Comparación de un Sistema de Colonias de Hormigas y una Estrategia Evolutiva para el Problema del Ruteo de Vehículos con Ventanas de Tiempo en un Contexto Multiobjetivo

Benjamín Barán y Augusto Hermosilla

Centro Nacional de Computación, Universidad Nacional de Asunción
San Lorenzo, Casilla de Correos 1439 - Paraguay
{bbaran, ahermosilla}@cnc.una.py
http://www.cnc.una.py

Resumen. Este trabajo compara un Sistema de Optimización basado en Colonias de Hormigas (*Ant Colony Optimization*) con una estrategia evolutiva (variante del *Pareto Archived Evolutionary Strategy*), utilizados en la resolución multiobjetivo del problema de ruteo de vehículos con ventanas de tiempo. (*Vehicle Routing Problem with Time Windows*).

1 Introducción

El problema del ruteo de vehículos (*Vehicle Routing Problem*, o VRP) es ya considerado un paradigma en la literatura especializada [1]. Este problema se plantea como un depósito central que cuenta con una flota de vehículos y debe atender a un conjunto de clientes geográficamente distribuidos. El objetivo del VRP es entregar bienes a este conjunto de clientes con demandas conocidas, al mínimo coste, encontrando las rutas óptimas que se originan y terminan en el referido depósito. Cada cliente es servido una sola vez y todos los clientes deben ser atendidos, para lo cual se los asigna a vehículos que llevarán la carga (demanda de los clientes que visitará) sin exceder su capacidad.

Para extender el VRP tradicional agregando la restricción adicional de asociar una ventana de tiempo a cada cliente, se define un intervalo en el que cada cliente debe ser atendido y se obtiene el problema del ruteo de vehículos con ventanas de tiempo (*Vehicle Routing Problem with Time Windows*, o VRPTW) [2]. Al considerar estas ventanas de tiempo, el costo total de ruteo y planeamiento (*scheduling*) incluyen: la distancia total recorrida que está asociada al tiempo total de viaje efectivo, el tiempo total de espera en que se incurre cuando un vehículo llega muy temprano a la ubicación de un cliente y el tiempo total de servicio (tiempo para descargar todas las mercaderías solicitadas por cada cliente). Claramente, el tiempo en el que se inicia el servicio a un cliente debe ser mayor o igual al inicio de su ventana de tiempo y el instante en que se llega a cada cliente debe ser menor o igual al fin

de su ventana de tiempo. Si un vehículo llega a la ubicación de un cliente antes del inicio de su ventana de tiempo, debe esperar hasta esa hora para servir a ese cliente.

El problema espacial del ruteo de vehículos ha sido extensamente estudiado en la literatura. Una gran variedad de abordajes han sido ya publicados para resolver este problema, utilizando implementaciones paralelas [3], estrategias híbridas combinando métodos de búsqueda local con algoritmos evolutivos [4], redes neuronales [5, 6], una heurística que utiliza información de feromonas [7] y trabajos que resuelven este problema con estrategias evolutivas para optimizar la demanda de cada vehículo además de la optimización del número de vehículos y el tiempo total de viaje [8].

La restricción de ventanas de tiempo solo ha sido considerada recientemente y varios abordajes han sido presentados para resolver el VRPTW: algoritmos paralelos con un número polinomial de procesadores [9], algoritmos genéticos [10, 11, 12, 13], recocido simulado paralelo [14, 15] y colonias múltiples de hormigas [16]. Estos trabajos analizan el problema en cuestión en un contexto mono-objetivo, aunque en algunos casos se proponen priorizaciones y/o combinaciones de objetivos, sin llegar a calcular el conjunto completo de soluciones Pareto. Estudios de las principales heurísticas y metaheurísticas pueden ser encontrados, por ejemplo, en [17] y [18].

Por su parte, el presente trabajo resuelve el problema del VRPTW en un contexto multiobjetivo, considerando dos técnicas evolutivas que han demostrado su capacidad de resolver adecuadamente problemas de similar complejidad:

(1) las optimizaciones con colonias de hormigas (ACO), en la versión conocida como MOACS-VRPTW, propuesta por Barán y Schaerer [19], a partir de una optimización del MACS-VRPTW publicado por Gambardella et al. [16]; y

(2) los algoritmos evolutivos multiobjetivos (*Multi-Objective Evolutionary Algorithms* – MOEAs) en una variante propuesta en este trabajo que daremos en llamar PAES-DRI, obtenida a partir de la propuesta ES-DRI presentada por Mester [21].

A conocimiento de los autores, esta es la primera comparación experimental entre un ACO y un MOEA para el problema del VRPTW en un contexto multiobjetivo, en el que los distintos objetivos tienen la misma prioridad [22].

El resto del trabajo es organizado de la siguiente manera: la sección 2 presenta la formulación tradicional del problema, la sección 3 contiene la formulación multiobjetivo, la sección 4 resume el MOACS-VRPTW, la 5 explica el PAES-DRI, la 6 presenta las métricas de desempeño utilizadas para la comparación, mientras la sección 7 presenta resultados experimentales. Finalmente, la sección 8 presenta las conclusiones del trabajo.

2 Formulación Matemática del VRPTW

El VRPTW es un problema combinatorial que puede ser formulado matemáticamente como un grafo dirigido ***G(V, A)***. El problema considera un conjunto de vértices $V = \{v_0, v_1, \ldots, v_n\}$, donde v_0 representa el depósito, mientras que v_i *(i = 1, ..., n)* representa a cada uno de los *n* clientes (o ciudades) a ser visitados. Por su parte, el conjunto de arcos está dado por: $A = \{(v_i, v_j) / v_i, v_j \in V; i \neq j\}$.

Por su parte, $C = \{ c_{ij} \} \in \Re^{(n+1) \times (n+1)}$ es una matriz de distancias o costos no negativos entre cada par de vértices v_i y v_j (incluyendo depósito y clientes).

d es el vector de las demandas de los clientes y q_i representa la cantidad de bienes requeridos por el cliente v_i.

m representa el número de vehículos, que se asumen todos idénticos, y con una capacidad *Q* por vehículo. A cada vehículo *i* se le asignará una ruta R_i.

$[e_i, l_i]$ representa la ventana de tiempo del cliente v_i; con e_i como la hora más temprana y l_i como la hora más tardía en que se puede iniciar el servicio a dicho cliente v_i. Por su parte, θ_i representa el tiempo requerido en descargar la cantidad de mercaderías q_i en el cliente v_i.

Como el problema aquí considerado es simétrico $c_{ij} = c_{ji}$ para cada $(v_i, v_j) \in A$ y es común reemplazar *A* por el conjunto $E = \{(v_i, v_j) | v_i, v_j \in V; i < j\}$. Sin pérdida de generalidad, y por simplicidad, se asume que el tiempo t_{ij} necesario para ir de v_i a v_j es igual a la distancia c_{ij} (se asume velocidad constante e unitaria para los vehículos). El intervalo $[e_0, l_0]$ en el depósito se llama horizonte de planeación (*scheduling horizon*).

Definiendo b_v como el instante de inicio del servicio al cliente v, se puede escribir la condición de factibilidad que debe cumplir una ruta $R_i = \{v_0^i, v_1^i, \ldots, v_{k_i}^i, v_{k_i+1}^i\}$, donde $v_j^i \in V$, $v_0^i = v_{k_i+1}^i = 0$ (el 0 denota el depósito) y k_i es la cantidad de clientes atendidos en la ruta i:

$$\sum_{j=1}^{k_i} q_j^i \leq Q. \qquad \textbf{(1)}$$

$$e_{v_j^i} \leq b_{v_j^i} \leq l_{v_j^i}. \qquad \textbf{(2)}$$

$$b_{v_{k_i}^i} + \omega_{v_{k_i}^i} + c_{v_{k_i}^i,0} \leq l_0, \qquad \text{con } 1 \leq j \leq k. \tag{3}$$

Asumiendo que todo vehículo viaja al próximo cliente una vez que haya terminado de servir al cliente actual, $b_{v_j^i}$ puede ser calculado recursivamente de la siguiente manera:

$$b_{v_j^i} = \max\left\{e_{v_j^i}; b_{v_{j-1}^i} + \omega_{v_{j-1}^i} + c_{v_{j-1}^i,v_j^i}\right\}, \qquad \text{con } b_0^i = e_0 \text{ y } \omega_0^i = 0. \tag{4}$$

Así, el tiempo de espera en el cliente v_j^i puede ser definido como:

$$w_{v_j^i} = \max\left\{0; b_{v_j^i} - b_{v_{j-1}^i} - \omega_{v_{j-1}^i} - c_{v_{j-1}^i,v_j^i}\right\}. \tag{5}$$

El costo de la ruta R_i puede estar dado por la duración total de la misma, en cuyo caso estaría definido como:

$$T(R_i) = \sum_{j=0}^{k_i} c_{v_j^i,v_{j+1}^i} + \sum_{j=1}^{k_i} \omega_{v_j^i} + \sum_{j=0}^{k_i} w_{v_j^i}. \tag{6}$$

Tradicionalmente, el objetivo primario es la minimización del tamaño *m* de la flota de vehículos y el objetivo secundario es la minimización de la distancia total recorrida (tiempo total de viaje) o la duración total de las rutas (tiempo total de entrega), [17, 18].

3 Formulación del VRPTW como un Problema Multiobjetivo

Como ya fuera discutido, el VRPTW es tradicionalmente formulado como un problema bi-objetivo lexicográfico. En contrapartida, este trabajo utiliza una formulación multiobjetivo presentada en [19]. En este contexto, la función objetivo es un vector tridimensional $F = [F_1, F_2, F_3]^T$, y ningún objetivo es más importante que los otros. Los tres objetivos utilizados son:

- $F_1 = m$ ... es el número de vehículos (o tamaño de la flota);
- $F_2 = \sum_{i=1}^{m} \sum_{j=0}^{k_i} c_{v_j^i,v_{j+1}^i}$ es el tiempo total de viaje (o distancia total recorrida);

- $F_3 = F_2 + \sum_{i=1}^{m}\sum_{j=1}^{k_i} o_{v_j^i} + \sum_{i=1}^{m}\sum_{j=0}^{k_i} w_{v_j^i}$ ... es el tiempo total de entrega.

donde v_j^i denota el cliente servido en el *j*-ésimo orden de la ruta R_i y k_i representa el número total de clientes servidos en dicha ruta R_i.

Dado que ningún objetivo se considera más importante que los demás, se trata de un contexto realmente multiobjetivo donde en principio, la solución puede no ser única, sino un conjunto de soluciones óptimas, no comparables entre sí, conocido como *Conjunto Pareto P*. La imagen de *P* en el contra-dominio de las funciones objetivo es conocida como Frente Pareto *FP* [22].

4 Algoritmo MOACS-VRPTW

Barán y Schaerer [19] propusieron una variación del Algoritmo de Colonias Múltiples de Hormigas para el VRPTW (MACS-VRPTW), originalmente propuesto por Gambardella et al. [16]. Esta variación fue denominada Colonia de Hormigas para Optimización Multiobjetivo del VRPTW (MOACS-VRPTW). Esta metaheurística utiliza una sola colonia de hormigas para minimizar simultáneamente las tres funciones objetivo, presentadas en la sección anterior. Todas las funciones comparten los mismos rastros de feromonas. De esta manera, el conocimiento de buenas soluciones es igualmente importante para cada función objetivo. El referido MOACS-VRPTW puede ser resumido con el siguiente pseudocódigo:

```
Pseudocode MOACS-VRPTW
/* Inicialización */
Repeat /* Loop principal */
 for each ant k ∈ {1,…, m} do
    Construir una solución (π^k)
    if π^k ∈ P  /* si solución encontrada forma parte del
                   conjunto Pareto */
      grabar π^k y borrar soluciones dominadas de P
    end if
 end for
 /* Actualizar feromonas τ */
 if τ0' > τ0
    /* Un mejor FP ha sido encontrado */
    Reinicializar rastros {τ} con τ0
 else
    for each π^P ∈ P do
       Actualización global de feromonas
    end for
 end if
until criterio de parada ha sido satisfecho.
```

5 Algoritmo PAES-DRI

El presente trabajo, propone una nueva variante de MOEA que damos en llamar PAES-DRI, inspirada en el ES-DRI originalmente propuesto por Mester [20]. El ES-DRI (*Evolutionary Strategy - Dichotomy Remove-Insert*) es un algoritmo heurístico muy efectivo para determinar la solución de versiones en gran escala del VRPTW. Este algoritmo esta basado en las ideas de estrategias evolutivas (1+1) y algunos operadores de mutación que trabajan con un esquema dicotómico de remover-insertar. Por su parte, el PAES (*Pareto Archived Evolutionary Strategy*) [23] es un esquema simple de evolución para problemas multiobjetivo. Este algoritmo es una estrategia evolutiva (1+1) que utiliza búsqueda local en una población de un individuo, pero usando un archivo de referencia de soluciones previamente encontradas a fin de identificar el ranking de dominancia aproximado de las soluciones *actual* y *candidata*. El PAES-DRI puede ser resumido con el siguiente pseudocódigo:

```
Pseudocode PAES-DRI
/* Inicialización.  */
Obtener una solución actual
Inicializar aleatoriamente R con 1 o 2
Inicializar β con la ecuación (8)
actual ← inicial    /* Inicializar solución actual */
Repeat /* Loop principal */
   candidata ← mutar(actual, R, β)  /*obtener candidata mutando
                          solución actual */
   If actual ≻ candidata /* si actual es mejor */
      candidata ← actual
   End if
   candidata ← optimización_local(candidata)
   If actual ≻ candidata /* si actual es mejor */
      Descartar candidata
      Cambiar aleatoriamente parámetros mutación
   Else if candidata ≥ actual /* candidata no es peor */
            actual ← candidata
            Agregar candidata a conjunto Pareto P
            Eliminar soluciones dominadas de P
         else /* candidata es dominada por algún miembro del
                     conjunto Pareto P */
            Descartar candidata
         endif
   Endif
Until un criterio de parada ha sido alcanzado.
```

Considerando que esta variante está siendo propuesta en este trabajo, se la describirá en mayores detalles en las siguientes subsecciones, de forma a conocer los elementos que conforman este algoritmo.

5.1 Solución inicial

La inicialización de la población es realizada utilizando una heurística basada en la inserción más económica [24]. Las rutas son construidas una a la vez en forma secuencial. Para esto, es necesario determinar los nodos que inicialicen las rutas. Estos nodos son denominados nodos *semillas* (*seed nodes*). Primero se determinan los cuatro nodos primarios (*PC*). El primer nodo es el cliente más lejano al depósito y los siguientes tres son determinados encontrando un nodo que se encuentre geográficamente lo más alejado posible de los clientes previamente seleccionados y del depósito. Luego, de forma similar se seleccionan cuatro nodos secundarios (*SC*) utilizando los nodos primarios. Como los nodos primarios forman un cuadrilátero, los nodos secundarios son los clientes más cercanos al punto medio de cada lado del mismo. Finalmente, se selecciona un conjunto *T* de los $\hat{n}$ clientes más alejados del depósito.

Una vez conformados los conjuntos *PC*, *SC* y *T*, se selecciona un cliente del nodo PC que esté más alejado del depósito, como el inicio de la primera ruta. El siguiente nodo *semilla* que inicialice la próxima ruta deberá ser seleccionado del conjunto $I = T \cup PC \cup SC$ de manera que sea el cliente no enrutado más cercano al primer nodo semilla. Finalmente, si no existen clientes no enrutados en *I*, el siguiente nodo semilla es el cliente no enrutado con el menor índice (los clientes son indexados de 1 a *n*, donde *n* es el número total de clientes).

Una vez determinado el nodo semilla de la ruta en formación, se selecciona el cliente no enrutado que minimiza la combinación ponderada del *detour* y el tiempo de espera y se lo incorpora en el mejor lugar factible de inserción. Para la inserción se consideran solamente los clientes más cercanos geográficamente a por lo menos uno de los clientes previamente insertados en la ruta. Se considera que un cliente es geográficamente cercano a una ruta, si la distancia del cliente a algún cliente previamente insertado en la ruta es menor o igual a una constante $\hat{d}$. De esta manera, después de cada inserción es necesario actualizar el conjunto de los nodos cercanos a la ruta parcial actual.

La función de costo de inserción de un cliente v_u está dada por:

$$C_u = \alpha_1 D_u + \alpha_2 W_u \quad \alpha_3 c_{0u} . \tag{7}$$

donde,

$$D_u = c_{iu} + c_{uj} - c_{ij} . \tag{7.1}$$

$$W_u = W_u^a - W_u^b . \tag{7.2}$$

$$\alpha_1 + \alpha_2 = 1; \qquad \alpha_3 > 0 . \tag{7.3}$$

Las distancias entre los correspondientes pares de clientes (v_i, v_u), (v_u, v_j) y (v_i, v_j) son denotados por c_{iu}, c_{uj} y c_{ij} respectivamente. Además W_u^a y

W_u^b corresponden el tiempo total de espera antes y después de la inserción, respectivamente.

5.2 Operador de mutación

Tal como se había mencionado con anterioridad, el PAES-DRI es una variación del ES-DRI [20]. El operador de mutación utilizado en dicho trabajo genera una nueva solución removiendo e insertando p clientes de la solución *actual* para obtener la solución *candidata*. Este procedimiento genera nuevas rutas factibles en un proceso de dos fases que construye y mejora nuevas soluciones. El número de clientes a ser removidos de la solución *actual* es determinado conforme a la ecuación:

$$p = (0.1 + 0.5rnd)n . \quad (\mathbf{8})$$

donde *n* es siempre el número total de clientes y *rnd* es un número generado aleatoriamente y distribuido uniformemente entre 0 y 1.

Los clientes removidos forman un conjunto denominado *reject*. Los clientes no removidos de la solución *actual* forman una solución parcial denominada *remnant*. Los clientes a ser removidos son determinados de dos maneras (el tipo de remoción está definido por la variable R, que es seleccionada aleatoriamente). La primera, consiste en remover p clientes en forma aleatoria. Para realizar este procedimiento asignamos a R el valor de 1. La segunda, que corresponde a $R = 2$, realiza remociones aleatorias de los clientes ubicados geográficamente en el interior de un anillo creado con dos círculos concéntricos con centro en el depósito y radios aleatorios.

Después de remover los clientes de la solución actual se procede a insertarlos en la solución *remnant* para obtener una solución *candidata* factible. La inserción se realiza en forma secuencial hasta que no queden nodos sin insertar. Se ordenan los nodos de *reject* en forma aleatoria y se los inserta uno por uno. Para insertar un cliente v_u se determinan todos los lugares factibles de inserción y se halla para cada uno de ellos el costo de inserción de acuerdo a la una función vectorial $[D_u, T_u]$. Para hallar D_u utilizamos la ecuación (7.1) y T_u es definido como sigue:

$$T_u = D_u + W_u . \quad (\mathbf{9})$$

donde W_u es calculada con la ecuación (7.2). Como la función de inserción tiene dos dimensiones, se determinan todas las opciones de inserción que no son dominadas y se selecciona una al azar. Cada $0.1n$ inserciones se realizan procedimientos de optimización local de la solución parcial obtenida. Para esto también se utiliza el criterio multiobjetivo arriba mencionado.

Si la solución *candidata,* obtenida con el operador de mutación, domina a la solución *actual*, los mismos parámetros de mutación son aplicados a la nueva solución *actual*. Si no, para generar una nueva solución *candidata,* el algoritmo utiliza otros parámetros de mutación. Este tipo de operador de mutación es de-

nominado "*Dichotomy Remove-Insert*". Después de la mutación, la solución *candidata* es mejorada con procedimientos combinatorios de complejidad $O(n^2)$ como por ejemplo:

(i) *Or-Opt* [25], o

(ii) *1-interexchange* [26].

En cuanto a la búsqueda local, se utiliza el criterio multiobjetivo explicado anteriormente para decidir si se acepta o no una nueva solución.

6 Métricas de desempeño

Para evaluar los resultados experimentales de los dos algoritmos arriba presentados, se seleccionaron tres métricas, considerando que no existe una única métrica que pueda por sí sola medir el desempeño, eficiencia y efectividad de los algoritmos evolutivos multiobjetivos [22]. Las métricas utilizadas para el presente trabajo son las siguientes:

- *Overall Non-dominated Vector Generation* (ONVG): simplemente cuenta el número de soluciones en el Frente de Pareto calculado, denotado como Y_{known}

$$ONVG \stackrel{\Delta}{=} |Y_{known}|_c. \quad (10)$$

donde $|\ |_c$ denota cardinalidad. A mayor valor del ONVG, se tiene un mejor conocimiento de los detalles del Frente de Pareto.

- *Overall True Non-dominated Vector Generation* (*OTNVG*): cuenta el número de soluciones en $\boldsymbol{Y}_{known}$ que también se encuentran en el Frente de Pareto Óptimo denotado como $\boldsymbol{Y}_{true}$. Como $\boldsymbol{Y}_{true}$ no es conocido en teoría, debe estimarse haciendo muchas corridas de diversos algoritmos multiobjetivos para el mismo problema, escogiendo las soluciones Pareto óptimas encontradas con la unión de los resultados obtenidos con todos los experimentos. Claramente, un mayor valor de OTNVG, indica que un conjunto $\boldsymbol{Y}_{known}$ es mejor que otro.

$$OTNVG \stackrel{\Delta}{=} \left|\left\{y \mid y \in Y_{known} \wedge y \in Y_{true}\right\}\right|_c. \quad (11)$$

- *Coverage* (C): cuenta el número de soluciones del Frente de Pareto de un algoritmo 1, denotado como $\boldsymbol{Y}_{known1}$, que son dominadas por alguna solución

del Frente de Pareto del Algoritmo 2 ($\boldsymbol{Y}_{known2}$). Lógicamente, al comparar dos algoritmos, el que tenga un menor valor de *C*, será superior.

7 Resultados Experimentales

El MOACS-VRPTW ha sido implementado en una computadora personal, corriendo un sistema operativo UNIX y utilizando el lenguaje C. El PAES-DRI ha sido implementado en MATLAB sobre un sistema operativo Windows 98. Los resultados presentados a continuación pertenecen al conjunto de datos (problema tipo), publicado en [2] con la denominación de "*R101*" . Los parámetros utilizados en la implementación del PAES-DRI son $\alpha_1 \_ 0.7, \alpha_2 \_ 0.3, \alpha_3 \_ 0.5, \hat{n} \_ 30, \hat{d} \_ 20$, escogidos en forma experimental sin hacer un ajuste fino de sus valores óptimos para este problema.

Se han realizado cinco corridas de cada algoritmo, formando los Frentes Pareto $\boldsymbol{Y}_{MOACS}$ y $\boldsymbol{Y}_{PAES\text{-}DRI}$. Como no se conoce la solución teórica óptima del problema, se aproximó $\boldsymbol{Y}_{true}$, mediante la unión de todas las soluciones obtenidas por los investigadores, utilizando los 2 algoritmos que se están comparando, y eliminando de este conjunto las soluciones dominadas.

Cabe destacar que el tiempo medio de ejecución del PAES-DRI fue de 4 horas 20 minutos, considerablemente mayor al tiempo de 1 hora 5 minutos, utilizado para la ejecución del MOACS-VRPTW, lo que en parte puede justificar los excelentes resultados obtenidos con el PAES-DRI, propuesto en este trabajo.

En la Tabla 1 se pueden observar los resultados experimentales que demuestran experimentalmente que el PAES-DRI obtiene en general mejores soluciones (ver métricas OTVGN y C) pero sin llegar a encontrar la misma variedad de soluciones que el MOACS-VRPTW, que resulta superior si se considera la métrica ONVG, y sobre todo, el tiempo requerido para encontrar buenas soluciones Pareto.

Tabla 1. Métricas de desempeño para los dos algoritmos comparados en el presente trabajo.

Algoritmo	*ONVG*	*OTVNG*	*C*
MOACS-VRPTW	12	0	12
PAES-DRI	5	5	0

En efecto, la Tabla 1 muestra que el MOACS-VRPTW obtiene un mayor número de soluciones no dominadas, pero todas estas soluciones terminan siendo subóptimas, pues a su vez son dominadas por las soluciones obtenidas con el PAES-DRI, que en todos los casos pertenecen al frente de Pareto óptimo. Es decir, todas las soluciones del MOACS-VRPTW son dominadas por el PAES-DRI.

7 Conclusión

El problema del ruteo de vehículos con ventanas de tiempo va ganando importancia en la medida que el mundo globalizado fuerza a las empresas distribuidoras modernas a ser cada vez más eficientes e incrementar su región de trabajo, para mantenerse competitivas. Como natural consecuencia, crece el interés en el área y se van proponiendo nuevos abordajes, entre los que se destacan los algoritmos evolutivos, por su flexibilidad para analizar circunstancias cambiantes y objetivos contradictorios. A raíz de esto, resulta interesante comparar las nuevas alternativas que se van proponiendo para lograr aplicar el abordaje más adecuado a cada realidad.

Es en este contexto que el presente trabajo propone una variante de algoritmo evolutivo multiobjetivo y realiza por primera vez una comparación entre un sistema de colonias de hormigas y una estrategia evolutiva para el problema del ruteo multiobjetivo de vehículos con ventanas de tiempo.

Los experimentos demuestran que aunque el MOACS-VRPTW obtenga mayor cantidad de soluciones no dominadas, todas éstas son a su vez dominadas por las soluciones obtenidas por el PAES-DRI, pero esta superioridad se ve opacada por el hecho de requerir un mayor tiempo de ejecución (en el orden de 4 veces, en las experiencias reportadas). En consecuencia, a pesar de que el MOACS-VRPTW no encuentra soluciones de la misma calidad que el PAES-DRI, todavía es razonable su utilización cuando es suficiente obtener buenas soluciones sub-óptimas en un tiempo más limitado.

Como los resultados aquí reportados no son totalmente conclusivos, los autores se encuentran trabajando en pruebas más exhaustivas, con problemas más grandes y con diversas variantes de algoritmos evolutivos multiobjetivos, con el objeto de reconocer claramente las circunstancias en que cada abordaje presenta sus mejores propiedades, dado que no es posible encontrar un único abordaje que presente soluciones igualmente favorables en todas las instancias del problema.

Referencias

1. Bodin, L. Golden, B., Assad, A. y Ball, M. Routing and Scheduling of Vehicles and Crews (1983). *The State of the Art*. Comput. Opns. Res. 10, 62-212.
2. Solomon, M.M. Algorithms for Vehicle Routing and Scheduling Problems with time window constraints. Northeastern University, Boston, Massachusetts, December 1985.
3. Lau, K.K., Kumar, M.J. y Achuthan, N.R. Parallel implementation of branch and bound algorithm for solving vehicle routing problem on NOWs. *3rd Internationl Symposium on Parallel Architectures, Algorithms, and Networks*, 1997, 247-253.
4. Pedroso, J.P. Niche search: An application in vehicle routing. *IEEE World Congress on Computational Intelligence*, 1998, 177–182.
5. Yoshiike, N.y Takefuji, Y. Vehicle routing problem using clustering algorithm by maximum neural networks. *2nd International Conf. on Intelligent Processing and Manufacturing of Materials*, 2, 1999, 1109–1113.
6. Gomes,L. y von Zuben, F.J. A neuro-fuzzy approach to the capacitated vehicle routing problem. *Int. Joint Conference on Neural Networks*, 2, 2002, 1930-1935.
7. Murao, H., Tohmata, K., Konishi, M. y Kitamura, S. Pheromone based transportation scheduling system for the multi-vehicle routing problem *IEEE Int. Conference on Systems, Man, and Cybernetics*, 4, 1999, 430–434.
8. Takeno, T., Tsujimura, Y. y Yamazaki, G. A single-phase method based on evolution calculation for vehicle routing problem. *4th Int. Conf. on Computational Intelligence and Multimedia Applications*, 2001, 103–07.
9. Gupta, A. y Krishnamurti, R. Parallel algorithms for vehicle routing problems. *Fourth International Conference on High-Performance Computing*, 1997, 144-151.
10. Louis, S.J., Xiangying, Y. y Zhen, Y.Y. Multiple vehicle routing with time windows using genetic algorithms. *Congress on Evolutionary Computation*, 1808(3), 1999.
11. Maeda, O., Nakamura, M., Ombuki, B.M. y Onaga, K. A genetic algorithm approach to vehicle routing problem with time deadlines in geographical information systems. *International Conference on Systems, Man, and Cybernetics*, 4, 1999, 595-600.
12. Chin, A., Kit, H. y Lim, A. A new GA approach for the vehicle routing problem. *11th IEEE International Conference on Tools with Artificial Intelligence*, 1999, 307-310.
13. Tan, K.C., Lee, T.H., Ou, K. y Lee, L.H. A messy genetic algorithm for the vehicle routing problem with time window constraints. *Congress on Evolutionary Computation*, 1, 2001, 679-686.
14. Arbelaitz, O., Rodriguez, C. y Zamkola, I. Low cost parallel solutions for the VRPTW optimization problem. *Fourth International Conference on Parallel Processing Workshops*, 2001, 176-181.
15. Czech, Z.J. y Czarnas, P. Parallel simulated annealing for the vehicle routing problem with time windows. *10th Euromicro Workshop on Parallel, Distributed and Network-based Proc.*, 2002, 376-383.
16. Gambardella, L., Taillard, E. y Agazzi, G. *News ideas in optimization*. Mac Graw-Hill, London 1999, 73–76.
17. Bräysy, O. y Gendreau, M. Route construction and local search algorithms for the vehicle routing problem with time windows, Report STF42 A01024, App. Mathematics, Dep. of Optimization, Norway. 2001.

18. Bräysy, O. y Gendreau, M. Metaheuristics for the vehicle routing problem with time windows, Internal Report STF42 A01025, SINTEF Applied Mathematics, Department of Optimization, Norway. 2001.
19. Barán, B. y Schaerer, M. A multiobjective Ant Colony System for Vehicle Routing Problem with Time Windows. Proceedings of the 21st IASTED International Conference APPLIED INFORMATICS. Innsbruck, Austria. 2003.
20. Mester, D. The Parallel algorithm for Vehicle Routing Problem with Time Windows restrictions. Scientific Report, Minerva Optimization Center, Technion, Israel. 1999.
21. Mester, D. An evolutionary strategies algorithm for large scale vehicle routing problem with capacitate and time windows restrictions, Institute of Evolution, Mathematical and Population Genetics, Univ. of Haifa, Israel. 2002.
22. Van Veldhuisen, D. A. Multiobjective Evolutionary Algorithms: Classifications, Analyses and New Innovations. PhD thesis, Department of Electrical and Computer Engineering. Graduate School of Engineering. Air Force Institute of Technology. Ohio, USA. May, 1999.
23. Knowles, J. and Corne, D. The Pareto Archived Evolution Strategy: A New Baseline Algorithm for Pareto Multiobjective Optimisation. Dept. of Computer Science. University of Reading. UK. 1999.
24. Bräysy, O. A reactive variable neighborhood for the Vehicle Routing Problem with Time Windows. 2001. To appear in Informs Journal on Computing.
25. Or, I. Traveling Salesman-Type Combinatorial Problems and their Relation to the Logistics of Regional Blood Banking, Ph.D. thesis, Northwestern University, Evanston, Illinois. 1976.
26. Osman, I.H. Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problems, *Annals of Operations Research* 41, 421–452. 1993.