, pages 3–28. Springer, 2008.

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

[4] M Dell'Amico, G Righini, and M Salani. A branch-and-price approach to the vehicle routing problem with simultaneous distribution and collection. *Transportation Science*, 40(2):235–247, 2006.

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

[6] J Dréo, A Pétrowski, and E Taillard. *Metaheuristics for Hard Optimization*. Springer, 2006.

[7] A Fernández-Arias. 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, 2010.

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

[9] P Hansen and N Mladenovic. Variable neighborhood search: Principles and applications. *European Journal of Operational Research*, (130):449–467, 2001.

[10] P Hansen, N Mladenovic, and JA Moreno-Pérez. Búsqueda de entorno variable. *Revista Iberoamericana de Inteligencia Artificial*, (19):72–92, 2003.

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

[12] O Polat, Kalayci C-B, O Kulak, and H-O Gunther. 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, 2015.

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

[14] S Salhi and G Nagy. A cluster insertion heuristic for the single and multiple depot vehicle routing problem with backhauling. *The Journal of the Operational Research Society*, 50(10):1034–1042, 1999.

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

[16] F-A Tang-Montané and R-D Galvao. A tabu search algorithm for the vehicle routing problem with simultaneous pick-up and delivery service. *Computers and Operations Reseach*, 33:595–619, 2006.

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

[18] P Toth and D Vigo. The vehicle routing problem. *SIAM, Monograph on Discrete Mathematics and Applications*, 9, 2002.

[19] E Zachariadis, C Tarantilis, and C Kiranoudis. A hybrid metaheuristic algorithm for the vehicle routing problem with simultaneous delivery and pick-up service. *Expert System with Applications*, 36:1070–1081, 2009.

[20] E Zachariadis, C Tarantilis, and C Kiranoudis. An adaptive memory methodology for the vehicle routing problem with simultaneous delivery and pick-up service. *Expert System with Applications*, 36:1070–1081, 2010.