Optimización basada en Colonias de Hormigas para el Problema del Vendedor Viajante con muchos objetivos contradictorios

Francisco Riveros[1], Néstor Benítez[1], Julio Paciello[1] y Benjamín Barán[1,2]

[1] Universidad Nacional de Asunción
{friverosn,nestorbdietrich,juliopaciello}@gmail.com,
bbaran@pol.una.py

[2] Universidad Nacional del Este

Abstract. Considerando el conocido problema de disminución de rendimiento que presentan los algoritmos evolutivos cuando resuelven problemas denominados *many-objective*, este trabajo propone la utilización de una variante de la Optimización basada en Colonias de Hormigas que denominamos λ base-p. La nueva propuesta fue sometida a distintas pruebas experimentales sobre instancias del Problema del Vendedor Viajante con muchos objetivos utilizando la métrica del Hipervolumen. La Asignación de λ base-p fue comparada con algoritmos MOACO del estado del arte y con el algoritmo evolutivo NSGA2, demostrando que logra calcular un mejor Hipervolumen cuando se resuelven problemas de muchos objetivos contradictorios (*many-objective*).

Keywords: MOACO, TSP, *many-objective*, Hipervolumen, NSGA2

1 Introducción

Los algoritmos *Multi-Objective Ant Colony Optimization* (MOACO) fueron tradicionalmente utilizados para resolver problemas con múltiples objetivos contradictorios, denominados *Multi-Objective Optimization Problems* (MOP), considerados como uno de los métodos con mejor desempeño para resolver el Problema del Vendedor Viajante (TSP-*Travelling Salesman Problems*) [1] . Trabajos anteriores como [2] , [3] y [4] utilizaron los algoritmos MOACO para resolver los denominados MOP del mundo real, y lo hicieron de manera efectiva para 2 o 3 objetivos. Sin embargo, esta efectividad puede verse afectada cuando el número de objetivos aumenta, como pasa con la mayoría de los algoritmos evolutivos [7]. La problemática tratada en este trabajo es el desempeño de los algoritmos MOACO cuando son utilizados para resolver problemas *many-objective* (problemas con 4 o más objetivos contradictorios). En este contexto, se implementó una nueva estrategia de MOACO para resolver problemas *many-objective* ya que las implementaciones actuales no contemplan resolver estos problemas con muchos objetivos. En las siguientes secciones se describe el TSP *many-objective*,

los algoritmos MOACO considerados en el presente trabajo, la estrategia propuesta por este trabajo para considerar problemas *many-objective* y finalmente, se presentan los resultados experimentales, las conclusiones y trabajos futuros.

2 TSP *many-objective*

El TSP puede ser representado por un grafo ponderado completamente conexo $G = (N, A)$, donde N representa un conjunto de $c = |N|$ nodos y A un conjunto de aristas que conectan los nodos de N. En un caso mono-objetivo, cada arista tiene asociado una función de costo d_{ij}, que representa la distancia entre los nodos i y j. El TSP consiste en encontrar el ciclo Hamiltoniano que minimice la distancia recorrida desde un nodo inicial, pasando por cada nodo exactamente una vez y volviendo finalmente al nodo inicial [1]. Para formular el TSP se puede considerar una variable dicotómica $x_{i,j}$ para todo $(i, j) \in A$, de forma que tome el valor 1 si el arco forma parte del ciclo Hamiltoniano y 0 en otro caso, entonces, el TSP consiste en:

$$minimizar \sum_{(i,j)\in A} d_{i,j}x_{i,j} \tag{1}$$

Para el caso del TSP *many-objective*, son consideradas k funciones de costos, teniendo cada arista (i, j) asociado un conjunto de k distancias $d^1_{ij}, d^2_{ij}..., d^k_{ij}$. El problema consiste en minimizar las k funciones de costos al mismo tiempo [8], es decir:

$$minimizar \left[\sum_{(i,j)\in A} d^1_{i,j}x_{i,j}; \sum_{(i,j)\in A} d^2_{i,j}x_{i,j}; ...; \sum_{(i,j)\in A} d^k_{i,j}x_{i,j} \right] \tag{2}$$

3 Algoritmos MOACO

Los algoritmos *Ant Colony Optimization* (ACO) [12] se inspiran en el comportamiento de colonias de hormigas reales para resolver problemas de optimización combinatoria. Se basan en una colonia de hormigas artificiales, es decir, colonia de agentes computacionales simples que trabajan de manera cooperativa y se comunican mediante rastros de feromonas artificiales. Para aplicar un ACO al problema aquí tratado, se utiliza un conjunto de agentes llamados hormigas. Las hormigas construyen las soluciones del problema recorriendo el grafo $G = (N, A)$, partiendo desde un nodo inicial. Para cada cambio de nodo, una hormiga considera los parámetros de visibilidad $(\eta_{i,j})$ y feromonas $(\tau_{i,j})$ de cada arco utilizando una política probabilística que está dada por la siguiente fórmula:

$$p_{i,j} = \begin{cases} \frac{\tau^{\alpha}_{i,j}\eta^{\beta}_{i,j}}{\sum_{x\in J_i}\tau^{\alpha}_{i,x}\eta^{\beta}_{i,x}} & \text{si } j \in J_i \\ 0 & \text{en otro caso} \end{cases} \tag{3}$$

donde J_i representa a todos los posibles nodos que pueden ser visitados desde el nodo i (esto es, los nodos que todavía no han sido visitado por una hormiga);

mientras que α y β son parámetros definidos a priori que reflejan la importancia relativa de las feromonas y la visibilidad respectivamente. La actualización de las feromonas está dado por:

$$\tau_{i,j} = (1-\rho)\tau_{i,j} + \rho\Delta\tau \tag{4}$$

donde ρ es el coeficiente que se utiliza para representar la evaporación y $\Delta\tau$ es la cantidad de feromonas que se va sumando a los arcos que forman parte de una solución escogida.
Un MOACO es una extensión de la metaheurística ACO para resolver problemas multi-objetivo. En general, las ecuaciones se modifican, conforme se muestra a continuación [4]:

$$p_{i,j} = \begin{cases} \frac{\tau_{i,j}^{\alpha}(\eta_{i,j}^{1}\eta_{i,j}^{2}\ldots\eta_{i,j}^{k})^{\beta}}{\sum_{x\in J_i}\tau_{i,x}^{\alpha}(\eta_{i,x}^{1}\eta_{i,x}^{2}\ldots\eta_{i,x}^{k})^{\beta}} & \text{si } j \in J_i \\ 0 & \text{en otro caso} \end{cases} \tag{5}$$

$$\Delta\tau = \frac{1}{\sum_{l=1}^{k}\sum_{(i,j)\in A} d_{i,j}^{l} x_{i,j}} \tag{6}$$

donde k es la cantidad de objetivos. En el Algoritmo 1 se puede observar el pseudo-código de un MOACO genérico:

Algoritmo 1. Pseudo-código de un MOACO genérico .

```
procedure MOACO()
   inicializarParametros()
   while not condicionParada()
       generacion = generacion + 1
       for ant = 1 to m //m es la cantidad de hormigas
             construirSolucion()
             evaluarSolucion()
             actualizarCPFeromonas() // definido en (4)
       end for
    end while
return ConjuntoPareto
end procedure

procedure construirSolucion()
   sol = {}
   while existenEstadosNoVisitados()
        siguiente = seleccionarSiguienteEstado() // definido en (5)
        sol = sol U siguiente
        marcarVisitado(siguiente)
        if (actualizacionPasoAPaso)
           actualizarFeromonasPasoAPaso() // por ejemplo, usando (4)
        end if
   end while
end procedure
```

Actualmente existen diversas implementaciones de MOACOs que son objeto de constantes investigaciones [5], [6]. Cada implementación incluye diferentes enfoques de actualización de feromonas, cantidad de tablas de feromonas, cantidad de hormigas, etc. Para este trabajo se seleccionaron tres algoritmos MOACO del estado del arte: el *Multi-objective Ant System* (MAS) , el *Multi-objective Ant Colony System* (MOACS) y el *Multi-objective Max-Min Ant System* (M3AS), en base a su buen comportamiento resolviendo el problema TSP experimentalmente demostrado en [4]. Cabe recordar que en un MOACO, cada hormiga construye una solución mediante la transición de estados dado por ejemplo en (5). En el caso específico del MOACS, se utiliza la selección pseudo-aleatoria dada por (7) que considera el uso de la probabilidad de transición ilustrada en la ecuación (8), que también es utilizada como regla de transición por las variantes M3AS y MAS.

$$j = \begin{cases} max_{j \in J_i}\{\tau_{i,j}^{\alpha}(\eta_{i,j}^{1\lambda_1}\eta_{i,j}^{2\lambda_2}...\eta_{i,j}^{k\lambda_k})^{\beta}\} & \text{si } q < q_0 \\ \hat{\text{p}} & \text{en otro caso} \end{cases} \tag{7}$$

donde la variable $\hat{\text{p}}$ se calcula mediante la probabilidad $p_{i,j}$ que se define como:

$$p_{i,j} = \begin{cases} \frac{\tau_{i,j}^{\alpha}(\eta_{i,j}^{1\lambda_1}\eta_{i,j}^{2\lambda_2}...\eta_{i,j}^{k\lambda_k})^{\beta}}{\sum_{x \in J_i}\tau_{i,x}^{\alpha}(\eta_{i,x}^{1\lambda_1}\eta_{i,x}^{2\lambda_2}...\eta_{i,x}^{k\lambda_k})^{\beta}} & \text{si } j \in J_i \\ 0 & \text{en otro caso} \end{cases} \tag{8}$$

y las variables λ_1 a λ_k son parámetros utilizados para ponderar la importancia relativa de cada objetivo. Típicamente, cada hormiga puede utilizar un conjunto diferente de parámetros λ_1 a λ_k de forma a guiar la búsqueda de cada hormiga a regiones diferentes del espacio objetivo, lo que explica el reconocido éxito del algoritmo MOACS [4]. Todos los MOACOs implementados fueron objetos de varias pruebas experimentales utilizando las instancias del TSP *many-objective* generadas considerando 2, 4, 8 y 10 objetivos. Las pruebas experimentales fueron ejecutadas en un contexto de minimización simultánea de todas las funciones objetivos (costos). Luego de un análisis previo, realizado a las implementaciones de MOACOs aplicados al TSP *many-objective*, se pudo observar: (1) un deterioro de sus rendimientos al aumentar el número de funciones objetivos; y (2) el MOACS resultaba levemente superior a las demás alternativas. En consecuencia, se decidió modificar los parámetros de ejecución del MOACS de forma a hacerlo aún más competitivo al resolver problemas *many-objective*.

4 Métrica de rendimiento

Para la comparación de los resultados experimentales, fue utilizada la métrica del Hipervolumen [9], reconociendo su amplia adopción entre los investigadores del área de optimización multi-objetivo [7]. El Hipervolumen considera el tamaño de la región de dominancia en el espacio objetivo combinando las métricas de distancia, distribución y extensión en un solo valor [9]. Dado un conjunto de soluciones $Z = \{z_1, z_2, ..., z_R\}$ con R soluciones, y un punto de referencia y_{ref},

el Hipervolumen se define como [9]:

$$H(Z) = H(Z, y_{ref}) = \Lambda(\bigcup_{i=1}^{R} H(z_i)) \tag{9}$$

El Hipervolumen de un Frente Pareto X se define como la unión de las porciones que están limitadas por el punto de referencia y_{ref} denominado punto anti-óptimo o peor posible y los puntos del Frente Pareto evaluado. Considerando un problema de dos funciones objetivos con dos Frentes Pareto conocidos $X = \{a, b, c, d, e\}$, $X' = \{a', b', c', d', e'\}$ y el punto de referencia y_{ref}, si el Hipervolumen de X es mayor al de X', entonces, el Frente Pareto X no es peor que el Frente Pareto X' [10].

5 Implementación propuesta

En las reglas de transición de las fórmulas (7) y (8) podemos notar un parámetro de gran importancia que se utiliza para asignar una cierta prioridad a cada uno de los objetivos tenidos en cuenta, este parámetro es denominado λ. Típicamente, el parámetro λ es un valor complementario; de modo que si se aumenta para un objetivo, se debe disminuir para del otro y viceversa. Con esta asignación de λ se busca ponderar más la información de un objetivo que la de otro para así especializarse en una zona del espacio de búsqueda. Por ejemplo, para problemas bi-objetivos [2], se puede escoger los valores de λ_1 y λ_2 utilizando la siguiente relación:

$$\lambda_2 = m - \lambda_1 + 1 \tag{10}$$

donde m representa el número de hormigas y λ_1 toma valores enteros entre 1 y m. Cabe mencionar que para un problema de k objetivos $f_1, f_2, ..., f_k$, se debe asignar valores a $\lambda_1, \lambda_2, ..., \lambda_k$ por cada hormiga, donde λ_1 pondera a f_1, λ_2 pondera a f_2 y así sucesivamente. En general, cada λ toma uno de m valores posibles por lo que se necesitaran k^m hormigas para tener todas las posibles permutaciones, lo cual resulta inviable al crecer el número de objetivos k. Para abordar esta problemática, este trabajo propone una nueva estrategia denominada Asignación de λ base-p.

5.1 Asignación de λ base-p

Esta nueva estrategia aplica una restricción a los valores posibles que puede tomar cada λ que queda así restringido por un parámetro $p \in \mathbb{N}$. Dados los objetivos $f_1, f_2, ..., f_k$, donde k es la cantidad de objetivos, cada hormiga debe escoger aleatoriamente sus valores $\lambda_1, \lambda_2, ..., \lambda_k$. Con este propósito, cada hormiga selecciona en forma aleatoria un número entero $\tilde{n}$ en base decimal, tal que, $\tilde{n} \in [0, (p^k - 1)]$, donde p es el parámetro de esta estrategia que representa la cantidad de valores posibles que puede tomar cada λ. El número entero $\tilde{n}$ es convertido a su representación en base p, de esta manera se obtiene un número con hasta k dígitos, donde cada dígito corresponde a un valor del parámetro λ.

En este trabajo se utilizó $p = 3$, por lo que cada λ_i puede tomar los valores: 0 (visibilidad no considerada), 1 (peso medio) y 2 (peso alto). Como ejemplo, consideremos un problema con $k = 8$ objetivos y $p = 3$. Una hormiga seleccionaría en forma aleatoria un número entero $\tilde{n}$ entre 0 y $(3^8 - 1)$. Suponiendo que escoja $\tilde{n} = 4589$, se realiza la conversión de $\tilde{n}$ a su representación en base 3 dando por resultado el número 20021222; entonces, cada dígito de éste número es asignado a cada valor de λ, es decir, $\lambda_1 = 2, \lambda_2 = 0, \lambda_3 = 0, \lambda_4 = 2, \lambda_5 = 1, \lambda_6 = 2, \lambda_7 = 2$ *y* $\lambda_8 = 2$. Como se puede notar, la estrategia de Asignación de λ base-p mantiene la escalabilidad con un mayor número de objetivos, aparte de ser una estrategia simple de implementar y comprender.

6 Resultados experimentales

En esta sección se presentan los resultados experimentales obtenidos al comparar el MOACS base-p propuesto en este trabajo con los algoritmos del estado del arte antes citados (MOACS, M3AS y MAS) utilizando la recogida métrica del Hipervolumen presentada en la sección 4. Todas las pruebas fueron ejecutadas en un ordenador Genuine Intel 2.3 GHz con una arquitectura de 64 bits y 15 GB RAM, bajo el sistema operativo Ubuntu 12.04.2 LTS.

A falta de instancias conocidas en la literatura para el TSP *many-objective*, se generaron en forma aleatoria problemas para 2, 4, 8 y 10 objetivos cuidando que la correlación entre dos matrices de costo de un mismo problema no supere el valor de 0.1. Con esto, los objetivos minimizados simultáneamente resultaron contradictorios o al menos poco correlacionados.

Para la ejecución de las pruebas se modificó el algoritmo original MOACS, implementando la Asignación de λ base-p, creando así, una nueva variante de MOACO denominada en adelante MOACS base-p, principal aporte de este trabajo. Dicha variante fue comparada con los mejores algoritmos MOACO del estado del arte según [4], MOACS, M3AS y MAS, además del NSGA2 (*Non Sorting Genetic Algorithm version 2*) considerado el principal algoritmo de referencia del estado del arte en problemas multi-objetivo [7], [11].

Para realizar un análisis de la evolución de los resultados, se definió como criterio de parada, 10.000 generaciones y se obtuvieron los resultados para 2.500, 5.000, 7.500 y 10.000 generaciones transcurridas.

Considerando la parametrización reportada en [4], para los MOACOs se utilizaron los siguientes parámetros: $m = 10$ hormigas, $\alpha = 1$, $\beta = 2$, $\rho = 0.3$, $\tau_0 = 0.1$, $\tau_{max} = 0.9$, $\tau_{min} = 0.1$ y $q_0 = 0.5$. Para el NSGA2, se utilizaron los siguientes parámetros: tamaño de población igual a 10, Probabilidad de Mutación de 0.8 y Probabilidad de cruzamiento de 0.98. Estos valores fueron seleccionados conforme [11].

Los resultados presentados más abajo corresponden al promedio de 4 ejecuciones de cada uno de los algoritmos mencionados. Se utilizaron instancias del TSP con 2, 4, 8 y 10 objetivos y cada instancia fue configurada de manera a considerar tres tamaños de problemas: 50, 75 y 100 ciudades.

En la Figura 1 se puede observar la evolución del Hipervolumen de los Frentes Pareto calculados por cada uno de los algoritmos comparados, cuando resuelven una instancia del TSP con 75 ciudades, considerando solamente 2 objetivos (este problema no puede todavía ser considerado *many-objective*). Se puede notar que el MOACS base-p es levemente superior a los demás algoritmos sin importar el número de generaciones transcurridas, aunque el MAS tiene un desempeño muy próximo.

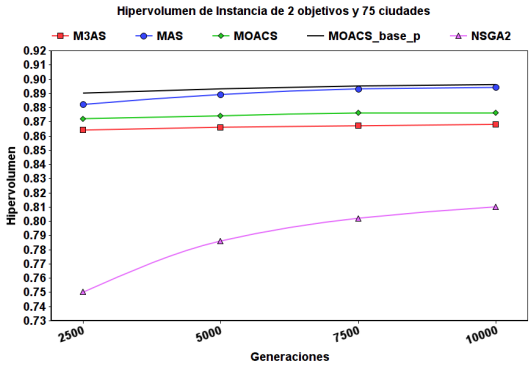

Fig. 1: Hipervolumen obtenido al resolver la Instancia con 2 objetivos y 75 ciudades

La Figura 2 muestra la evolución del Hipervolumen para cada uno de los algoritmos comparados, cuando resuelven una instancia del TSP con 75 ciudades, considerando 4 objetivos simultáneos (ya puede ser considerado un problema *many-objective*). Se puede notar que la variante propuesta es claramente superior a los demás algoritmos sin importar el número de generaciones transcurridas, verificandose la ventaja de utilizar la Asignación de λ base-p propuesta en este trabajo.

Las Figuras 3 y 4 muestran la evolución del Hipervolumen para cada uno de los algoritmos comparados, cuando resuelven una instancia del TSP con 75 ciudades, considerando 8 y 10 objetivos respectivamente, pudiéndose notar una clara superioridad del algoritmo propuesto en este trabajo que mejora su desempeño notoriamente con respecto a los demás algoritmos del estado del arte al aumentar el número de objetivos, confirmando la ventaja que significa utilizar el algoritmo propuesto.

Este mismo comportamiento fue observado al resolver instancias con 50 y 100 ciudades. Para instancias de más de 2 objetivos el MOACS base-p presenta de manera significativa el mejor Hipervolumen de entre todos los algoritmos estudiados en cualquier cantidad de generaciones. Por falta de espacio solo se presentará a continuación los resultados obtenidos utilizando instancias de 50 ciudades, resumidos en las Tablas 1 y 2.

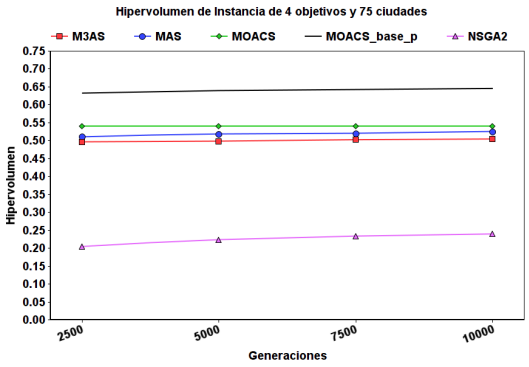

Fig. 2: Hipervolumen obtenido al resolver la Instancia con 4 objetivos y 75 ciudades

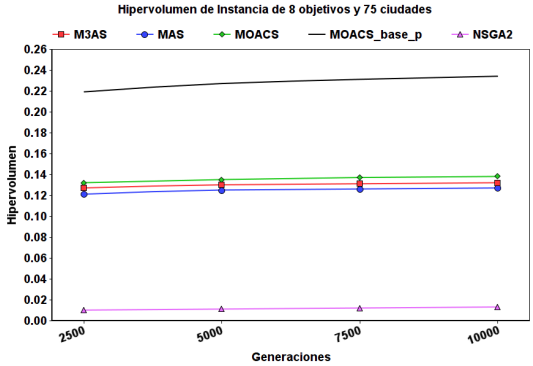

Fig. 3: Hipervolumen obtenido al resolver la Instancia con 8 objetivos y 75 ciudades

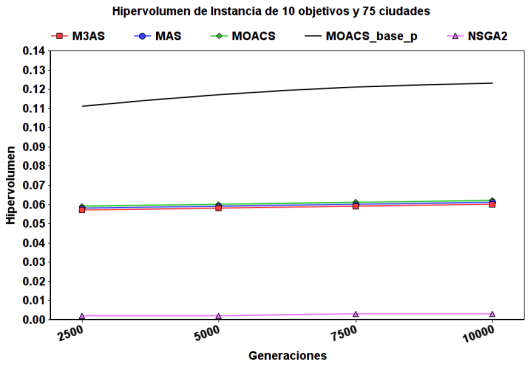

Fig. 4: Hipervolumen obtenido al resolver la Instancia con 10 objetivos y 75 ciudades

Objetivos	2				4			
Generaciones	2.500	5.000	7.500	10.000	2.500	5.000	7.500	10.000
M3AS	0,893	0,895	0,896	0,897	0,643	0,648	0,65	0,651
MAS	0,903	0,911	0,912	0,912	0,633	0,638	0,645	0,656
MOACS	0,898	0,899	0,9	0,901	0,653	0,656	0,658	0,66
MOACS_base_p	0,915	0,916	0,917	0,917	0,743	0,748	0,751	0,753
NSGA2	0,743	0,78	0,796	0,806	0,201	0,215	0,225	0,233

Tabla 1: Hipervolumen obtenido al resolver la Instancias con 50 ciudades para 2 y 4 objetivos

Objetivos	8				10			
Generaciones	2.500	5.000	7.500	10.000	2.500	5.000	7.500	10.000
M3AS	0,289	0,293	0,295	0,296	0,178	0,181	0,182	0,183
MAS	0,284	0,288	0,292	0,294	0,181	0,186	0,188	0,19
MOACS	0,292	0,295	0,297	0,298	0,178	0,182	0,184	0,186
MOACS_base_p	0,411	0,421	0,426	0,431	0,279	0,289	0,294	0,298
NSGA2	0,009	0,011	0,012	0,013	0,002	0,002	0,002	0,003

Tabla 2: Hipervolumen obtenido al resolver la Instancias con 50 ciudades para 8 y 10 objetivos

7 Conclusiones y trabajos futuros

Este trabajo presentó por primera vez en la literatura un estudio del comportamiento de los algoritmos MOACO aplicados a problemas *many-objective*. Se introdujo una nueva y simple variante del MOACS denominada MOACS base-p, que fue empíricamente comparada contra los algoritmos MOACO de mejor rendimiento al resolver el TSP (MOACS, M3AS, MAS) [4] y el algoritmo evolutivo de referencia para resolver problemas multi-objetivo, el NSGA2 [7]. Todos estos algoritmos fueron utilizados para resolver el TSP con 2, 4, 8 y 10 objetivos; almacenando los Frentes Pareto para diferentes cantidades de generaciones, y promediando los resultados de varias ejecuciones. Estos resultados fueron comparados utilizando la reconocida métrica de Hipervolumen, ampliamente utilizada [7].

Descrita la problemática existente en la asignación del parámetro λ en los algoritmos MOACO cuando resuelven problemas *many-objective*, se propuso como solución, una nueva estrategia de asignación de λ denominada Asignación de λ base-p, en la cual se basa el MOACS base-p.

Resultados experimentales demostraron que el MOACS base-p es superior a los mejores MOACO del estado del arte así como al algoritmo de referencia NSGA2, incluso con solo 2 objetivos aunque claramente su ventaja aumenta en la medida que crece el número de objetivos, conforme se verificó experimentalmente hasta 10 objetivos, quedando como trabajo futuro considerar un mayor número de objetivos para lo cual se deberá reconsiderar el uso de la métrica de rendimiento dada la complejidad del cálculo del Hipevolumen que crece ex-

ponencialmente con el número de objetivos. Otras extensiones del presente trabajo serían: implementar el esquema de Asignación de λ base-p en otras implementaciones de MOACO y compararlas con otra metaheurística como el *Particle Swarm Optimization* (PSO), extendiendo el análisis comparativo a otros casos de estudios como son el *Quadratic Assignment Problem* (QAP)[5], el *Vehicle Routing Problem with Time Windows* (VRPTW)[2]; u otros problemas multiobjetivo que puedan ser extendidos para el caso *many-objective*; por último, se propone analizar otros parámetros de los algoritmos MOACO, como por ejemplo la utilización de grupos de tablas de feromonas donde cada tabla de feromonas corresponderá a un grupo de objetivos.

Referencias

1. Lawler, E.L., Lenstra, J.K., Rinnooy Kan, A.H.G., Shmoys, D.B.: The travelling salesman problem. Wiley, New York (1985)
2. Barán, B., Schaerer, M.: A multiobjective Ant Colony System for Vehicle Routing Problems with Time Windows. Proc. Twenty first IASTED International Conference on Applied Informatics, pp. 97–102. Insbruck, Austria (2003)
3. Pinto, D., Barán, B.: Solving Multiobjective Multicast Routing Problem with a new Ant Colony Optimization approach. In Proceedings of the 3rd international IFIP/ACM Latin American conference on Networking, pp. 11-19, Cali, Colombia (2005)
4. Paciello, J., Martínez, H., Lezcano, C., Barán, B.: Algoritmos de Optimizacin multiobjetivos basados en colonias de hormigas. XXXII Conferencia Latinoamericana de Informática - CLEI'2006. Santiago de Chile (2006)
5. Martínez, H., Paciello, J., Barán, B.: Equipo distribuido de algoritmos ACO multiobjetivos. VIII Argentine Symposium on Artificial Intelligence - ASAI, en el marco de las 35 Jornadas Argentinas de Informática e Investigación Operativa - 35 JAIIO. Mendoza, Argentina (2006)
6. Paciello, J., Martínez, H., Barán, B.: Aplicación de un equipo de algoritmos ACO al VRPTW bi-objetivo. XIII Conferencia Latino-IberoAmericana de Investigación Operativa - CLAIO'2006. Montevideo, Uruguay (2006)
7. von Lücken, C., Barn, B., Brizuela, C.: A survey on multy-objective evolucionay algorithms for many-objective problems. Computational Optimization and Applications 58, no. 3, pp. 707–756, (2014)
8. Ishibuchi, H., Tsukamoto, N., Nojima, Y.: Evolutionary many-objective optimization: A short review. In IEEE Congress on Evolutionary Computation CEC'2008, pp. 2419-2426 (2008)
9. Zitzler, E.: Evolutionary algorithms for multiobjective optimization: Methods and applications. Vol. 63. Ithaca: Shaker, (1999)
10. While, L., Hingston, P., Barone, L., Huband, S.: A faster algorithm for calculating hypervolume. Evolutionary Computation, IEEE Transactions on, 10 no. 1, pp. 29–38 (2006)
11. Optymalizacji, H. M., Kot, K., Pietraszko, A.: Heurystyczne metody wielokryterialnej optymalizacji kombinatorycznej. Akademia Górniczo-Hutnicza Thesis (2009)
12. Barán, B., Gómez, O.: Reasons of ACO's Success in TSP, ANTS'2004 - Fourth International Workshop on Colony Optimization and Swarm Intelligence. Bruselas, Bélgica. (2004)