Algoritmo Multiobjetivo de Colonia de Abejas Artificiales aplicado al Problema de Orientación

Rodrigo Martín-Moreno
Depto. Tecnología de Computadores y Comunicaciones
Universidad de Extremadura
Cáceres, España
rmartinky@alumnos.unex.es

Miguel A. Vega-Rodríguez
Depto. Tecnología de Computadores y Comunicaciones
Universidad de Extremadura
Cáceres, España
mavega@unex.es

***Resumen*—Hemos desarrollado un nuevo algoritmo para la resolución de problemas de orientación multiobjetivo, que tienen aplicación directa en el cálculo de rutas óptimas, logística, o bien el diseño de rutas turísticas, nuestra motivación principal. Los turistas priorizan según sus gustos el visitar puntos de interés, los cuales se pueden categorizar (p. ej. culturales u ocio), por lo que es necesaria la utilización de sistemas expertos multiobjetivo. Para conseguir las mejores soluciones de Pareto, hemos adaptado el algoritmo de colonia de abejas artificiales al contexto multiobjetivo. El rendimiento del algoritmo se ha comparado con dos algoritmos usados previamente en la literatura de los problemas de orientación multiobjetivo (P-ACO y P-VNS), y los resultados obtenidos indican que este nuevo algoritmo es acertado para resolver problemas de orientación multiobjetivo.**

***Palabras clave*—Algoritmo de colonia de abejas artificiales, Optimización multiobjetivo, Problema de orientación, Inteligencia de enjambre, Computación evolutiva.**

I. Introducción

Existe una gran variedad de sistemas de apoyo a la toma de decisiones, usados para diferentes propósitos como marketing, finanzas, logística o recursos humanos, por ejemplo. Sin embargo, existen pocos sistemas para ayudar a la gente a diseñar rutas turísticas que encajen en sus preferencias, más bien se les ofrece un conjunto de rutas prefijadas poco personalizables. Cuando los visitantes planean una visita a un destino, priorizan qué lugares o puntos de interés (POIs) merecen ser visitados, teniendo en cuenta su presupuesto, tiempo e interés y decidiendo su orden en la ruta. Por ello, un sistema de apoyo a la toma de decisiones, que ayude a los visitantes en este proceso, puede ser muy interesante. En todo caso, la decisión final recaerá siempre sobre el visitante, ya que el sistema solo ofrecerá las mejores soluciones para sus requerimientos, pero no contemplará la parte emocional de planear una visita a un destino.

El hecho de crear rutas que conecten POIs se define como un Problema de Orientación (OP) [1]. En este caso, nos centramos en el Problema de Orientación MultiObjetivo (MOOP), donde existen varias categorías para cada POI (p. ej.: cultural u ocio) y cada uno de ellos tiene diferentes beneficios para cada categoría. Hemos desarrollado un algoritmo MultiObjetivo de Colonia de Abejas Artificiales (MOABC), basándonos en el algoritmo mono-objetivo ABC propuesto por [2], para resolver el MOOP de una manera competitiva. Es la primera vez que el algoritmo MOABC se aplica en la resolución de problemas de orientación multiobjetivo. Como veremos, los resultados obtenidos son muy competitivos al compararlos con los de otros algoritmos multiobjetivo (P-ACO y P-VNS) del estado del arte en este campo. Los experimentos han sido realizados en instancias de test e instancias del mundo real (un total de 216 instancias agrupadas en 10 conjuntos), y hemos usado tres indicadores de calidad para comparar los resultados.

Este artículo se estructura de la siguiente manera. La sección II introduce al lector en la investigación desarrollada en este campo. La sección III presenta la definición formal del problema y su formulación matemática. La sección IV desgrana nuestra estrategia multiobjetivo para resolver el problema. La sección V muestra los resultados obtenidos y compara los resultados de nuestro algoritmo con los de otros algoritmos previamente publicados. Por último, la sección VI culmina este artículo y muestra también posibles líneas de trabajo futuras.

II. Trabajo Relacionado

La optimización multiobjetivo ha sido un campo importante, con bastantes trabajos de investigación, en las dos últimas décadas. Dentro de ella, la resolución de MOOP es una área de gran aplicabilidad, aunque con poca actividad de investigación.

Hemos basado nuestro trabajo en el algoritmo ABC y lo hemos adaptado al contexto multiobjetivo. La elección del ABC se debe a que ha sido extensamente estudiado y aplicado para resolver problemas reales en múltiples campos [3], incluyendo optimización mono-objetivo (p. ej. [4], [5] y [6]) y optimización multiobjetivo (p. ej. [7], [8] y [9])

Por otro lado, OP fue definido por [1], y algunos autores lo han considerado como un tipo de TSP (Problema del Viajante) con beneficios (ver [10]) o TSP selectivo. En OP, cada vértice tiene asociado algún beneficio, y la finalidad es visitar un grupo de vértices que maximice la suma de los beneficios, mientras se satisfaga la restricción de coste/longitud del tour. Dos estudios completos del estado del arte del problema de orientación se pueden encontrar en [11] y [12].

Relacionado con OP está el problema de orientación de equipo o TOP (ver [13] y [14]), donde el problema se extiende a múltiples tours.

Con respecto a MOOP, el problema abordado en este trabajo, muy pocas propuestas se pueden encontrar en la

literatura. Dentro del problema de orientación biobjetivo podemos destacar [15] que usó el algoritmo P-ACO (*Pareto Ant Colony Optimization*) y el algoritmo P-VNS (*Pareto Variable Neighborhood Search*) combinados con PR (*Path Relinking*), [16] que usó un algoritmo evolutivo también combinado con PR y [17] que utilizó GRASP combinado con PR.

III. Definición del Problema MOOP

Puede modelarse como un grafo dirigido $G = (V, A)$ con un conjunto de vértices, $V = \{v_0, v_1, v_2, ..., v_{n+1}\}$, y un conjunto de aristas, $A = \{(v_i, v_j) : v_i, v_j \in V \wedge v_i \neq v_j \wedge v_i \neq v_{n+1} \wedge v_j \neq v_0\}$. Cada vértice $v_i \in V \backslash \{v_0, v_{n+1}\}$ tiene asociados K beneficios b_{ik} $(k = 1, ..., K)$. Los vértices inicial y final, v_0 y v_{n+1}, no tienen beneficios asociados. Además, cada arista $(v_i, v_j) \in A$ tiene un coste c_{ij} que puede ser interpretado como distancia, dinero o tiempo invertido para ir desde v_i hasta v_j.

En todas las instancias usadas en este trabajo, v_0 y v_{n+1} representan el mismo punto, por lo tanto, entendemos que cada solución es un "tour" en lugar de "ruta". La finalidad del problema de orientación multiobjetivo es encontrar los mejores tours (que maximicen los beneficios para todos los objetivos) desde v_0 hasta v_{n+1}, cumpliendo la restricción marcada de coste/longitud C_{max}.

Podemos definir matemáticamente el problema como:

$$maximizar \quad F(x) = (f_1(x), ..., f_K(x)), \tag{1}$$

$$f_k = \sum_{v_i \in V \backslash \{v_0, v_{n+1}\}} (b_{ik} \cdot y_i) \quad (k = 1, ..., K), \tag{2}$$

donde y_i, una variable binaria, toma valor 1 cuando v_i es visitado, y 0 en caso contrario. Además, hay que tener en cuenta que:

$$\sum_{v_i \in V \backslash \{v_j\}} a_{ij} = y_j \quad (v_j \in V \backslash \{v_0\}), \tag{3}$$

$$\sum_{v_j \in V \backslash \{v_i\}} a_{ij} = y_i \quad (v_i \in V \backslash \{v_{n+1}\}), \tag{4}$$

$$\sum_{\{v_i, v_j\} \in S} a_{ij} \leq |S| - 1 \quad (S \subseteq V \wedge S \neq \emptyset), \tag{5}$$

$$y_0 = y_{n+1} = 1, \tag{6}$$

$$\sum_{(v_i, v_j) \in A} c_{ij} \cdot a_{ij} \leq C_{max}, \tag{7}$$

$$a_{ij} \in \{0, 1\} \quad \Big((v_i, v_j) \in A \Big), \tag{8}$$

$$y_i \in \{0, 1\} \quad (v_i \in V). \tag{9}$$

La variable binaria a_{ij} toma valor 1 cuando $(v_i, v_j) \in A$ es usado, y 0 en caso contrario. La ecuación 1 indica que para resolver MOOP hay que maximizar las diferentes funciones objetivo. La ecuación 2 define cada función objetivo como la suma de sus beneficios correspondientes. Las ecuaciones 3 y 4 implican que para cada vértice visitado solo existe una arista entrante y una arista saliente. La ecuación 5 impide subtours. La ecuación 6 implica que los puntos de comienzo y fin se usan en todos los tours. La ecuación 7 garantiza que el coste del tour no es superior al límite establecido C_{max}. En este caso, resolveremos el problema de orientación biobjetivo, por lo que $K = 2$.

IV. Solución. Nuevo Algoritmo Multiobjetivo

En [18] se demostró que OP es NP-hard. Por ello, es necesario aplicar técnicas metaheurísticas en la resolución del problema de orientación multiobjetivo.

En esta sección explicamos el algoritmo MOABC que hemos diseñado y desarrollado. Inicialmente el algoritmo Colonia de Abejas Artificiales fue propuesto por [2] en el contexto mono-objetivo. Nosotros lo hemos adaptado al contexto multiobjetivo y particularmente a la resolución del problema de orientación multiobjetivo. Modelamos cada solución en MOOP como una lista de puntos (aquellos que se han incluido en el tour correspondiente), lo cual es la manera más natural para representar una solución para este problema.

IV-A. Optimización Multiobjetivo

Debido a la naturaleza multiobjetivo del problema a resolver, es muy difícil elegir una solución óptima donde todos los objetivos se maximicen. Sin embargo, si restringimos a soluciones no dominadas, la elección se limitará a un conjunto razonable de soluciones candidatas. Las siguientes definiciones ayudarán a clarificar este aspecto.

Una solución x domina a una solución x' si x no es peor que x' en ninguna de las funciones objetivo, y es mejor en al menos una de las funciones objetivo. Formalmente: para maximizar $F(x) = (f_1(x), ..., f_K(x))$, x domina x' si $f_k(x) \geq f_k(x')$ para todo $k = 1, ..., K$, y $f_k(x) > f_k(x')$ para al menos un k. Si esto ocurre, escribimos $x \succ x'$.

Si ninguna solución domina a la solución x^*, decimos que x^* es no dominada o Pareto eficiente. En este caso, decimos que $z^* = F(x^*) = (f_1(x^*), ..., f_K(x^*))$ es un vector no dominado. El conjunto de todos los vectores no dominados es llamado frente de Pareto. La relación $\succ$ puede ser extendida desde el espacio de soluciones al espacio de objetivos. En ese caso, dados dos vectores $z = (z_1, ..., z_K)$ y $z' = (z'_1, ..., z'_K)$, escribimos $z \succ z'$ si $z_k \geq z'_k$ para todo $k = 1, ..., K$ y $z_k > z'_k$ en al menos un k.

IV-B. MOABC

El algoritmo ABC es un algoritmo metaheurístico, basado en inteligencia de enjambre, que se inspira en el comportamiento alimenticio de las colonias de abejas melíferas y se compone de tres componentes principales: abejas obreras, abejas observadoras y abejas exploradoras. Las abejas obreras buscan fuentes de alimento (soluciones). Existen otros dos comportamientos interesantes que se requieren para la auto-organización y la inteligencia de enjambre: enviar nuevas abejas (abejas observadoras) hacia las fuentes de alimento prometedoras (feedback positivo) y abandonar fuentes de alimento agotadas (feedback negativo), generando abejas exploradoras.

Como hemos mencionado anteriormente, hemos adaptado el algoritmo ABC original al contexto multiobjetivo y en particular a la resolución de MOOP. Un pseudocódigo de nuestro

algoritmo MOABC se muestra en algoritmo 1. Las siguientes subsecciones detallan todas las partes de este algoritmo.

Algoritmo 1 Colonia de Abejas Artificiales Multiobjetivo

entrada: T_{max} (tiempo máximo de ejecución), TP (tamaño de la población) y $limite$ (criterio de abandono)
salida: $Fichero\_SND$ (fichero con las soluciones no dominadas)

1: $t \leftarrow 0$; $Fichero\_SND \leftarrow \emptyset$
2: IniciarFuentesAlimento(TP) ▷ ver IV-C
3: **while** $t < T_{max}$ **do**
4: FaseAbejasObreras(TP) ▷ ver IV-D
5: CalcularCantidadNectar(TP) ▷ ver IV-E
6: FaseAbejasObservadoras(TP) ▷ ver IV-F
7: FaseAbejasExploradoras(TP, $limite$) ▷ ver IV-G
8: GuardarSND($Fichero\_SND$) ▷ ver IV-H
9: **end while**

IV-C. Iniciar Fuentes de Alimento

Todas las soluciones (fuentes de alimento) de la población (TP: tamaño de la población) se generan aleatoriamente por las abejas obreras siguiendo estos pasos: primero calculamos los 20 mejores movimientos desde cada punto a cualquier otro. Este cálculo solo se realiza una vez y se usa cuando sea necesario. Los mejores movimientos son calculados siguiendo la ecuación 10. Para ir desde i a j se calcula el $ratio_{ij}$, donde b_{j1} y b_{j2} son los beneficios del punto j y c_{ij} es el coste para ir desde i hasta j. Por lo tanto, la ecuación 10 relaciona los beneficios con el coste. Cabe destacar que los valores b_{j1}, b_{j2} y c_{ij} son normalizados para evitar sesgos. Los movimientos con mejor ratio se consideran mejores movimientos.

$$ratio_{ij} = \frac{(b_{j1} + b_{j2})}{c_{ij}}. \tag{10}$$

Después, comenzando por el punto inicial, seleccionamos un movimiento de la lista de 20 movimientos. Este proceso se repite hasta que no se pueden añadir más puntos por la restricción de coste C_{max}. Indicar que todos los tours finalizan en el punto de fin y que el resto de restricciones son también comprobadas durante la generación aleatoria de una solución.

IV-D. Fase Abejas Obreras

Las abejas obreras buscan mejores soluciones, dentro de sus vecindarios. Esto consiste en encontrar una solución vecina y evaluar su calidad. Para obtener soluciones vecinas para las abejas obreras, usamos dos operadores: acortamiento e inserción. El acortamiento intenta reordenar los vértices, dentro de un tour, para minimizar su coste. En este caso usamos el operador de acortamiento 2-opt propuesto por [19]. La inserción comprueba si es factible insertar un nuevo punto después de realizar el acortamiento. Para cada posible inserción, se elige de manera aleatoria uno de los mejores 10 puntos candidatos (según ecuación 10) de los puntos no visitados. Cualquier posible inserción se realiza teniendo en cuenta la restricción de coste C_{max}. Después de generar la solución vecina, se compara con la solución actual y si la primera domina a la segunda, entonces se reemplaza la solución actual para la abeja obrera.

IV-E. Calcular Cantidad de Néctar

Usamos dos operadores multiobjetivo para determinar cuáles son las mejores soluciones asociadas a las abejas obreras: *rank* y *crowding*. El primero clasifica las soluciones en los diferentes frentes de Pareto acorde a sus relaciones de dominancia, mientras que el segundo tiene en cuenta la distancia de amontonamiento, que consigue variedad en las soluciones. Una explicación más detallada de estos dos operadores puede encontrarse en [20]. Una vez calculados, los combinamos para cada solución en un único valor (llamado fitness multiobjetivo o $MOfitness$), según la ecuación 11. $MOfitness$ es un valor importante porque un mayor fitness multiobjetivo implicará una mayor probabilidad de selección, fundamental en la siguiente subsección.

$$MOfitness(x) = \frac{1}{2^{rank(x)} + \frac{1}{1+crowding(x)}}. \tag{11}$$

IV-F. Fase Abejas Observadoras

Las abejas obreras comparten la información del fitness de sus soluciones con las abejas observadoras, y estas eligen sus soluciones de manera probabilística basándose en las probabilidades calculadas en base a los valores $MOfitness$ (ver ecuación 11). Aplicamos el método de selección de ruleta basado en fitness propuesto por [21]. Por ello, las mejores soluciones reclutarán más abejas observadoras (feedback positivo).

La probabilidad $p(x^i)$ por la que una solución x^i es elegida por una abeja observadora se calcula con la ecuación 12.

$$p(x^i) = \frac{MOfitness(x^i)}{\sum_{m=1}^{TP} MOfitness(x^m)}. \tag{12}$$

Después de que una solución x^i sea probabilísticamente elegida por una abeja observadora, se genera una solución vecina usando dos operadores: intercambio de vértices y acortamiento. Intercambio de vértices intenta reemplazar cada punto existente del tour por un punto no visitado. En cada posición, el punto no visitado se selecciona a través de los mejores 10 puntos (de acuerdo a la ecuación 10) no visitados para esa posición. Después, se usa la misma operación de acortamiento de la subsección IV-D para mejorar la solución.

Tras generar la solución vecina, se compara con la solución original y si la primera no es dominada por la segunda, entonces se reemplaza la solución original para la abeja observadora.

IV-G. Fase Abejas Exploradoras

Las abejas obreras y observadoras cuyas soluciones no pueden ser mejoradas después de un número de intentos, indicado por un parámetro de configuración del algoritmo MOABC llamado $limite$, se convierten en abejas exploradoras y sus soluciones son abandonadas. Destacar que MOABC utiliza más abejas exploradoras que el algoritmo ABC original, donde se utiliza

como mucho una por iteración [22]. En MOABC todas las abejas obreras y observadoras con intentos $>= limite$ se convierten en exploradoras. En todo caso, las comparaciones con P-ACO y P-VNS (subsección V-C) son justas ya que se basan en la utilización del mismo tiempo de ejecución. Las abejas exploradoras buscan de manera aleatoria nuevas soluciones (mejorando la exploración del algoritmo). Por lo tanto, aquellas soluciones que no pueden ser mejoradas por explotación, son abandonadas (feedback negativo). La nueva solución asociada a la abeja exploradora se genera aleatoriamente de la misma manera que en la subsección IV-C. Posteriormente, la abeja exploradora intenta mejorar la solución aplicando los tres operadores vistos anteriormente: inserción, intercambio y acortamiento.

IV-H. Guardar Soluciones No Dominadas

En cada iteración, las soluciones no dominadas se guardan en un fichero para preservarlas mientras se generan nuevas soluciones. El operador utilizado para calcular las soluciones no dominadas es el *rank*, el cual clasifica las soluciones en diferentes frentes de Pareto de acuerdo a sus relaciones de dominancia. Además se usa el operador *crowding* que junto con el operador *rank* es usado para ordenar las soluciones actuales. Una explicación más detallada de estos operadores puede encontrarse en [20]. Después, las mejores TP (tamaño de la población) soluciones son tomadas por las abejas obreras para la siguiente iteración y las soluciones no dominadas (con $rank = 0$) se guardan en *Fichero_SND*.

V. Resultados

Hemos implementado nuestro algoritmo MOABC en C++, con el compilador GNU g++ 4.8.4 en Ubuntu Server 14.04 LTS 64 bits. Todas las ejecuciones se han realizado en un procesador Intel Pentium 4 D de 3,2 GHz y 1 GB de RAM, un entorno computacional muy similar al usado para los algoritmos P-ACO y P-VNS.

V-A. Instancias de Comparación

Hemos utilizado las mismas instancias que fueron usadas por [15] y que se encuentran disponibles en http://prolog.univie.ac.at/research/OP. Los conjuntos de instancias se pueden categorizar en 10 grupos, de acuerdo al número de vértices. En cada grupo existen varias instancias con distinta restricción de longitud máxima de tour (C_{max}). Los primeros 3 grupos (2_p21, 2_p32 y 2_p33) con 21, 32 y 33 vértices, y 11, 18 y 20 instancias respectivamente fueron creados por [1]. Los 2 siguientes grupos (2_p64 y 2_p66) con 64 y 66 vértices, y 14 y 26 instancias fueron propuestos por [23]. Estos 5 grupos fueron creados por sus autores con un único objetivo (beneficio), por lo que [15] añadió un segundo objetivo. La distancia entre vértices se calcula usando la fórmula de la distancia Euclídea. Estos 5 grupos pueden considerarse como instancias de test (benchmark).

Los siguientes 5 grupos fueron creados totalmente por [15] y son instancias reales de algunas ciudades y regiones europeas. Dentro de estos 5 grupos tenemos 97 y 273 vértices (2_p97 y 2_p273) con 20 instancias cada uno y con una restricción de distancia máxima de tour (C_{max}) entre 1 y 20 kilómetros; y 559, 562 y 2143 vértices (2_p559, 2_p562 y 2_p2143) con 29 instancias y una C_{max} desde 10 hasta 150 kilómetros. Todas estas se proporcionan con una matriz de distancias entre vértices. Además, para estos 5 grupos de instancias se usa una distancia de servicio entre vértices de 0,5 kilómetros que se puede entender como el tiempo necesario para visitar un POI.

V-B. Indicadores o Métricas de Calidad

Hemos usado el software disponible en http://www.tik.ee.ethz.ch/sop/pisa/?page=assessment.php para calcular los indicadores, según las instrucciones proporcionadas por [24]. Todos los indicadores usados en este trabajo han sido calculados usando valores normalizados de las funciones objetivo. Para algunos indicadores unarios se necesita un conjunto de referencia, por lo que hemos generado un conjunto de referencia con las soluciones no dominadas de todas las ejecuciones de los algoritmos a comparar (MOABC, P-ACO y P-VNS). Hemos usado las soluciones proporcionadas por [15] para cada instancia calculada por los algoritmos P-ACO y P-VNS, y realizado una comparación de acuerdo a varios indicadores o métricas utilizadas habitualmente en la literatura de optimización multiobjetivo: hipervolumen, épsilon unario y R3. Cada uno de estos tres indicadores se basa en diferentes propiedades, lo que permite una evaluación completa y balanceada que incrementa su fiabilidad. Puede encontrarse información detallada sobre los indicadores hipervolumen, épsilon unario y R3 en [25], [26] y [27] respectivamente.

V-C. Comparación entre Algoritmos

En esta subsección, comparamos los resultados obtenidos por MOABC con los resultados de los algoritmos P-ACO y P-VNS [15]. Hemos usado el mismo número de ejecuciones independientes que fueron usadas en esos algoritmos, 10 ejecuciones por instancia. También hemos usado los resultados obtenidos por P-ACO y P-VNS en cada instancia, los cuales utilizaron los parámetros de configuración más óptimos, como puede comprobarse en [15]. En el caso de nuestro algoritmo MOABC, siguiendo las recomendaciones de [28], hemos estudiado no solo el parámetro TP (tamaño de la población), sino también el parámetro $limite$ (criterio de abandono). Después de realizar tests preliminares con un rango de diferentes valores para cada parámetro, hemos establecido la configuración de los parámetros MOABC a un TP de 60 y un $limite$ de 10. Todos los algoritmos fueron parados tras consumir el mismo tiempo de ejecución, el usado por el algoritmo P-ACO. Los tiempos de ejecución y resultados se encuentran disponibles en http://prolog.univie.ac.at/research/OP para todas las ejecuciones e instancias de los algoritmos P-ACO y P-VNS.

En primer lugar, mostramos una comparación de los resultados obtenidos para el indicador hipervolumen (I_H). La tabla I muestra los resultados de los tres algoritmos para cada conjunto de instancias.

Tabla I
Resultados del indicador hipervolumen. Los valores en negrita indican superioridad. Los valores altos son mejores.

Conjuntos	I_H(MOABC)	I_H(P-ACO)	I_H(P-VNS)
2_p21	**3,525**	3,390	3,471
2_p32	3,366	**3,375**	3,250
2_p33	3,575	**3,638**	3,615
Media (Pequeño)	**3,489**	3,468	3,445
2_p64	3,363	**3,586**	3,569
2_p66	3,346	**3,609**	3,554
2_p97	**3,501**	3,315	3,335
Media (Mediano)	3,403	**3,503**	3,486
2_p273	**3,472**	3,329	3,320
2_p559	**3,648**	3,415	3,441
2_p562	**3,634**	3,451	3,370
2_p2143	**3,640**	3,144	2,839
Media (Grande)	**3,598**	3,335	3,242
Media (Todos)	**3,497**	3,435	3,391

La tabla I muestra la media de los resultados obtenidos por conjunto de instancia. Por ejemplo, el conjunto de instancias 2_p21 tiene 11 instancias, por lo tanto, esta tabla muestra la media de los resultados de 110 ejecuciones. Además, hemos agrupado los conjuntos de instancias en tres grupos, teniendo en cuenta sus tamaños (número de vértices): pequeño, mediano y grande; y mostrado su media de resultados. Finalmente, mostramos la media para todas las instancias. En instancias pequeñas, MOABC obtiene la mejor media de hipervolumen, y consigue mejores resultados en 2_p21 que P-ACO y P-VNS. En instancias medianas, el algoritmo MOABC obtiene el mejor hipervolumen para 2_p97. En instancias grandes, el algoritmo MOABC es definitivamente superior que P-ACO y P-VNS, obteniendo los mejores hipervolúmenes para todas las instancias grandes (las más complicadas). De hecho, es aquí donde las diferencias entre los tres algoritmos son mayores, especialmente en el conjunto de instancias con 2143 vértices (2_p2143). Como conclusión, en la tabla I podemos ver que MOABC tiene la mejor media global de hipervolumen.

Tabla II
Resultados del indicador épsilon unario. Los valores en negrita indican superioridad. Los valores bajos son mejores.

Conjuntos	I_ϵ(MOABC)	I_ϵ(P-ACO)	I_ϵ(P-VNS)
2_p21	**1,000**	1,108	1,034
2_p32	**1,024**	1,028	1,124
2_p33	1,074	**1,032**	1,050
Media (Pequeño)	**1,032**	1,056	1,069
2_p64	1,085	**1,036**	1,038
2_p66	1,104	**1,033**	1,049
2_p97	**1,000**	1,158	1,137
Media (Mediano)	**1,063**	1,076	1,074
2_p273	**1,072**	1,196	1,161
2_p559	**1,037**	1,118	1,113
2_p562	**1,039**	1,112	1,128
2_p2143	**1,076**	1,188	1,265
Media (Grande)	**1,056**	1,153	1,167
Media (Todos)	**1,050**	1,095	1,103

En segundo lugar, y usando la misma metodología que en el indicador anterior, la tabla II muestra los resultados para los tres algoritmos y su comparación basada en el indicador épsilon unario (I_ϵ). En pequeñas instancias, MOABC obtiene la mejor media en el indicador épsilon, y también obtiene mejores resultados para los conjuntos de instancias 2_p21 y 2_p32 que P-ACO y P-VNS. En instancias medianas, el algoritmo MOABC nuevamente consigue la mejor media en el indicador épsilon, y también los mejores resultados para 2_p97. En instancias grandes, MOABC es claramente mejor que P-ACO y P-VNS ya que obtiene los mejores resultados para todos los conjuntos de instancias grandes (las más complejas). Analizando la tabla II podemos observar que MOABC tiene la mejor media global del indicador épsilon, consiguiendo el mejor resultado en 7 de 10 conjuntos de instancias.

Tabla III
Resultados del indicador R3. Los valores en negrita indican superioridad. Los valores bajos son mejores.

Conjuntos	I_{R3}(MOABC)	I_{R3}(P-ACO)	I_{R3}(P-VNS)
2_p21	**0,000**	0,031	0,010
2_p32	**0,003**	**0,003**	0,030
2_p33	0,018	**0,005**	0,010
Media (Pequeño)	**0,007**	0,013	0,017
2_p64	0,032	0,011	**0,007**
2_p66	0,039	**0,008**	0,011
2_p97	**0,000**	0,046	0,041
Media (Mediano)	0,024	0,022	**0,020**
2_p273	**0,030**	0,067	0,063
2_p559	**0,011**	0,054	0,048
2_p562	**0,010**	0,041	0,059
2_p2143	**0,038**	0,135	0,204
Media (Grande)	**0,022**	0,074	0,093
Media (Todos)	**0,018**	0,036	0,043

Finalmente, y usando la misma metodología, la tabla III muestra los resultados para los tres algoritmos y su comparación del indicador R3 (I_{R3}). En pequeñas instancias, MOABC obtiene la mejor media del indicador R3, y también los mejores resultados para los conjuntos de instancias 2_p21 y 2_p32. En instancias medianas, MOABC consigue los mejores resultados para el conjunto de instancias 2_p97. En instancias grandes, MOABC es definitivamente mejor que P-ACO y P-VNS, ya que como en los anteriores indicadores consigue los mejores resultados para todas las instancias grandes. Teniendo en cuenta la tabla III, podemos afirmar que MOABC tiene la mejor media global del indicador R3, obteniendo los mejores resultados para 7 de 10 conjuntos de instancias.

Resumiendo los resultados de las tres tablas anteriores podemos concluir que:

- MOABC tiene la mejor media global para los tres indicadores (I_H, I_ϵ y I_{R3}).
- En las instancias pequeñas, MOABC obtiene la mejor media para los tres indicadores, y generalmente, proporciona mejores resultados que P-ACO y P-VNS en 2 de 3 conjuntos de instancias (2_p21 y 2_p32).
- En las instancias medianas, de acuerdo con los tres indicadores, MOABC claramente supera a los otros dos algoritmos (P-ACO y P-VNS) en el conjunto de instan-

cias 2_p97 (uno de los tres conjuntos de instancias del grupo).

- En las instancias grandes, MOABC es el mejor algoritmo, teniendo en cuenta los tres indicadores, sus medias correspondientes, y también los resultados individuales para cada uno de los 4 conjuntos de instancias de este grupo. Las diferencias respecto a los tres algoritmos son especialmente mayores en el último conjunto de instancias con 2143 vértices (el más difícil).

VI. Conclusiones

Hemos desarrollado un enfoque metaheurístico para solucionar el problema de orientación multiobjetivo. Nuestra propuesta, MOABC, es un algoritmo de optimización multiobjetivo basado en el comportamiento de las abejas melíferas. MOABC ha sido comparado con dos algoritmos multiobjetivo que fueron publicados anteriormente, P-ACO y P-VNS. Las comparaciones fueron realizadas usando instancias de test e instancias del mundo real del problema de orientación, con un total de 216 instancias agrupadas en 10 conjuntos. Utilizamos tres indicadores de calidad multiobjetivo, utilizados habitualmente en la literatura, para comparar los resultados de los tres algoritmos. MOABC es mejor en general que P-ACO y P-VNS, de acuerdo a la media de los indicadores y sensiblemente mejor cuanto mayor (más compleja) es la instancia del problema a resolver (mayor número de vértices). Por lo tanto, podemos concluir que MOABC es una aproximación muy interesante para resolver el problema de orientación.

En el futuro intentaremos aplicar este enfoque a un planificador de rutas turísticas. Además, trataremos de mejorar estos resultados usando otras estrategias distintas (p. ej.: otras variantes de ABC u otros algoritmos evolutivos). Su diseño, implementación y ejecución, y la comparación con algoritmos multiobjetivo es interesante para futuras investigaciones. Finalmente, la aplicación de MOABC en otros tipos de problemas de orientación también supone una línea interesante de trabajo.

Agradecimientos

Este trabajo ha sido parcialmente financiado por la AEI (Agencia Estatal de Investigación, España) y el FEDER (Fondo Europeo de Desarrollo Regional, UE), bajo el proyecto TIN2016-76259-P (proyecto PROTEIN). Gracias también a la Junta de Extremadura por la ayuda GR18090 otorgada al grupo de investigación TIC015.

Referencias

[1] T. Tsiligirides, Heuristic methods applied to orienteering, Journal of the Operational Research Society 35 (9) (1984) 797–809.

[2] D. Karaboga, B. Basturk, A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm, Journal of Global Optimization 39 (3) (2007) 459–471.

[3] D. Karaboga, B. Gorkemli, C. Ozturk, N. Karaboga, A comprehensive survey: artificial bee colony (ABC) algorithm and applications, Artificial Intelligence Review 42 (1) (2014) 21–57.

[4] Y. Zhang, L. Wu, Optimal multi-level thresholding based on maximum Tsallis entropy via an artificial bee colony approach, Entropy 13 (4) (2011) 841–859.

[5] Y. Zhang, L. Wu, S. Wang, Magnetic resonance brain image classification by an improved artificial bee colony algorithm, Progress In Electromagnetics Research 116 (2011) 65–79.

[6] S. Wang, Y. Zhang, Z. Dong, S. Du, G. Ji, J. Yan, J. Yang, Q. Wang, C. Feng, P. Phillips, Feed-forward neural network optimized by hybridization of PSO and ABC for abnormal brain detection, International Journal of Imaging Systems and Technology 25 (2) (2015) 153–164.

[7] W. Zou, Y. Zhu, H. Chen, B. Zhang, Solving multiobjective optimization problems using artificial bee colony algorithm, Discrete Dynamics in Nature and Society 2011 (2011) 569784.

[8] H. Zhang, Y. Zhu, W. Zou, X. Yan, A hybrid multi-objective artificial bee colony algorithm for burdening optimization of copper strip production, Applied Mathematical Modelling 36 (6) (2012) 2578–2591.

[9] J. Huo, L. Liu, An improved multi-objective artificial bee colony optimization algorithm with regulation operators, Information 8 (1) (2017) 18.

[10] D. Feillet, P. Dejax, M. Gendreau, Traveling salesman problems with profits, Transportation Science 39 (2) (2005) 188–205.

[11] A. Gunawan, H. C. Lau, P. Vansteenwegen, Orienteering problem: A survey of recent variants, solution approaches and applications, European Journal of Operational Research 255 (2) (2016) 315–332.

[12] P. Vansteenwegen, W. Souffriau, D. V. Oudheusden, The orienteering problem: A survey, European Journal of Operational Research 209 (1) (2011) 1–10.

[13] P. Vansteenwegen, W. Souffriau, G. V. Berghe, D. V. Oudheusden, A guided local search metaheuristic for the team orienteering problem, European Journal of Operational Research 196 (1) (2009) 118–127.

[14] I.-M. Chao, B. L. Golden, E. A. Wasil, The team orienteering problem, European Journal of Operational Research 88 (3) (1996) 464–474.

[15] M. Schilde, K. F. Doerner, R. F. Hartl, G. Kiechle, Metaheuristics for the bi-objective orienteering problem, Swarm Intelligence 3 (3) (2009) 179–201.

[16] D. Purevsuren, G. Cui, S. ur Rehman, Evolutionary multiobjective optimization algorithms with path relinking for bi-orienteering problem, in: Software Engineering and Service Science (ICSESS), 2015 6th IEEE International Conference on, 2015, pp. 132–135.

[17] R. Martí, V. Campos, M. G. Resende, A. Duarte, Multiobjective GRASP with path relinking, European Journal of Operational Research 240 (1) (2015) 54–71.

[18] B. L. Golden, L. Levy, R. Vohra, The orienteering problem, Naval Research Logistics (NRL) 34 (3) (1987) 307–318.

[19] G. A. Croes, A method for solving traveling-salesman problems, Operations Research 6 (6) (1958) 791–812.

[20] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II, IEEE Transactions on Evolutionary Computation 6 (2) (2002) 182–197.

[21] D. E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1989.

[22] M. Mernik, S.-H. Liu, D. Karaboga, M. Črepinšek, On clarifying misconceptions when comparing variants of the Artificial Bee Colony algorithm by offering a new implementation, Information Sciences 291 (2015) 115–127.

[23] I.-M. Chao, B. L. Golden, E. A. Wasil, A fast and effective heuristic for the orienteering problem, European Journal of Operational Research 88 (3) (1996) 475–489.

[24] J. Knowles, L. Thiele, E. Zitzler, A tutorial on the performance assessment of stochastic multiobjective optimizers, Technical report no. 214, Computer Engineering and Networks Laboratory (TIK), ETH Zurich, Switzerland, revised version (Feb 2006).

[25] N. Beume, C. M. Fonseca, M. López-Ibáñez, L. Paquete, J. Vahrenhold, On the complexity of computing the hypervolume indicator, IEEE Transactions on Evolutionary Computation 13 (5) (2009) 1075–1082.

[26] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, V. G. da Fonseca, Performance assessment of multiobjective optimizers: an analysis and review, IEEE Transactions on Evolutionary Computation 7 (2) (2003) 117–132.

[27] M. P. Hansen, A. Jaszkiewicz, Evaluating the quality of approximations to the non-dominated set, Tech. Rep. IMM-REP-1998-7, Institute of Mathematical Modelling, Technical University of Denmark (1998).

[28] N. Veček, S.-H. Liu, M. Črepinšek, M. Mernik, On the importance of the artificial bee colony control parameter 'limit', Information Technology and Control 46 (4) (2017) 566–604.