Revista Cubana de Ciencias Informáticas
Vol. 12, No. Especial UCIENCIA, Septiembre 2018
ISSN: 2227-1899 | RNPS: 2301
Pág. 28-41
http://rcci.uci.cu

Tipo de artículo: Artículo original
Temática: Matemática Computacional
Recibido: 04/06/2018 | Aceptado: 10/09/2018

Metaheurística GRASP para el problema Vertex Bisection Minimization

GRASP metaheuristics for the Vertex Bisection Minimization problem

Jorge Moreno Ramírez[1*]

[1]Departamento de Ciencia de la Computación, Universidad Federal Fluminense, 24210-240, Brasil

*Autor para correspondencia: jmoreno@ic.uff.br

Resumen
Dado un grafo no orientado $G = (V, E)$, donde V denota el conjunto de vértices y E representa el conjunto de aristas, el problema Vertex Bisection Minimization consiste en particionar el conjunto V en dos subconjuntos B y B', de manera que $|B| = \lfloor |V|/2 \rfloor$ y se minimice el número de vértices en B que son adyacentes a algún vértice de B'. Este problema pertenece al conjunto de los problemas de diseño de grafos y tiene aplicaciones en áreas como optimización de redes, teoría de grafos, recuperación de información, etc. Este problema es NP-difícil sobre grafos en general, aunque polinomialmente soluble para árboles e hipercubos. Por su importancia, diversos enfoques heurísticos han sido realizados con el propósito de encontrar soluciones de calidad. En este trabajo fue desarrollada una metaheurística GRASP para abordar este problema. Los resultados experimentales muestran que el algoritmo propuesto obtiene resultados de mejor calidad que los algoritmos heurísticos encontrados en la literatura para este problema.

Palabras claves: Metaheurística, GRASP, Bisección de vértices

Abstract
Given a non-oriented graph $G = (V, E)$, where V denotes the set of vertices and E represents the set of edges, the Vertex Bisection Minimization problem consists of partitioning V into two subsets B and B', such that $|B| = \lfloor |V|/2 \rfloor$ and minimizing the number of vertices in B that are adjacent to some vertex of B'. This problem belongs to the set of graph layout problems and has applications in areas such as network optimization, graph theory, information retrieval, etc. This problem is NP-hard on graphs in general, but polynomially soluble for trees and hypercubes. Because of its importance, various heuristic approaches have been carried out with the purpose of finding quality solutions. In this work, a GRASP metaheuristics was developed to address this problem. The experimental results show that the proposed algorithm obtains better quality results than the heuristic algorithms found in the literature for this problem.

Keywords: *Metaheuristics, GRASP, Vertex Bisection*

Revista Cubana de Ciencias Informáticas
Vol. 12, No. Especial UCIENCIA, Septiembre 2018
ISSN: 2227-1899 | RNPS: 2301
Pág. 28-41
http://rcci.uci.cu

Introducción

Los problemas de diseño de grafos aparecen con frecuencia en áreas como optimización de redes, diseño de circuitos VLSI (*Very Large Scale Integration*), teoría de grafos, recuperación de información, etc (Díaz et˜al., 2002). Estos problemas son generalmente NP-difíciles, por lo que con frecuencia se recurre al uso de procedimientos heurísticos.

Dado un grafo $G = (V, E)$, donde V representa el conjunto de vértices y E denota el conjunto de aristas, el problema Vertex Bisection Minimization (VBM) consiste en particionar el conjunto de vértices del grafo en dos subconjuntos B y B', de manera que $|B| = \lfloor |V|/2 \rfloor$ y se minimice el número de vértices en B que son adyacentes a algún vértice de B' (Díaz et˜al., 2002). Este problema pertenece al conjunto de los problemas de diseño de grafos y ha sido abordado a través de métodos exactos y heurísticos al tratarse de un problema NP-difícil (Brandes and Fleischer, 2009). Un algoritmo exacto basado en ramificación y poda fue propuesto en (Jain et˜al., 2016a) para el problem VBM. Ese algoritmo consiguió resolver instancias con grafos de hasta 24 vértices. Otro algoritmo exacto, pero basado en la Programación Lineal en Enteros es presentado en (Jain et˜al., 2016b), resolviendo este problema en grafos de hasta 48 vértices. Desafortunadamente, el uso de algoritmos exactos se torna inviable sobre grafos de mayor tamaño. En ese sentido, varios algoritmos heurísticos han sido desarrollados para abordar este problema.

Algunas de las heurísticas para el problema Vertex Bisection Minimization están basadas en las heurísticas desarrolladas para el problema Graph Bisection. En ese problema, el objetivo es particionar el conjunto de vértices del grafo en dos subconjuntos, de manera que se minimice el número de aristas entre los subconjuntos. Entre las heurísticas desarrolladas para el problema Graph Bisection están una heurística basada en Simulated Annealing (Johnson et˜al., 1989), un algoritmo genético (Bui and Moon, 1996), así como un procedimiento basado en la metaheurística Colonia de Hormigas (Aguilar, 2016).

Un algoritmo memético fue propuesto para el problema Graph Bisection en (Galinier et˜al., 2011) con resultados relevantes para ese problema. En (Jain et˜al., 2016c) se implementa una adaptación de ese algoritmo para el problema VBM denominada AMAGP. En ese mismo artículo se presenta un nuevo algoritmo memético para el problema VBM denominado MAVBMP. El algoritmo MAVBMP usa diferentes heurísticas para generar la población inicial, así como diferentes operadores de cruzamiento. Otro de los elementos incorporados es un operador de mejora, cuyo objetivo es mejorar una solución mediante intercambios de vértices. A pesar de esto, este operador contiene muchas restricciones para ser aplicado y como consecuencia no puede ser usado en varias situaciones.

La metaheurística GRASP (*Greedy Randomize Adaptive Search Procedure*) ha sido usada exitosamente en numerosos problemas de optimización combinatoria (Mestria et˜al., 2013; Díaz et˜al., 2017). Esta metaheurística

Revista Cubana de Ciencias Informáticas
Vol. 12, No. Especial UCIENCIA, Septiembre 2018
ISSN: 2227-1899 | RNPS: 2301
Pág. 28-41
http://rcci.uci.cu

ha sido ampliamente estudiada y posee características deseables, como la posibilidad de ser implementada usando paralelismo y la simplicidad de su estructura. El objetivo de este trabajo fue desarrollar un procedimiento heurístico basado en la metaheurística GRASP que presente resultados competitivos con tiempos cortos de respuesta.

El resto de este documento está organizado como se explica a continuación. La siguiente sección describe los algoritmos implementados que constituyen la estructura básica de la metaheurística GRASP. A continuación son descritos los experimentos computacionales realizados y son analizados los resultados obtenidos. Finalmente, son presentadas las conclusiones del trabajo y las referencias bibliográficas del mismo.

Metaheurística desarrollada

La metaheurística Greedy Randomized Adaptive Search Procedure (GRASP) es un método para obtener soluciones de buena calidad en problemas de optimización combinatoria (Resende and Ribeiro, 2016). GRASP puede definirse como un proceso iterativo, donde en cada iteración se realiza una fase de construcción y una búsqueda local.

En la fase de construcción, la idea es combinar un algoritmo ávido con una componente aleatoria. Los algoritmos ávidos suelen ser generalmente iterativos, de manera que en cada iteración se seleccione el elemento que proporcione el mayor incremento de calidad a la solución que está siendo construida. Como consecuencia, este tipo de procedimiento no siempre obtiene el valor óptimo y en algunas instancias de problemas puede dar lugar a soluciones muy alejadas del valor óptimo. Este inconveniente es enfrentado por la metaheurística GRASP mediante la inclusión de una componente aleatoria en el procedimiento de construcción. La inclusión de esta componente aleatoria es realizada con el propósito de incorporar diversificación a las soluciones creadas, lo que pudiera conducir a buenos resultados.

Una vez que se genera una solución, la metaheurística GRASP emplea un procedimiento de búsqueda local. Este procedimiento consiste en buscar soluciones mejores en una vecindad de la solución generada en la fase constructiva. Este proceso de crear soluciones y realizar búsquedas locales se repite hasta alcanzar cierto criterio de parada establecido. Finalmente, el algoritmo devuelve como resultado la mejor solución encontrada entre todas las iteraciones.

Construcción de las soluciones iniciales

Para la construcción de las soluciones iniciales fueron utilizados dos métodos que son descritos a continuación. Ambos métodos están basados en la idea que los vértices en B deben tener la mayoría de sus vértices (o todos)

Revista Cubana de Ciencias Informáticas
Vol. 12, No. Especial UCIENCIA, Septiembre 2018
ISSN: 2227-1899 | RNPS: 2301
Pág. 28-41
http://rcci.uci.cu

Algorithm 1: CONST_INIC1

Input: Un grafo no orientado $G = (V, E)$ y el parámetro α
Output: El conjunto $B \subset V$

1 $B \leftarrow \emptyset$
2 **while** $|B| < \lfloor |V|/2 \rfloor$ **do**
3 $\quad f_{min} \leftarrow \text{mín}\{|N_{V \setminus B}(v)| \mid v \in V \setminus B\}$
4 $\quad f_{max} \leftarrow \text{máx}\{|N_{V \setminus B}(v)| \mid v \in V \setminus B\}$
5 $\quad LRC \leftarrow \{v \in V \setminus B \mid |N_{V \setminus B}(v)| \leq f_{min} + (f_{max} - f_{min}) \cdot \alpha\,)$
6 $\quad u \leftarrow$ elemento aleatorio de LRC
7 $\quad B \leftarrow B \cup \{u\}$
8 $\quad S \leftarrow N_{V \setminus B}(u)$
9 $\quad$ **si** $|B| + |S| \leq \lfloor |V|/2 \rfloor$ **entonces**
10 $\quad\quad B \leftarrow B \cup S$
11 $\quad$ **sino**
12 $\quad\quad S' \leftarrow$ subconjunto de S con $\lfloor |V|/2 \rfloor - |B|$ elementos
13 $\quad\quad B \leftarrow B \cup S'$
14 **retornar** B

adyacentes en B. El conjunto de vértices adyacentes a un vértice v y que pertenecen a un conjunto $S \subseteq V$ se denotará como $N_S(v)$.

El Algoritmo 1 presenta el primero de estos métodos, denominado como Const_Inic1. Este método está basado en la heurística ávida $H1$ (Jain et˜al., 2016c), pero introduciendo el uso de una componente aleatoria. Esta componente aleatoria se encuentra en el proceso de selección del vértice que será adicionado al conjunto B. La heurística $H1$ selecciona en cada paso el vértice con menor número de vértices adyacentes en $V \setminus B$. Para evitar esta selección golosa se crea una lista con aquellos vértices de B que tienen "pocos" vértices adyacentes en $V \setminus B$. Esto se conoce como *Lista Restricta de Candidatos* (LRC) y está condicionada a un parámetro α (línea 5). Si el parámetro $\alpha = 0$ se estaría en presencia de una selección ávida o golosa. Por otro lado, si $\alpha = 1$ se estaría en presencia de una selección aleatoria. Una vez creada la LRC, un elemento $u \in B$ es seleccionado al azar de esta lista (línea 6). Después de seleccionar aleatoriamente un elemento u de la LRC e insertarlo en B, son analizados los vértices adyacentes a u que no están en B. Si todos los vértices adyacentes a u pueden ser adicionados a B, entonces estos vértices son insertados en B (línea 10). De no ser posible, se adicionan a B tantos vértices adyacentes de u como sea posible hasta que $|B| = \lfloor |V|/2 \rfloor$ (líneas 12-13).

El otro procedimiento utilizado en este trabajo para generar soluciones iniciales es mostrado en el Algoritmo 2, denominado como Const_Inic2. Este algoritmo comienza adicionando al conjunto B un vértice seleccionado

Revista Cubana de Ciencias Informáticas
Vol. 12, No. Especial UCIENCIA, Septiembre 2018
ISSN: 2227-1899 | RNPS: 2301
Pág. 28-41
http://rcci.uci.cu

Algorithm 2: CONST_INIC2

Input: Un grafo no orientado $G = (V, E)$ y el parámetro α
Output: El conjunto $B \subset V$

1 $v \leftarrow$ elemento aleatorio de V
2 $B \leftarrow \{v\}$
3 **while** $|B| < \lfloor |V|/2 \rfloor$ **do**
4 $\quad f_{min} \leftarrow \text{mín}\{|N_B(v)| \mid v \in V \setminus B\}$
5 $\quad f_{max} \leftarrow \text{máx}\{|N_B(v)| \mid v \in V \setminus B\}$
6 $\quad LRC \leftarrow \{v \in V \setminus B \mid |N_B(v)| \geq f_{min} + (f_{max} - f_{min}) \cdot \alpha\,)$
7 $\quad u \leftarrow$ elemento aleatorio de LRC
8 $\quad B \leftarrow B \cup \{u\}$
9 **retornar** B

aleatoriamente. En su ciclo principal (líneas 3-9), el algoritmo selecciona un vértice de la lista de candidatos LRC, creada con aquellos vértices de $V \setminus B$ con "muchos" vértices adyacentes en B. A diferencia del algoritmo anterior, el algoritmo Const_Inic2 presenta un carácter ávido cuando $\alpha = 1$ y un carácter aleatorio cuando $\alpha = 0$. Como será mostrado más adelante, cada uno de los algoritmos Const_Inic1 y Const_Inic2 generará la mitad de las soluciones iniciales usadas por la metaheurística.

Búsqueda Local

Las soluciones generadas en la fase constructiva del GRASP pueden estar alejadas de la solución óptima. Por ese motivo es realizada una exploración por soluciones vecinas a la solución generada, con el objetivo de encontrar soluciones de mayor calidad.

La búsqueda local desarrollada en este trabajo usa la estrategia de *best improvement*. Esta estrategia consiste en actualizar la mejor solución encontrada con aquella solución vecina de mayor calidad.

Sea $S = (B, B')$ una solución para el problema VBM, una solución vecina S' se obtiene a partir de S, intercambiando un vértice en B por un vértice en B'. Para cada vértice $u \in B'$ se calcula $\phi(u)$, que representa cuántos vértices podrían ser eliminados del valor de la solución al pasar el vértice $u \in B'$ para el conjunto B. Posteriormente se calcula para cada vértice en $v \in B$ el valor $\sigma(v)$, que determina el número de vértices en B que serían adicionados al valor de la solución si se pasara $v \in B$ para el conjunto B'. Finalmente se determina el par $(v, u) \in (B, B')$ que minimice el valor $\psi(v, u) = \phi(u) - \sigma(v)$. Si el par encontrado $(v, u) \in (B, B')$ satisface que $\psi(v, u) > 0$, entonces la solución $S' = ((B \setminus \{v\}) \cup \{u\}, (B' \setminus \{u\}) \cup \{v\})$ será mejor que la solución S.

En (Jain et˜al., 2016c) se propone un operador de mejora que también intercambia vértices entre los conjuntos

Revista Cubana de Ciencias Informáticas
Vol. 12, No. Especial UCIENCIA, Septiembre 2018
ISSN: 2227-1899 | RNPS: 2301
Pág. 28-41
http://rcci.uci.cu

Algorithm 3: BUSQUEDA_LOCAL

Input: Una solución $S = (B, B')$
Output: La mejor solución en la vecindad de S

foreach $u \in B'$ **do**
  Calcular el valor $\phi(u)$
foreach $v \in B$ **do**
  Calcular el valor $\sigma(v)$
$(v, u) = argmin_{(s,t)\in(B,B')}\{\psi(s, t)\}$
si $\psi(v, u) > 0$ **entonces**
  $S' = ((B \setminus \{v\}) \cup \{u\}, (B' \setminus \{u\}) \cup \{v\})$
sino
  $S' = S$
retornar S'

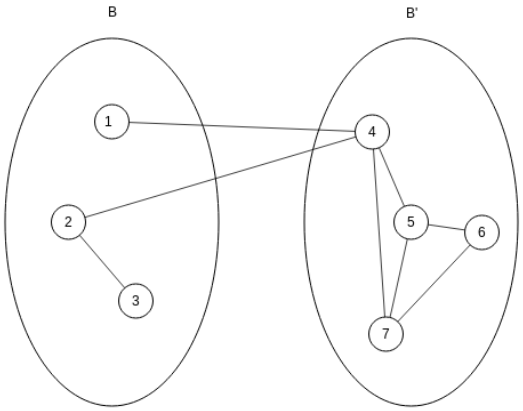

Figura 1: Una solución para el problema con dos vértices de B adyacentes a vértices en B'.

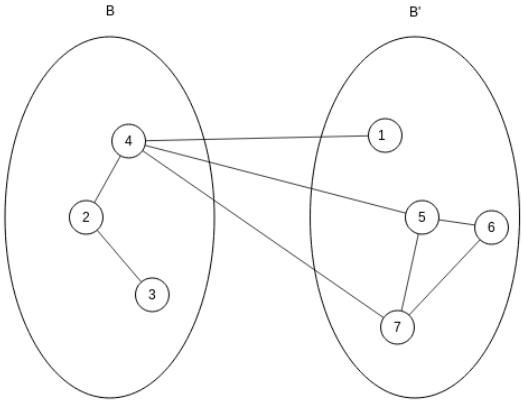

Figura 2: Solución después de la búsqueda local con un vértice de B adyacente a vértices en B'

B y B' pero con restricciones muy específicas. Dicho operador selecciona un vértice $u \in B'$ de manera que $N_{B'}(u) = \emptyset$. Ese vértice es intercambiado por un vértice $v \in B$ tal que todos los vértices adyacentes a v son adyacentes a algún vértice en B'. La Figura 2 muestra un ejemplo de la búsqueda local usada en el GRASP. En este caso, el operador de mejora definido en (Jain et˜al., 2016c) es incapaz de mejorar la calidad de la solución original mientras que la búsqueda local especificada en el Algoritmo 3 proporciona una solución mejor.

Revista Cubana de Ciencias Informáticas
Vol. 12, No. Especial UCIENCIA, Septiembre 2018
ISSN: 2227-1899 | RNPS: 2301
Pág. 28-41
http://rcci.uci.cu

Integración de los métodos

Como se expuso anteriormente, la metaheurística GRASP consta de una fase constructiva y de una fase de búsqueda local. Estos métodos fueron definidos en las subsecciones anteriores y son integrados en la metaheurística GRASP como se muestra en el Algoritmo 4. Inicialmente son construidas una solución ávida usando el método Const_Inic1 y una solución ávida usando el método Const_Inic2. Una vez aplicada la búsqueda local sobre cada una de estas soluciones, es seleccionada la solución de mejor calidad (línea 7). Durante la primera mitad de las iteraciones son generadas soluciones usando el método Const_Inic1, mientras que en la segundad mitad de las iteraciones se utiliza el método Const_Inic2. En el ciclo principal, cada vez que es generada una solución, se realiza la búsqueda local usando el método Busqueda_Local (líneas 15-16). Si la solución S' obtenida en la búsqueda local es mejor que la mejor solución encontrada S^*, entonces se actualiza el valor de S^* con S' (líneas 17-18).

El parámetro α es incrementado en 0.1 cada $\frac{MAXITER}{10}$ iteraciones (línea 20). De esta forma se usan los valores $\alpha \in \{0.1,0.2,0.3,0.4,0.5\}$ y $\alpha \in \{0.5,0.6,0.7,0.8,0.9\}$ para Const_Inic1 y Const_Inic2 respectivamente. Esta selección del parámetro α está basada en el hecho de que para valores superiores a 0.5 el algoritmo Const_Inic1 presenta un comportamiento más aleatorio. Por su parte, el algoritmo Const_Inic2 presenta un comportamiento más aleatorio para valores inferiores a 0.5.

Resultados y discusión

En esta sección se presentan los resultados experimentales obtenidos sobre varias de las instancias usadas en (Jain et˜al., 2016c). Estos experimentos fueron desarrollados sobre un computador Intel (R) Core(TM) i3-3110M CPU @ 2.40GHz, con 2 Gb de RAM sobre el sistema operativo Fedora 22. Todos los métodos fueron programados en el lenguaje C++ usando el compilador gcc y el número de iteraciones para el GRASP fue fijado en 100. La metaheurística GRASP fue comparada con los algoritmos AMAGP y MAVBMP, propuestos en (Jain et˜al., 2016c) e implementados en MATLAB 7.0 sobre un computador Dual Xeon, 6 con 24GB de RAM. Las instancias usadas para analizar el comportamiento del GRASP son clasificadas como:

Grafos pequeños: consiste en 23 grafos del conjunto de 84 grafos disponible en http://www.optsicom.es/vsp. El número de vértices de estos grafos varía entre 16 y 24 mientras que el número de aristas varía entre 18 y 34.

Grafos Harwell-Boeing: un subconjunto de los grafos disponibles en la biblioteca de dominio público Matrix Market.

Grafos estándares: un conjunto de grafos conteniendo grafos bipartitos e hipercubos. Los grafos bipartitos poseen diferentes dimensiones, con grafos pequeños como $K_{4,15}$ y otros de mayor tamaño como $K_{100,100}$. Por

Revista Cubana de Ciencias Informáticas
Vol. 12, No. Especial UCIENCIA, Septiembre 2018
ISSN: 2227-1899 | RNPS: 2301
Pág. 28-41
http://rcci.uci.cu

Algorithm 4: GRASP_VBM

Input: Un grafo no orientado $G = (V, E)$ y el número máximo de iteraciones $MAXITER$
Output: El conjunto $B^* \subset V$

$B_1 \leftarrow$Const_Inic1$(G, 0)$
$S_1 \leftarrow \{B_1, V \setminus B_1\}$
$S_1^* \leftarrow$Busqueda_Local$(B_1, V \setminus B_1)$
$B_2 \leftarrow$Const_Inic2$(G, 1)$
$S_2 \leftarrow \{B_2, V \setminus B_2\}$
$S_2^* \leftarrow$Busqueda_Local(S_2)
$S^* \leftarrow$ mejor solución entre S_1^* y S_2^*
$num\_iter \leftarrow 0$
$\alpha \leftarrow 0.1$
while $num\_iter < MAXITER$ **do**
    si $num\_iter < \frac{MAXITER}{2}$ **entonces**
        $B \leftarrow$Const_Inic1(G, α)
    sino
        $B \leftarrow$Const_Inic2(G, α)
    $S \leftarrow \{B, V \setminus B\}$
    $S' \leftarrow$Busqueda_Local(S)
    si S' *es mejor que* S^* **entonces**
        $S^* \leftarrow S'$
    $num\_iter \leftarrow num\_iter + 1$
    Incrementar α en 0.1 cada $\frac{MAXITER}{10}$ iteraciones
    si $num\_iter = \frac{MAXITER}{2}$ **entonces**
        $\alpha \leftarrow 0.5$
retornar S^*

su parte, los hipercubos analizados van desde Q_3 hasta Q_{10}.

La tabla 1 muestra los resultados obtenidos para los grafos pequeños. Estos resultados se corresponden con el mejor resultado obtenido después de 30 ejecuciones de cada algoritmo. Los tiempos presentados para cada uno se corresponden con el tiempo total empleado. En este conjunto de grafos todos los algoritmos comparados poseen un comportamiento similar. En negrita aparecen aquellos resultados obtenidos por un algoritmo que son estrictamente mejores que los resultados obtenidos por los otros algoritmos. Considerando este criterio, el mejor desempeño lo tiene GRASP, con una solución que supera en calidad a los otros algoritmos (instancia p52_20_27). Por otro lado, los tiempos usados por la metaheurística GRASP resultaron ser competitivos en relación a los otros algoritmos.

Revista Cubana de Ciencias Informáticas
Vol. 12, No. Especial UCIENCIA, Septiembre 2018
ISSN: 2227-1899 | RNPS: 2301
Pág. 28-41
http://rcci.uci.cu

Tabla 1: Resultados para grafos pequeños

	AMAGP		MAVBMP		GRASP	
File	**best**	**time**	**best**	**time**	**best**	**time**
p17_16_24	3	13.19	3	0.89	3	0.33
p18_16_21	2	11.22	2	0.93	2	0.36
p19_16_19	2	9.45	2	1.03	2	0.36
p20_16_18	2	7.09	2	1.06	2	0.33
p21_17_20	2	11.96	2	1.22	2	0.40
p23_17_23	2	11.50	2	1.27	2	0.40
p24_17_29	3	13.47	3	1.17	3	0.41
p32_18_20	2	10.63	2	1.26	2	0.44
p34_18_21	2	9.81	2	1.21	2	0.44
p36_18_20	2	9.98	2	1.27	2	0.44
p40_18_32	4	14.18	4	1.12	4	0.43
p44_19_25	3	14.00	3	1.25	3	0.49
p46_19_20	2	10.33	2	1.31	2	0.50
p49_19_22	2	11.60	3	1.28	2	0.47
p52_20_27	4	14.15	4	1.35	**2**	0.50
p53_20_22	2	14.15	2	1.4	2	0.50
p54_20_28	3	14.04	3	1.31	3	0.50
p57_20_24	2	13.20	2	1.24	2	0.51
p94_24_31	3	95.86	3	1.78	4	0.75
p95_24_27	2	40.59	2	1.92	2	0.75
p97_24_26	2	15.36	2	1.82	2	0.72
p98_24_29	2	59.04	3	1.74	2	0.72
p100_24_34	3	170.04	3	1.8	3	0.75

La Tabla 2 muestra los resultados obtenidos para los grafos Harwell-Boeing. Nuevamente en negrita están resaltadas aquellas soluciones de un algoritmo que mejoran a las soluciones obtenidas por los otros algoritmos. En este sentido, la metaheurística GRASP obtuvo el mejor desempeño. De acuerdo con la tabla, GRASP obtuvo 13 soluciones de mayor calidad que las obtenidas por los otros algoritmos. Solamente para la instancia can___73.mtx.rnd el algoritmo MAVBMP obtuvo un resultado superior.

Para los hipercubos el valor óptimo para el problema VBM es conocido y dado el hipercubo Q_n la solución para este problema viene dado por la expresión $\binom{n}{\frac{n}{2}}$ (Brandes and Fleischer, 2009). De acuerdo con la Tabla 3, en todos los casos los algoritmos MAVBMP y GRASP obtuvieron los valores óptimos, pero GRASP usa menos tiempo para alcanzar esos valores. En el caso de AMAGP, el desempeño no fue tan bueno sobre estos grafos y solo consigue encontrar los valores óptimos hasta el hipercubo Q_6.

Revista Cubana de Ciencias Informáticas
Vol. 12, No. Especial UCIENCIA, Septiembre 2018
ISSN: 2227-1899 | RNPS: 2301
Pág. 28-41
http://rcci.uci.cu

Tabla 2: Resultados para los Grafos Harwell-Boeing

	AMAGP		MAVBMP		GRASP	
File	**best**	**time**	**best**	**time**	**best**	**time**
arc130.mtx.rnd	15	19287.00	8	4.37	8	1.86
ash85.mtx.rnd	7	16475.00	7	4.02	7	0.84
bcspwr01.mtx.rnd	3	1210.90	3	3.38	3	0.18
bcspwr02.mtx.rnd	2	3365.50	2	4.59	2	0.29
bcspwr03.mtx.rnd	6	17589.00	5	8.73	**4**	1.57
bcsstk22.mtx.rnd	4	17482.00	5	8.01	4	1.30
can__144.mtx.rnd	6	20013.00	6	5.05	6	2.13
can__161.mtx.rnd	22	20828.00	24	11.77	**16**	2.86
can__715.mtx.rnd	50	92169.00	36	239.13	36	59.40
can___24.mtx.rnd	4	2080.40	4	1.54	4	0.09
can___61.mtx.rnd	5	15605.00	5	4.80	5	0.40
can___62.mtx.rnd	3	14625.00	3	8.41	3	0.45
can___73.mtx.rnd	11	15844.00	**8**	10.45	10	0.59
curtis54.mtx.rnd	6	14724.00	5	4.98	**4**	0.33
dwt__162.mtx.rnd	8	20052.00	9	10.86	**7**	2.62
dwt__193.mtx.rnd	24	23542.00	23	11.99	23	4.77
dwt__234.mtx.rnd	5	17124.00	4	8.50	4	1.49
gent113.mtx.rnd	21	17572.00	19	3.96	**14**	1.30
gre__115.mtx.rnd	22	17506.00	20	8.44	**18**	1.48
gre__185.mtx.rnd	21	21805.00	20	14.72	20	4.01
impcol_b.mtx.rnd	18	15335.00	17	6.31	**15**	0.47
impcol_c.mtx.rnd	29	18756.00	23	5.91	**21**	2.21
lns__131.mtx.rnd	14	17904.00	11	5.12	11	1.99
nos1.mtx.rnd	3	19684.00	3	9.95	3	2.57
nos4.mtx.rnd	8	16941.00	7	6.10	7	1.15
steam3.mtx.rnd	4	16557.00	4	8.20	4	0.67
west0132.mtx.rnd	29	18594.00	22	4.64	**19**	2.18
west0156.mtx.rnd	35	19793.00	35	7.72	**26**	2.93
west0167.mtx.rnd	25	20847.00	32	9.14	**19**	3.41
will199.mtx.rnd	72	22711.00	65	11.44	**53**	4.81
will57.mtx.rnd	4	14983.00	4	5.89	**3**	0.39

La Tabla 4 muestra los resultados sobre grafos bipartitos. Para estos grafos, todos los resultados obtenidos por GRASP se corresponden con los valores obtenidos por las heurísticas AMAGP y MAVBMP en (Jain et˜al., 2016c). Los tiempos utilizados por esos algoritmos no son presentados en ese artículo.

En general, la metaheurística GRASP mostró un comportamiento superior a las otras heurísticas comparadas,

Revista Cubana de Ciencias Informáticas
Vol. 12, No. Especial UCIENCIA, Septiembre 2018
ISSN: 2227-1899 | RNPS: 2301
Pág. 28-41
http://rcci.uci.cu

Tabla 3: Resultados obtenidos para hipercubos

	AMAGP		MAVBMP		GRASP	
File	**best**	**time**	**best**	**time**	**best**	**time**
Q3	3	4.97	3	0.33	3	0.01
Q4	6	13634.00	6	0.69	6	0.04
Q5	10	14817.00	10	2.10	10	0.16
Q6	20	15613.00	20	6.55	20	0.42
Q7	40	18276.00	35	5.31	35	1.57
Q8	70	27813.00	70	28.37	70	6.18
Q9	160	57034.00	126	158.02	126	24.32
Q10	280	159430.00	252	494.23	252	97.86

Tabla 4: Resultados sobre los grafos bipartitos

Grafo	**best**	**time**	**Grafo**	**best**	**time**
$K_{4,15}$	4	0.06	$K_{10,20}$	10	0.12
$K_{K4,20}$	4	0.10	$K_{10,30}$	10	0.20
$K_{4,30}$	4	0.14	$K_{10,50}$	10	0.36
$K_{4,50}$	4	0.28	$K_{10,100}$	10	1.12
$K_{4,100}$	4	1.02	$K_{20,15}$	15	0.13
$K_{5,15}$	5	0.05	$K_{20,20}$	20	0.16
$K_{5,20}$	5	0.09	$K_{20,30}$	20	0.29
$K_{5,30}$	5	0.13	$K_{20,50}$	20	0.50
$K_{5,50}$	5	0.32	$K_{20,100}$	20	1.35
$K_{5,100}$	5	1.02	$K_{50,4}$	4	0.28
$K_{6,15}$	6	0.06	$K_{50,30}$	30	0.64
$K_{6,20}$	6	0.11	$K_{50,50}$	50	0.97
$K_{6,30}$	6	0.14	$K_{50,100}$	50	2.17
$K_{6,50}$	6	0.30	$K_{100,4}$	4	1.04
$K_{6,100}$	6	1.10	$K_{100,10}$	10	1.15
$K_{10,10}$	10	0.08	$K_{100,50}$	50	2.09
$K_{10,15}$	10	0.07	$K_{100,100}$	100	3.76

tanto en términos de tiempo como en calidad de las soluciones. En resumen, 14 nuevas soluciones fueron encontradas. Cada una de estas soluciones representa un nuevo límite superior para la solución óptima del problema sobre estas instancias.

Debido a que las heurísticas comparadas en este trabajo cuentan con componentes aleatorias, los últimos experimentos fueron encaminados a explorar la estabilidad de sus soluciones. En particular, fueron analizados los resultados mínimos, medios y máximos obtenidos por los métodos MAVBMP y GRASP después de 30

Revista Cubana de Ciencias Informáticas
Vol. 12, No. Especial UCIENCIA, Septiembre 2018
ISSN: 2227-1899 | RNPS: 2301
Pág. 28-41
http://rcci.uci.cu

ejecuciones. Estos experimentos fueron realizados sobre los grafos de las instancias Harwell-Boeing, pues es para estas instancias que son ofrecidos esos valores en (Jain et˜al., 2016c). En la Tabla 5, las columnas 2, 3 y 4 representan los valores mínimos, medios y máximos obtenidos por el algoritmo memético MAVBMP, mientras que las columnas 5, 6 y 7 tienen el mismo significado pero para el algoritmo GRASP. Al analizar los resultados de esta tabla puede observarse como el algoritmo GRASP presenta los valores mínimos y máximos mucho más próximos de los valores medios que el algoritmo MAVBMP, indicando un menor error absoluto. En general, GRASP obtuvo resultados medios de mejor calidad en 24 grafos, mientras que MAVBMP solo obtuvo un resultado medio de mejor calidad que el algoritmo GRASP. Estos resultados apuntan a una mayor calidad y estabilidad en las soluciones obtenidas por el método GRASP.

Conclusiones

En este trabajo se desarrolló una metaheurística GRASP para el problema Vertex Bisection Minimization. De acuerdo con la estructura básica del método GRASP, fueron diseñadas e implementadas las fases constructivas y de búsqueda local. Para comprobar la efectividad del algoritmo propuesto, fueron desarrollados experimentos sobre las instancias más usadas en la literatura para este problema. Los resultados obtenidos muestran que el algoritmo propuesto obtiene soluciones de buena calidad, superando a las mejores heurísticas encontradas para este problema en la literatura revisada. Por otro lado, el hecho de proporcionar rápidamente soluciones de buena calidad, hacen de este algoritmo un método apropiado para ser integrado en algoritmos exactos para la resolución del problema tratado.

Agradecimientos

Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq)

Referencias

Aguilar, J. (2016). A general ant colony model to solve combinatorial optimization problems. *Revista Colombiana de Computación-RCC*, 2(1).

Brandes, U. and Fleischer, D. (2009). Vertex bisection is hard, too. *Journal of Graph Algorithms and Applications*, 13(2):119–131.

Bui, T. N. and Moon, B. R. (1996). Genetic algorithm and graph partitioning. *IEEE Transactions on computers*, 45(7):841–855.

Revista Cubana de Ciencias Informáticas
Vol. 12, No. Especial UCIENCIA, Septiembre 2018
ISSN: 2227-1899 | RNPS: 2301
Pág. 28-41
http://rcci.uci.cu

Tabla 5: Resultados mínimos, médios y máximos para los Grafos Harwell-Boeing

	MAVBMP			GRASP		
File	**min**	**ave**	**max**	**min**	**ave**	**max**
arc130.mtx.rnd	8	8.23	9	8	**8.00**	8
ash85.mtx.rnd	7	8.03	11	7	**7.00**	7
bcspwr01.mtx.rnd	3	3.00	3	3	3.00	3
bcspwr02.mtx.rnd	2	2.80	4	2	**2.07**	3
bcspwr03.mtx.rnd	5	6.33	11	4	**4.07**	5
bcsstk22.mtx.rnd	5	8.87	9	4	**4.00**	4
can__144.mtx.rnd	6	6.00	6	6	6.00	6
can__161.mtx.rnd	24	24.00	24	16	**16.00**	16
can__715.mtx.rnd	36	44.70	72	36	**40.97**	51
can___24.mtx.rnd	4	4.00	4	4	4.00	4
can___61.mtx.rnd	5	5.00	5	5	5.00	5
can___62.mtx.rnd	3	3.23	4	3	**3.10**	4
can___73.mtx.rnd	8	**9.00**	10	10	10.00	10
curtis54.mtx.rnd	5	5.00	5	4	**4.00**	4
dwt__162.mtx.rnd	9	9.00	9	7	**7.80**	8
dwt__193.mtx.rnd	23	29.77	49	23	**23.33**	29
dwt__234.mtx.rnd	4	4.87	8	4	**4.00**	4
gent113.mtx.rnd	15	20.47	28	14	**14.63**	16
gre__115.mtx.rnd	19	22.60	27	18	**18.03**	19
gre__185.mtx.rnd	20	21.13	26	20	**20.00**	20
impcol_b.mtx.rnd	17	17.7	18	15	**15.03**	16
impcol_c.mtx.rnd	23	26.13	31	21	**21.57**	23
lns__131.mtx.rnd	11	12.9	16	11	**12.63**	13
nos1.mtx.rnd	3	3.00	3	3	3.00	3
nos4.mtx.rnd	7	7.63	8	7	**7.00**	7
steam3.mtx.rnd	4	4.00	4	4	4.00	4
west0132.mtx.rnd	22	25.50	28	19	**19.33**	20
west0156.mtx.rnd	35	40.73	45	26	**26.97**	28
west0167.mtx.rnd	32	42.10	50	19	**19.80**	21
will199.mtx.rnd	65	68.37	72	53	**54.93**	57
will57.mtx.rnd	3	4.73	6	3	**3.00**	3

Díaz, J., Petit, J., and Serna, M. (2002). A survey of graph layout problems. *ACM Computing Surveys (CSUR)*, 34(3):313–356.

Díaz, J. A., Luna, D. E., Camacho-Vallejo, J.-F., and Casas-Ramírez, M.-S. (2017). Grasp and hybrid grasp-tabu heuristics to solve a maximal covering location problem with customer preference ordering. *Expert Systems with Applications*, 82:67–76.

Galinier, P., Boujbel, Z., and Fernandes, M. C. (2011). An efficient memetic algorithm for the graph partitioning problem. *Annals of Operations Research*, 191(1):1–22.

Jain, P., Saran, G., and Srivastava, K. (2016a). Branch and bound algorithm for vertex bisection minimization problem. In *Advanced Computing and Communication Technologies*, pages 17–23. Springer.

Jain, P., Saran, G., and Srivastava, K. (2016b). A new integer linear programming and quadratically constrained quadratic programming formulation for vertex bisection minimization problem. *Journal of Automation Mobile Robotics and Intelligent Systems*, 10.

Jain, P., Saran, G., and Srivastava, K. (2016c). On minimizing vertex bisection using a memetic algorithm. *Information Sciences*, 369:765–787.

Johnson, D. S., Aragon, C. R., McGeoch, L. A., and Schevon, C. (1989). Optimization by simulated annealing: an experimental evaluation; part i, graph partitioning. *Operations research*, 37(6):865–892.

Mestria, M., Ochi, L. S., and de Lima Martins, S. (2013). Grasp with path relinking for the symmetric euclidean clustered traveling salesman problem. *Computers & Operations Research*, 40(12):3218–3229.

Resende, M. G. and Ribeiro, C. C. (2016). *Optimization by GRASP*. Springer.