UN ALGORITMO DEL MÉTODO DE INTEGRACIÓN DE VARIABLES PARA LA SOLUCIÓN DEL PROBLEMA MÁXIMO CLIQUE PONDERADO

Gustavo Toranzo Lorca*[1] y José Arzola Ruiz**
*Departamento de Visualización y Realidad Virtual. Universidad de las Ciencias Informáticas (UCI). Carretera a San Antonio Km 2½, La Habana, Cuba.
**Centro de Estudios de Matemática para las Ciencias Técnicas. Instituto Superior Politécnico "José Antonio Echeverría" (CUJAE). Calle 114, No. 11901 entre 119 y 127, Marianao, C. Habana, Cuba.

ABSTRACT
In an undirected graph with positive weights on the vertices, the maximum weight clique problem is to find a complete subgraph with maximum sum of vertices weights. This problem is NP-complete. In the present work is proposed a heuristic algorithm based in the Integration of Variables Method for the maximum weight clique problem. A new coding system to the representation of the problem by the Integration of Variables Method is showed. The operators to evolve the population's code are defined. The algorithm is tested with DIMACS graphs and for generated random graphs. The results are compared with results of Lemke's method. Tests have shown that different stop criteria can be used to apply the algorithm to the needs of a particular application.

KEY WORDS: maximum weight clique, Integration of Variables Method, Metaheuristics.

MSC: 90C27

RESUMEN
En un grafo no dirigido y con pesos positivos asociados a los vértices, el problema máximo clique ponderado consiste en encontrar un subgrafo completo donde la suma de los pesos de los vértices sea máxima. El problema es NP-completo. En este trabajo es presentado un algoritmo heurístico basado en el Método de Integración de Variables para el problema máximo clique ponderado. Se presenta un nuevo sistema de codificación para la representación del problema mediante el Método de Integración de Variables y la definición de los operadores que permiten alterar la composición de los códigos de las soluciones en las poblaciones sucesivas. Son mostrados los resultados obtenidos en pruebas realizadas utilizando grafos DIMACS y grafos generados aleatoriamente. Los resultados obtenidos son comparados con resultados del método de Lemke. Las pruebas realizadas demuestran que distintos criterios de parada pueden utilizarse para adaptar el algoritmo a las necesidades de una aplicación en particular.

1. INTRODUCCIÓN

En un grafo G no dirigido con un peso positivo asociado a cada vértice, un clique es un subgrafo donde todos los vértices son adyacentes. El problema del máximo clique ponderado (*Maximum Weight Clique* MWC) consiste en encontrar en G un clique donde la suma de los pesos de los vértices sea máxima. El problema MWC es bien conocido por ser NP-completo [20]. El problema de máximo clique (*Maximum Clique* MC) es un caso particular del problema MWC cuando todos los vértices de G tienen pesos iguales y puede presentarse en un grafo no dirigido y no ponderado.
Los problemas descritos tienen gran variedad de aplicaciones. En particular el problema MWC tiene aplicaciones en la identificación de caras poligonales como proceso de la reconstrucción tridimensional [14] y en la segmentación de objetos en secuencias de video
[15]. También encontramos aplicaciones en el diagnóstico de fallas en sistemas con múltiples procesadores [7], teoría de códigos y biología molecular

[1] gtoranzo@uci.cu

[13]. Un análisis más extenso acerca de ambos problemas y sus aplicaciones es realizado en
[9].
Existen dos enfoques principales para resolver el problema MWC, la resolución mediante un algoritmo exacto o mediante un algoritmo heurístico
[9]. Han sido elaborados diferentes algoritmos de ambos tipos para el problema. Los algoritmos exactos tienen la desventaja que exploran en el peor de los casos todas las soluciones posibles, por lo cual solo pueden aplicarse para grafos pequeños. Liu y Lee presentan en [14] un algoritmo enumerativo con poda, y es utilizado en la identificación de caras poligonales de un objeto alámbrico proyectado en el plano. Otros algoritmos exactos pueden encontrarse en
[13] y
[21].
En 1994 L. Babel presenta un método de la técnica *branch and bound* con heurística de coloración ponderada para grafos arbitrarios
[6]. También han sido elaborados algoritmos heurísticos con técnicas de Redes Neuronales, Algoritmos Genéticos y Búsqueda Tabú
[9]. Marchiori presenta en
[16] un algoritmo genético para el problema MWC. En [8] se propone la aproximación del problema MWC usando Replicator Dynamics en Redes Neuronales.
Un conjunto de grafos de referencias de acceso público es mantenido por *Center for Discrete Mathematics and Theoretical Computer Science* (DIMACS). Los grafos públicos de DIMACS y grafos construidos aleatoriamente pueden ser usados como indicadores de la calidad de los nuevos algoritmos. Aunque muchos algoritmos han sido publicados desde 1970, la comparación entre estos se hace difícil pues no existen publicaciones amplias sobre comparaciones [19].
El presente trabajo describe un algoritmo no exacto basado en el Método de Integración de Variable [3] para el problema MWC. En los siguientes epígrafes se aborda el Método de Integración de Variables, el algoritmo propuesto basado en el método descrito, y por último son mostrados los resultados obtenidos mediante experimentos en grafos DIMACS y aleatorios que demuestran la competitividad del algoritmo.

2. MÉTODO DE INTEGRACIÓN DE VARIABLES

El concepto del método de Integración de Variables se vincula a la evolución de la cantidad requerida de códigos con ayuda de cualquier conjunto de operadores para actualizar los miembros de una población [4]. La Figura 1 representa el esquema general del método. Los rasgos generales son los siguientes:

- Las posibles variantes de solución son codificadas en uno o más códigos variables.
- Se genera, según un procedimiento característico para cada realización particular del método, un juego de *n* soluciones próximas al óptimo. En particular, en calidad de procedimientos pueden usarse diferentes métodos iterativos de la Programación No lineal y Discreta aplicados a los códigos de solución, con entornos de búsqueda seleccionados al azar, determinísticamente o de forma combinada. Cada procedimiento particular de generación de poblaciones se corresponde con una variante concreta de aplicación del método. Un caso particular de procedimiento está constituido por una sucesión dada de operadores genéticos aplicada a una población inicial, lo que lleva a los Algoritmos Genéticos.
- Aunque la selección del criterio de parada obedece, en el caso general, a las particularidades de cada aplicación concreta, en la mayoría de los casos debe ser eficaz la condición de lograr una cantidad dada de iteraciones sin modificaciones de la población (o lo que es equivalente, sin variar el valor de la función de calidad del peor elemento de la población).

De tal forma, la aplicación de cualquier heurística derivada del método de Integración de Variables requiere de la definición de los siguientes problemas:

- Un sistema de codificación para la representación de las posibles soluciones del problema estudiado.
- Un método para la creación de la población inicial.
- Una función de calidad (*fitness*) que permita ordenar los códigos de acuerdo con los valores de la función objetivo.
- Operadores que permiten alterar la composición de los códigos de las soluciones en las poblaciones sucesivas.
- Valores de los parámetros requeridos por el algoritmo utilizado (tamaño de la población, probabilidades asociadas con la aplicación de ciertos operadores, etc.).

Aunque Arzola propuso por primera vez explícitamente el concepto del método de Integración de Variables y sus particularidades distintivas, este concepto ha sido aplicado de hecho en numerosos trabajos aparecidos en años recientes en la literatura especializada. Así, en el artículo de Michalewicz [18] se explican algunos algoritmos evolutivos aplicados a los problemas de optimización no lineales con restricciones, los que aportan nuevos operadores al método de Integración de Variables. Por otro lado, los Algoritmos Genéticos estructurados y otros Algoritmos Genéticos complejos constituyen casos particulares del uso de conjuntos de códigos para describir opciones de la solución

- [10]. El Método de Selección de Propuestas [1] [2] constituye otro ejemplo del uso de códigos variables múltiples para describir las soluciones.

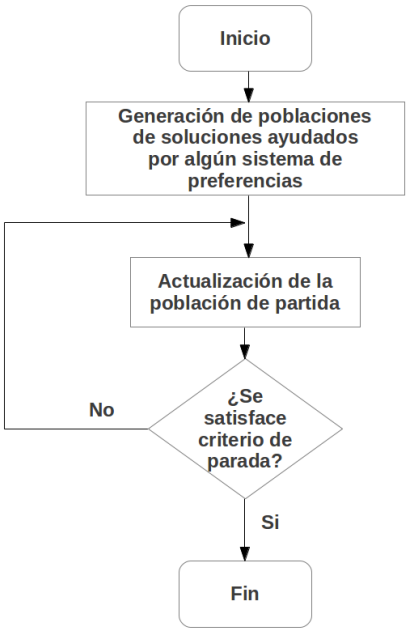

Figura 1. Esquema general del Método de Integración de Variables

3. ALGORITMO PARA EL PROBLEMA MÁXIMO CLIQUE PONDERADO

El algoritmo de búsqueda propuesto para el problema MWC responde al esquema general del Método de Integración de Variables. A continuación se describen las características principales.

Codificación

Se define al conjunto de n vértices de G por $V = \{1,2,...,n\}$. Cada elemento de la población responde a tres vectores denominados $VLibres$, $VBloqueados$ y $VSol$. Cada vector almacena los índices de los nodos de G que pertenecen a él. El conjunto de vértices que pertenecen a $VSol$ cumplen la propiedad de ser un clique en G, son un subgrafo completo de G. $VLibres$ almacena el conjunto de vértices disponibles para aumentar la cantidad de vértices en $VSol$, cada uno de los vértices de $VLibres$ es adyacente a todos los vértices de $VSol$. El conjunto de vértices en $VBloqueados$ cumplen la propiedad de no ser adyacentes con al menos uno de los vértices en $VSol$. Ninguno de los vértices de $VBloqueados$ puede ser agregado a $VSol$ porque entonces $VSol$ dejaría de ser un clique.

Método para la creación de la población inicial

La cantidad de elementos ce tiene un impacto fundamental en la calidad de la solución a encontrar y en el tiempo de ejecución. A mayor tamaño de la población se obtiene mejor exploración de soluciones diferentes. Se denota como $VLibres_i, VBloqueados_i, VSol_i$ a los vectores correspondientes al elemento i de la población. La configuración inicial para cada elemento se define:

$ce > 0, ce \in Z$

$i \in \{1,2,...,ce\}$

$VLibres_i = \{1,2,...,n\}$

$VBloqueados_i = \{\}$

$VSol_i = \{\}$

Función de calidad

Siendo ω_j el valor de ponderación del vértice j y W_i la calidad del elemento i ; se define como función de calidad:

$$W_i = \sum_{j \in VSol_i} \omega_j$$

La anterior definición representa para un elemento i la suma de pesos de los vértices que se encuentran en el vector de soluciones $VSol$. En cada iteración se compara W_i con el valor de MWC global W , y si $W < W_i$ entonces el valor W es actualizado y una copia de $VSol_i$ es hecha para $VBest$. Al finalizar la ejecución $VBest$ almacena los índices de los vértices del mejor clique encontrado.

Operadores para generar poblaciones sucesivas

Se definen dos operadores que permiten la evolución de una población, el operador de adición y el operador de eliminación. Cada operador es aplicado de manera independiente a un elemento de la población. Estos operadores tienen funciones opuestas y permiten explorar el conjunto de soluciones.

El operador de adición tiene como vector de partida a $VLibres$, su función es seleccionar aleatoriamente un vértice de $VLibres$, eliminarlo y adicionarlo al vector $VSol$. Los vértices de $VLibres$ cumplen la propiedad de ser adyacentes a todos los vértices de $VSol$, por lo tanto $VSol$ con este nuevo vértice continúa siendo un clique. El resto de los vértices de $VLibres$ tienen que ser explorados para eliminar aquellos que no son adyacentes al vértice adicionado a $VSol$ y adicionarlos al vector $VBloqueados$, de esta manera se conserva la propiedad de los vértices de $VLibres$. El desarrollo de este operador está inspirado en el algoritmo heurístico E1 descrito por Johnson para MC en [12].

El operador de eliminación tiene como vector de partida a $VSol$, y su función es seleccionar aleatoriamente un vértice de $VSol$, eliminarlo y adicionarlo a $VLibres$. Los vértices de $VBloqueados$ tienen que ser explorados para seleccionar aquellos que no son adyacentes al vértice eliminado de $VSol$ y que además sean adyacentes a todos los vértices de $VSol$ para adicionarlos a $VLibres$. Este operador permite realizar un retroceso en el proceso de búsqueda semejante a la técnica *backtracking*.

La probabilidad de aplicación del operador de adición debe ser alta. Las probabilidades de los operadores de adición y eliminación son complementarias. Si la probabilidad indica la aplicación de un operador, pero este no puede ser aplicado porque no existen vértices en su vector de partida entonces el operador opuesto debe ser aplicado. La complejidad computacional del operador de adición es $O(n)$ y del operador de eliminación es $O(n^2)$. Una característica importante en la eficiencia del algoritmo es la aplicación del operador de adición con una probabilidad alta pues la complejidad computacional de este es menor que la complejidad de su opuesto.

Criterio de parada

Existen diversos criterios de parada que se pueden establecer, escoger uno de ellos depende de la aplicación concreta del problema MWC. Aquí son presentados tres criterios fundamentales que responden a necesidades diferentes.

El primer criterio de parada consiste en realizar un número de iteraciones igual al número de vértices del grafo. Este criterio garantiza que se conoce el número de iteraciones a ejecutar y que el tiempo de ejecución puede estimarse con bastante exactitud. El algoritmo propuesto utilizando este criterio de parada será denominado en lo adelante LBS1.

El segundo criterio de parada consiste en terminar la ejecución cuando se hayan realizado una cantidad de iteraciones mayor o igual que $\log_2 n$ sin haberse mejorado el valor W. Con este criterio el algoritmo garantiza no terminar mientras se mejore la solución global en un número pequeño de iteraciones. El algoritmo propuesto utilizando este criterio de parada será denominado en lo adelante LBS2.
El criterio número tres sólo se diferencia del primer criterio en que el número de iteraciones para terminar la ejecución tiene que ser mayor o igual que n. Este criterio garantiza resultados mejores que el primer criterio pero en un tiempo de ejecución más alto. El algoritmo propuesto utilizando este criterio de parada será denominado en lo adelante LBS3.

4. RESULTADOS COMPUTACIONALES

En esta sección se realizan experimentos para evaluar correctamente el nuevo algoritmo. Será mostrado el rendimiento en grafos de referencias DIMACS y en grafos aleatorios. Para cada caso de prueba el algoritmo fue ejecutado cien veces. El tamaño de la población utilizado en las ejecuciones es igual al número de vértices del grafo. La probabilidad de aplicación del operador de adición fue de 0.95. La media y la varianza del máximo clique ponderado encontrado y la media de tiempo de ejecución fueron calculados. El procedimiento fue implementado en C++. Las pruebas se realizaron en una Laptop Acer con 2 GB de memoria RAM y un procesador Intel P6200 a 2.13 GHz. Un sistema operativo Linux fue utilizado. Los tiempos de usuario para los grafos de referencias DIMACS destinados a la comparación del rendimiento de PC denominados r100.5, r200.5, r300.5, r400.5 y r500.5 fueron 0.00, 0.04, 0.27, 1.62, y 6.15 segundos respectivamente.

4.1. Grafos DIMACS

Los grafos DIMACS no son ponderados, sin embargo aquí es utilizado un valor constante como peso para cada uno de los nodos. Los resultados son mostrados en las tablas 1 y 2. El nuevo algoritmo es comparado con el método de Lemke utilizando el criterio de resolución por degeneración lexicográfica (LDR) y utilizando heurística basada en pivote (PBH) [17]. Los datos de LDR y PBH son tomados de las tablas 1 y 2 de [17]. En [17] sólo es descrita la PC donde fueron realizadas las pruebas, por lo cual solo es posible realizar una comparación aproximada en cuanto a los tiempos de ejecución.

Tabla 1. Resultados en grafos DIMACS (parte I)

		LDR		PBH		LBS1			LBS2			LBS3		
Grafos	ω	ω	Time	ω	Time	μ	σ^2	μ_T	μ	σ^2	μ_T	μ	σ^2	μ_T
c-fat200-1	12	12	0.07	12	5.0	12.00	0.00	0.04	12.00	0.00	0.00	12.00	0.00	0.04
c-fat200-2	24	24	0.12	24	9.0	24.00	0.00	0.04	24.00	0.00	0.00	24.00	0.00	0.04
c-fat200-5	58	58	0.28	58	22.5	58.00	0.00	0.03	58.00	0.00	0.00	58.00	0.00	0.04
c-fat500-1	14	14	0.48	14	100.3	14.00	0.00	0.78	14.00	0.00	0.03	14.00	0.00	0.80
c-fat500-2	26	26	0.79	26	185.2	26.00	0.00	0.75	26.00	0.00	0.03	26.00	0.00	0.78
c-fat500-5	64	64	1.83	64	464.5	64.00	0.00	0.66	64.00	0.00	0.04	64.00	0.00	0.76
c-fat500-10	126	126	3.59	126	1024.2	126.00	0.00	0.52	126.00	0.00	0.06	126.00	0.00	0.69
hamming6-2	32	32	0.01	32	0.4	31.51	1.38	0.00	31.30	0.98	0.00	31.56	0.98	0.00
hamming6-4	4	4	0.00	4	0.1	4.00	0.00	0.00	4.00	0.00	0.00	4.00	0.00	0.00
hamming8-2	128	128	0.98	128	252.6	107.19	22.83	0.68	100.6	54.04	0.02	115.12	7.86	0.22
hamming8-4	16	16	0.14	16	22.8	16.00	0.00	0.16	15.47	1.14	0.01	16.00	0.00	0.18
hamming10-2	512	512	61.01	512	-	362.12	89.26	4.50	303.16	251.77	1.03	454.52	25.56	32.75
hamming10-4	$\geq$40	32	4.1	32	-	40.00	0.00	10.40	31.92	0.61	0.35	40.00	0.00	15.12
johnson8-2-4	4	4	0	4	0.0	4.00	0.00	0.00	4.00	0.00	0.00	4.00	0.00	0.00
johnson8-4-4	14	14	0.01	14	0.3	14.00	0.00	0.00	13.56	0.92	0.00	14.00	0.00	0.00
johnson16-2-4	8	8	0.01	8	1.1	8.00	0.00	0.01	8.00	0.00	0.00	8.00	0.00	0.01
johnson32-2-4	$\geq$16	16	0.54	16	184.8	16.00	0.00	0.93	16.00	0.00	0.03	16.00	0.00	0.95
p_hat300-1	8	6	0.1	8	14.0	8.00	0.00	0.28	7.41	0.24	0.01	8.00	0.00	0.30
p_hat300-2	25	16	0.2	25	34.9	25.00	0.00	0.26	21.09	0.82	0.02	25.00	0.00	0.39
p_hat300-3	36	21	0.25	35	61.0	34.35	0.56	0.20	29.59	1.22	0.02	35.66	0.42	0.51
p_hat500-1	9	6	0.27	9	83.5	9.00	0.00	1.45	8.45	0.24	0.04	9.00	0.00	1.50
p_hat500-2	36	26	0.82	36	282.5	35.98	0.01	1.27	28.49	1.66	0.07	36.00	0.00	2.23
p_hat500-3	$\geq$50	30	0.94	48	485.7	48.54	0.28	0.99	39.14	1.70	0.06	49.62	0.23	2.42
p_hat700-1	11	5	0.47	10	249.4	11.00	0.00	4.37	9.05	0.14	0.11	11.00	0.00	5.05
p_hat700-2	44	20	1.26	44	1022.3	44.00	0.00	3.79	35.11	2.35	0.18	44.00	0.00	6.11
p_hat700-3	$\geq$62	29	1.76	62	1804.0	60.6	0.36	2.84	47.41	4.20	0.17	61.91	0.08	6.82
p_hat1000-1	$\geq$10	7	1.17	10	798.0	10.00	0.00	13.49	9.32	0.21	0.26	10.00	0.00	13.79
p_hat1000-2	$\geq$46	18	2.37	46	-	46.00	0.00	12.14	36.08	2.67	0.40	46.00	0.00	19.04
p_hat1000-3	$\geq$66	31	3.82	64	-	65.01	0.72	9.10	49.55	2.48	0.37	67.59	0.32	27.20
p_hat1500-1	12	9	3.12	12	-	11.37	0.23	50.57	10.15	0.12	0.73	11.36	0.23	59.41
p_hat1500-2	$\geq$65	28	7.69	64	-	64.99	0.00	43.80	47.97	3.76	1.20	65.00	0.00	76.68
p_hat1500-3	$\geq$94	43	11.43	91	-	92.05	0.36	35.13	64.06	6.27	1.19	93.79	0.18	100.00

La primera columna ω muestra el valor de máximo clique ponderado y las otras columnas ω son los valores de máximo clique ponderado para los algoritmos LDR y PBH respectivamente. μ es la media de máximos cliques ponderados encontrados en las cien ejecuciones y σ^2 la varianza. μ_T es la media de tiempo de ejecución en segundos.

LDR y PBH no son algoritmos exactos, utilizan la formulación del problema máximo clique ponderado como problema de complementariedad lineal. PBH mejora los resultados de LDR ingresando una regla de pivote denominada por los autores *look-ahead.*

Para el grafo p_hat1000-3 el algoritmo LBS3 encontró cliques con tamaño 68. La siguiente es una solución de tamaño 68 (los vértices están enumerados comenzando por 1): {8, 38, 65, 92, 104, 162, 171, 186, 192, 194, 202, 204, 226, 227, 242, 243, 262, 271, 283, 330, 334, 335, 370, 378, 401, 406, 411, 461, 492, 495, 497, 527, 537, 538, 549, 562, 598, 599, 604, 621, 623, 639, 650, 651, 662, 671, 705, 719, 778, 782, 783, 784, 798, 780, 819, 824, 836, 864, 868, 876, 897, 900, 914, 922, 927, 967, 992, 993}.

Diferentes familias de grafos son utilizadas en las pruebas basadas en los grafos DIMACS. La Tabla 1 muestra las pruebas realizadas con grafos CFats, Hamming, Johnson y PHat. Los grafos CFats surgen del problema de diagnóstico de fallas en sistemas con múltiples procesadores [7]. Los grafos de Hamming y Johnson están basados en los esquemas de Hamming y Johnson respectivamente. PHat son grafos aleatorios con alta varianza en la distribución del grado de los nodos, estos grafos tienen cliques mayores que los grafos aleatorios
[16].

En los resultados correspondientes a los grafos CFats de la Tabla 1 las tres versiones del algoritmo propuesto obtienen el óptimo, destacándose la versión LBS2 por tener menor media de tiempo de ejecución. Para el grafo hamming10-2 el algoritmo propuesto obtiene resultados inferiores a LDR y PBH en cuanto a clique se refiere pero los tiempos de ejecución se mantienen por debajo.

Tabla 2. Resultados en grafos DIMACS (parte II)

		LDR		PBH		LBS1			LBS2			LBS3		
Grafos	ω	ω	Time	ω	Time	μ	σ^2	μ_T	μ	σ^2	μ_T	μ	σ^2	μ_T
MANN_a9	16	16	0.00	16	0.1	16.00	0.00	0.00	16.00	0.00	0.00	16.00	0.00	0.00
MANN_a27	126	125	2.18	125	699.7	123.22	0.23	0.18	12.78	0.43	0.05	123.42	0.34	0.28
MANN_a45	345	340	43.72	342	-	336.88	0.38	4.15	335.88	1.68	1.16	336.97	0.44	5.88
keller4	11	7	0.03	11	3.6	11.00	0.00	0.04	10.95	0.04	0.00	11.00	0.00	0.05
keller5	27	15	1.27	26	1093.5	26.64	0.23	4.47	22.88	0.60	0.15	26.80	0.16	7.43
keller6	$\geq$59	31	45.54	-	-	53.97	0.60	782.19	45.11	1.11	8.09	54.46	0.72	1484.86
brock200_1	21	13	0.07	20	9.7	19.82	0.20	0.07	17.94	0.37	0.00	19.99	0.12	0.12
brock200_2	12	7	0.04	11	5.1	10.84	0.43	0.08	9.91	0.68	0.00	10.87	0.33	0.12
brock200_3	15	10	0.6	14	6.4	13.75	0.38	0.08	12.45	0.42	0.00	13.83	0.40	0.12
brock200_4	17	11	0.06	16	7.3	15.62	0.33	0.08	14.02	0.29	0.00	15.87	0.35	0.13
brock400_1	27	17	0.37	24	111.6	23.66	0.38	0.63	20.66	0.38	0.03	24.00	0.38	1.17
brock400_2	29	17	0.37	24	113.3	23.70	0.29	0.62	20.87	0.39	0.03	24.01	0.46	1.11
brock400_3	31	17	0.37	24	111.2	23.76	1.82	0.63	20.67	0.36	0.03	24.12	1.82	1.11
brock400_4	33	16	0.35	24	112.7	24.66	6.62	0.60	20.79	0.44	0.03	25.02	6.53	1.11
brock800_1	23	13	0.18	21	858.6	20.10	0.11	6.73	17.49	0.26	0.16	20.19	0.15	10.16
brock800_2	24	13	1.19	20	866.4	20.08	0.07	6.71	17.61	0.37	0.16	20.16	0.13	9.79
brock800_3	25	15	1.34	20	864.5	20.13	0.13	6.73	17.57	0.28	0.16	20.23	0.27	9.88
brock800_4	26	16	1.40	20	862.4	20.07	0.10	6.72	17.47	0.32	0.16	20.19	0.21	10.22
san200_0.7_1	30	16	0.09	30	9.9	27.01	11.40	0.07	20.73	17.99	0.00	27.68	10.67	0.12
san200_0.7_2	18	12	0.08	17	8.2	15.72	1.30	0.06	14.22	0.21	0.00	15.94	1.81	0.10
san200_0.9_1	70	38	0.19	70	28.8	59.12	13.86	0.04	54.30	22.63	0.01	65.36	5.15	0.13
san200_0.9_2	60	30	0.16	60	22.8	50.29	7.92	0.04	41.71	10.96	0.00	57.47	2.84	0.14
san200_0.9_3	44	25	0.13	44	19.0	35.84	1.63	0.04	32.32	0.47	0.00	42.01	5.04	0.17
san400_0.5_1	13	7	0.20	13	52.3	10.14	3.46	0.53	8.31	0.25	0.02	10.37	4.15	0.67
san400_0.7_1	40	20	0.43	40	142.0	28.36	52.41	0.58	22.48	2.02	0.03	30.34	55.86	0.85
san400_0.7_2	30	15	0.35	30	110.7	20.30	17.15	0.60	17.89	0.39	0.03	20.87	21.73	0.82
san400_0.7_3	22	14	0.31	17	93.8	17.93	2.70	0.62	15.62	0.25	0.03	18.92	3.91	1.16
san400_0.9_1	100	45	0.88	100	397.8	84.46	35.04	0.40	58.59	59.12	0.04	95.37	8.15	1.24
sanr200_0.7	18	12	0.07	18	8.2	17.44	0.24	0.07	15.62	0.41	0.00	17.62	0.23	0.12
sanr200_0.9	42	32	0.16	41	21.4	38.48	0.78	0.04	35.15	1.06	0.00	40.18	0.90	0.13
sanr400_0.5	13	10	0.25	13	59.5	12.47	0.24	0.76	11.20	0.18	0.03	12.59	0.24	1.04
sanr400_0.7	$\geq$21	16	0.36	20	101.9	20.38	0.23	0.65	18.08	0.33	0.03	20.55	0.24	1.10
san1000	15	8	1.34	15	1185.0	10.01	0.72	9.33	9.81	0.45	0.23	9.92	0.31	9.69

1.1 Las pruebas de la Tabla 2 utilizan grafos MANN, Keller, Brock, San y Sanr. Los grafos MANN son una reducción al problema máximo clique del problema *Minimum Set Cover* [17]. Los grafos de Keller tienen su base en la conjetura de Keller [11]. San y Sanr son grafos aleatorios con máximo clique conocido y para los grafos Brock el máximo clique es mucho mayor que el esperado [16].

En sentido general los tiempos de ejecución y el valor de máximo clique de LBS1, LBS2 y LBS3 respecto a LDR y PBH son mejores. Se aprecia el caso del grafo keller6 en el cual para PBH no se reportan datos debido al prolongado tiempo de ejecución y el máximo clique encontrado por LDR (31) en 45.54 segundos es inferior al encontrado por LBS1, LBS2 y LBS3; para este caso es significativo que en promedio LBS2 demoró 8.09 segundos [17].

4.2.Grafos aleatorios

El algoritmo presentado ha sido extensamente probado en grafos aleatorios. Un grafo aleatorio es un grafo de n vértices donde la probabilidad de existencia de una arista entre cada par de nodos es p. p es la densidad de aristas en el grafo. Si el número de vértices es fijado entonces la densidad de aristas puede ser interpretada como una medida de la dificultad del problema MWC, según aumenta p la dificultad aumenta [5].

El número de vértices y la densidad de aristas han sido variados en las pruebas para medir el comportamiento del algoritmo. Los pesos de los nodos son valores enteros generados aleatoriamente en el intervalo [1,10]. La Tabla 3 muestra los resultados obtenidos. El valor de máximo clique ponderado para cada grafo generado fue obtenido mediante un algoritmo exacto, por lo tanto el valor de la columna ω es óptimo.

Tabla 3. Resultados en grafos aleatorios

			LBS1			LBS2			LBS3		
Vértices	p	ω	μ	σ^2	μ_T	μ	σ^2	μ_T	μ	σ^2	μ_T
50	0.1	22	22.00	0.00	0.00	21.68	0.34	0.00	22.00	0.00	0.00
50	0.2	29	28.73	0.74	0.00	27.62	2.52	0.00	28.79	0.59	0.00
50	0.3	39	38.83	0.98	0.00	36.95	11.31	0.00	38.78	1.43	0.00
50	0.4	44	44.00	0.00	0.00	43.05	3.15	0.00	44.00	0.00	0.00
50	0.5	51	50.28	0.60	0.00	48.62	3.70	0.00	50.44	0.51	0.00
50	0.6	53	52.80	0.34	0.00	51.75	3.15	0.00	52.98	0.02	0.00
50	0.7	79	76.60	3.38	0.00	73.65	9.49	0.00	77.53	1.53	0.00
50	0.8	113	105.27	18.50	0.00	101.17	30.42	0.00	109.79	11.57	0.00
50	0.9	157	149.84	13.07	0.00	147.31	19.85	0.00	152.91	9.40	0.00
100	0.1	28	28.00	0.00	0.01	27.61	0.56	0.00	28.00	0.00	0.01
100	0.2	36	35.24	2.46	0.01	33.92	5.11	0.00	34.88	3.23	0.01
100	0.3	48	47.82	0.33	0.01	45.40	7.60	0.00	47.82	0.33	0.01
100	0.4	55	54.10	2.91	0.01	49.48	5.41	0.00	54.79	0.85	0.02
100	0.5	71	68.11	9.56	0.01	60.47	16.97	0.00	69.92	4.59	0.02
100	0.6	80	78.69	2.91	0.01	72.37	10.11	0.00	79.36	1.69	0.02
100	0.7	91	86.23	6.72	0.01	79.75	19.91	0.00	87.86	3.44	0.02
100	0.8	126	119.33	9.18	0.01	110.15	24.41	0.00	121.69	7.75	0.02
200	0.1	36	36.00	0.00	0.06	34.83	3.32	0.00	35.99	0.01	0.07
200	0.2	47	46.98	0.02	0.07	43.45	9.21	0.01	47.00	0.00	0.09
200	0.3	55	54.97	0.09	0.08	49.98	7.94	0.01	54.97	0.09	0.10
200	0.4	64	63.23	0.84	0.09	56.33	10.46	0.01	63.46	0.73	0.13
200	0.5	82	80.18	5.31	0.09	70.40	18.00	0.01	81.02	3.32	0.15
200	0.6	95	91.12	5.83	0.08	81.74	12.33	0.01	92.60	2.88	0.16
500	0.1	38	38.00	0.00	1.00	37.40	0.50	0.04	38.00	0.00	1.04
500	0.2	51	50.94	0.06	1.22	48.75	2.89	0.05	50.96	0.04	1.45
500	0.3	62	61.67	0.62	1.61	56.24	6.64	0.05	61.78	0.41	2.03
500	0.4	80	77.49	2.79	1.52	68.41	11.94	0.06	77.97	2.75	2.56
500	0.5	103	93.11	9.26	1.58	81.13	12.01	0.06	93.96	13.20	2.82
1000	0.1	47	47.00	0.00	9.00	44.04	5.46	0.25	47.00	0.00	9.89
1000	0.2	61	59.20	3.06	11.02	53.27	7.12	0.25	58.88	3.47	14.92
1000	0.3	78	73.11	11.56	13.04	62.43	11.71	0.27	73.50	8.37	20.59
1000	0.4	92	86.10	3.39	14.31	76.02	7.20	0.29	86.63	2.75	23.70

En la Tabla 3 se puede apreciar que solamente existen cuatro entradas donde LBS1 supera en la media de máximo clique ponderado a LBS3. En los datos presentados el algoritmo LBS3 tiene su valor de media más alejado del óptimo para el grafo aleatorio de 500 vértices y densidad de aristas 0.5, donde la diferencia con respecto al óptimo es de 9.04. La media de tiempo de ejecución de LBS2 nunca supera un

segundo y en todos los casos es menor o igual que la media de tiempo de LBS1 y LBS3. Para LBS2 la media de máximo clique ponderado no supera las medias de LBS1 y LBS3.

5. CONCLUSIONES

En este artículo ha sido presentado un nuevo algoritmo basado en el Método de Integración de Variables para encontrar el máximo clique ponderado. Fueron utilizados grafos generados aleatoriamente y grafos DIMACS en la pruebas, obteniéndose un buen comportamiento del algoritmo, lo que se demuestra en tablas que lo comparan con el método de Lemke. El algoritmo presentado combina dos operadores complementarios para explorar el conjunto de soluciones, un operador de adición y un operador de eliminación. Distintos criterios de parada permiten que el algoritmo pueda ser utilizado en aplicaciones diferentes. La codificación presentada posibilita la investigación de nuevos operadores, correspondientes a nuevos algoritmos del Método de Integración de Variables.

RECEIVED MARCH 2013
REVISED AUGUST 2013

REFERENCIAS

[1] ARZOLA, R. J. (1989): **Selección de Propuestas**. Editorial Científico Técnica, La Habana.
[2] ARZOLA, R. J. (2003): La tarea de Selección de Propuestas bajo Criterios Múltiples. Métodos de Solución. **Revista de Matemática**, 11.
[3] ARZOLA, R. J. (2008): **Análisis y Síntesis de Sistemas de Ingeniería**. Disponible en http://www.bibliomaster.com/.
[4] ARZOLA, R. J., SIMEÓN, R. E. y MACEO, A. (2003): El Método de Integración de Variables: una generalización de los Algoritmos Genéticos. **Proceeding of Intensive Workshop: Optimal Design of Materials and Structures,** París.
[5] BABEL, L. (1991): Finding Maximum Cliques in Arbitrary and in Special Graphs. **Computing**, 46, 321-341.
[6] BABEL, L. (1994): A Fast Algorithm for the Maximum Weight Clique Problem. **Computing**, 52, 31-38.
[7] BERMAN, P. y PELC, A. (1990): Distributed fault diagnosis for multiprocessors systems. **20th Annual Int. Symp. on Fault-Tolerant Computing**, 340-346.
[8] BOMZE, I. R., PELILLO, M. y STIX, V. (2000): Approximating the maximum weight clique using replicator dynamics. **Neural Networks, IEEE Transactions on**, 11, 1228-1241.
[9] BOMZE, I., BUDINICH, M., PARDALOS, P. M. y PELILLO, M. (1999): The Maximum Clique Problem. **Handbook of combinatorial optimization**, 4, 1-74.
[10] DÍAZ, A., GLOVER, F., GONZALES, J. L., LAGUNA, M., MOSCAZO, P. y GHAZIRI, H. M. (1996): **Optimización heurística y redes neuronales**. Editorial Paraninfo, Madrid.
[11] HASSELBERG, J., PARDALOS, P. M. y VAIRAKTARAKIS, G. (1993): Test Case Generators and Computational Results for the Maximum Clique Problem. Disponible en ftp://**dimacs.rutgers.edu**
[12] JOHNSON, D. S. (1974): Approximation algorithms for combinatorial problems. **Journal of computer and system sciences**, 9, 256-278.
[13] KUMLANDER, D. (2004): A new exact algorithm for the maximum-weight clique problem based on a heuristic vertex-coloring and a backtrack search. **5th International Conference on Modelling, Computation and Optimization in Information Systems and Management Sciences: MCO**, 4, 202-208.
[14] LIU, J. y LEE, Y. T. (2001): A graph-based method for face identification from a single 2d line drawing. **Pattern Analysis and Machine Intelligence, IEEE Transactions,** 23, 1106-1119.
[15] MA, T. y LATECKI, L. J. (2012): Maximum Weight Cliques with Mutex Constraints for Video Object Segmentation. **Computer Vision and Pattern Recognition (CVPR), IEEE Conference** , 670-677.
[16] MARCHIORI, E. (1998): A Simple Heuristic Based Genetic Algorithm for the Maximum Clique Problem. **Symposium on Applied Computing: Proceedings of the 1998 ACM symposium on Applied Computing**, 27, 366-373.
[17] MASSARO, A., PELILLO, M. y BOMZE, I.M. (2002): A complementary pivoting approach to the maximum weight clique problem. **SIAM Journal on Optimization**, 12, 928-948.

[18] MICHALEWICZ, Z. y SCHOENAUER, M. (1996): Evolutionary Algorithms for Constrained Parameter Optimization Problems. **Evolutionary Computation**, 4, 1-32.
[19] ÖSTERGÅRD, P. (2002): A fast algorithm for the maximum clique problem. **Discrete Applied Mathematics**, 120, 197-207.
[20] PROSSER P. (2012): Exact Algorithms for Maximum Clique A Computational Study. **Algorithms**, 5, 545-587.
[21] YAMAGUCHI, K. y MASUDA, S. (2008): A New Exact Algorithm for the Maximum Weight Clique Problem. **23rd International Conference on Circuits/Systems, Computers and Communications** (ITC-CSCC'08).