Control difuso del parámetro β de una heurística constructiva tipo GRASP para el problema de la bisección de vértices de un grafo

Javier Alberto Rangel González, Jesús David Terán Villanueva, Héctor Joaquín Fraire Huacuja, Juan Javier González Barbosa, José Antonio Martínez Flores, Guadalupe Castilla Valdez, Apolinar Ramírez Saldivar

Instituto Tecnológico de Ciudad Madero (ITCM), Cd. Madero, Tam., México

javieralberto64@hotmail.com, david_teran01@yahoo.com.mx, automatas2002@yahoo.com.mx, jjgonzalezbarbosa@hotmail.com, jose.mtz@itcm.edu.mx, gpe_cas@yahoo.com.mx, apolinar_r@yahoo.com

Resumen. El problema de la bisección de vértices de un grafo conexo es un problema de la clase NP-duro con importantes aplicaciones prácticas y que consiste en separar los vértices en dos conjuntos del mismo (o casi del mismo) tamaño de tal manera que se minimice la cantidad de vértices que tengan al menos una arista entre ambos conjuntos. A pesar de que actualmente se considera que los métodos heurísticos constituyen una de las herramientas más prometedoras para la solución de problemas NP-duros, en la literatura revisada no se encontraron reportes de soluciones del problema con métodos heurísticos. En éste artículo se describe una heurística constructiva de tipo GRASP, en la que el parámetro β se ajusta automáticamente usando un sistema de control difuso. Se presenta evidencia experimental de que el control difuso mejora el desempeño del algoritmo y se verifica que las diferencias observadas son estadísticamente significativas aplicando una prueba de hipótesis no paramétrica.

Palabras clave: controlador difuso, sistema difuso, control de parámetros, GRASP.

1. Introducción

El problema de la bisección de vértices de un grafo conexo es un problema de la clase NP-duro, que consiste en separar los vértices en dos conjuntos del mismo (o casi del mismo) tamaño de tal manera que se minimice la cantidad de vértices que tengan al menos una arista entre ambos conjuntos [1], [2], [3]. Este problema tiene una aplicación práctica en la distribución de información en redes de comunicaciones. Las tres principales tareas que se realizan en una red para distribuir la información son: divulgación, acumulación y *gossip* [4]. Para realizar eficientemente la divulgación, la técnica de *gossip* divide la red en dos subgrafos con aproximadamente el mismo número de nodos e identifica los nodos que conectan ambas regiones de la red. Dichos

nodos se utilizan para acumular la información en la región en que se ubican y los enlaces son utilizados para transmitir información de una región a la otra. A pesar de la relevancia práctica del problema en la literatura revisada solo se encontró un reporte en el que se proponen dos métodos exactos [5]. En dicho trabajo describen un método basado en la técnica de ramificación y acotamiento y uno basado en planos de corte y se considera que el grafo tiene un número de vértices par y que todas las aristas tienen un peso asignado. El problema consiste en separar los vértices del grafo en dos conjuntos del mismo tamaño, de tal manera que se minimice el costo total de los cruces entre dichos conjuntos. Si se considera que los costos de las aristas son uno, el problema se reduce al de la bisección de vértices. Una limitación de ese trabajo [5] es que no se aplica cuando el número de vértices es impar y que solo resuelve el problema para instancias de pequeña o en el mejor de los casos de mediana escala. Actualmente se considera que los métodos heurísticos constituyen una de las herramientas más prometedoras para la solución de instancias grandes de problemas NP-duros.

En éste artículo se describe una heurística constructiva de tipo GRASP [6] [7] en la que el parámetro β se ajusta automáticamente usando un sistema de control difuso.

2. Marco teórico

2.1. Sistemas de inferencia difusa

La lógica difusa fue propuesta por Lofthi Zadeh como una extensión de la lógica tradicional (booleana), la cual utiliza como concepto básico el grado de pertenencia (o membresía) a un conjunto difuso [8]. Los conjuntos difusos se aplican cuando no es muy clara la pertenencia de un elemento a un conjunto.

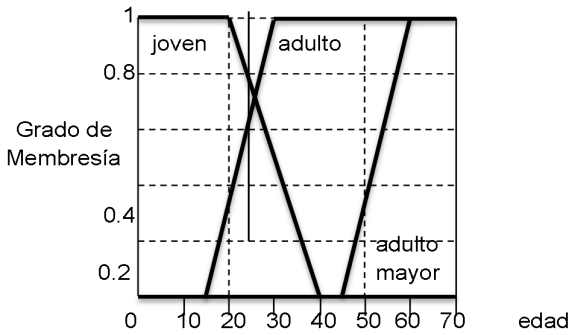

Fig. 1. Ejemplo de conjuntos difusos.

Una persona de 23 años de edad, ¿es un joven o un adulto?; una persona de 1.80 metros, ¿es alto o no?, una estrella de mar, ¿es un animal o no? Si se usa la lógica tradicional, no se podría contestar a estas preguntas de forma precisa. Sin embargo, la lógica difusa permite definir relaciones de pertenencia en una escala del cero al 1. Una persona de 23 años de edad pertenecerá al conjunto "joven" con un valor de .85

aproximadamente y al conjunto "adulto" con un valor de 0.55 aproximadamente (ver Fig. 1)

Los sistemas de inferencia difusa (FIS, por sus siglas en inglés), transforman una serie de entradas no difusas en un conjunto de salidas no difusas. Una vez realizada la transformación las salidas son utilizadas para tomar algún tipo de decisión. Para realizar el proceso estos sistemas utilizan conjuntos difusos, funciones de membresía asociadas a las entradas y las salidas, operadores lógicos difusos, reglas de inferencia difusa de tipo si-entonces y un motor de inferencia difusa. Estos sistemas han sido usados en una gran variedad de áreas tales como el control automático, la clasificación de datos, el análisis de decisiones, los sistemas expertos, la predicción de series de tiempo, la robótica y en el reconocimiento de patrones. La figura siguiente muestra la arquitectura de un FIS.

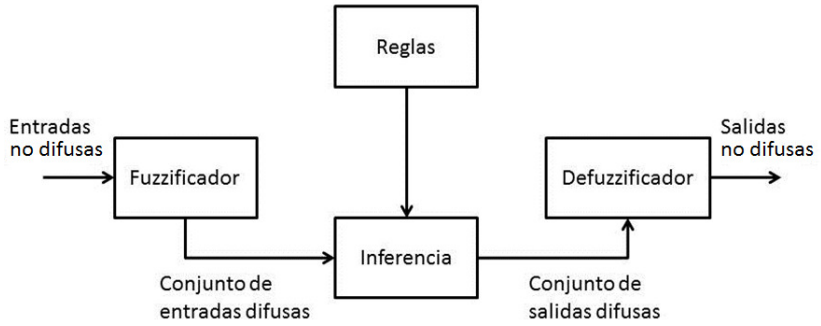

Fig. 2. Arquitectura de un sistema de inferencia difusa (FIS).

El fuzzificador utiliza funciones de membresía para transformar las entradas no difusas en valores difusos que activan el motor de inferencia. Las reglas de inferencia utilizan operadores lógicos difusos para establecer como se relacionan los valores difusos de las entradas con los valores difusos de las salidas. El motor de inferencia realiza en paralelo las transformaciones establecidas en todas las reglas de inferencia. Los modelos de tipo Mamdani y Sugeno son los que se utilizan con mayor frecuencia en la implementación del motor de inferencia [9], [10]. Finalmente el módulo de defuzzificación transforma los valores difusos de las salidas en valores no difusos.

3. Heurística constructiva tipo GRASP con control difuso

3.1. Descripción del algoritmo

La metodología GRASP fue propuesta por Feo y Resende [6], [7] y el acrónimo fue establecido por Feo et al. en [7]. El algoritmo GRASP es un algoritmo de reinicio el cual consiste principalmente de una etapa de construcción y de un algoritmo de búsqueda local. En este artículo se muestra un algoritmo constructivo tipo GRASP diseñado para el problema de Bisección de Vértices (ver Figura 3).

Al inicio del algoritmo se inicializa la cantidad de elementos seleccionados (Elem), todos los vértices se encuentran en el conjunto derecho y se considera que cada vértice tiene a todos sus adyacentes (Gradov) en el lado derecho (AdyToR). El primer elemento seleccionado será aquél con menor grado, aquél que tenga menos enlaces a elementos de la derecha, este elemento será removido del conjunto derecho y asignado al izquierdo. Además, se realizará una actualización de todos los adyacentes del elemento seleccionado, esto se hace con el fin de identificar aquéllos vértices que tienen menos conexiones hacia el conjunto de la derecha (AdyToR). Después se creará la lista de candidatos (LC) con todos los elementos que todavía se encuentran en el conjunto de la derecha (R). Y la lista de candidatos restringida (LCR) contendrá los mismos elementos que LC, menos aquellos elementos que tengan más adyacentes a la derecha que los definidos por la variable Límite. Siendo el Límite un valor entre el mínimo y máximo número de adyacentes a la derecha de todos los vértices en LC, el valor del límite depende de la variable β que es un valor entre cero y uno. Si la variable β tiende a cero entonces el límite será muy restrictivo, mientras que si β tiende a uno entonces el límite será muy laxo. Luego se selecciona un vértice (u) de LCR de manera aleatoria, se elimina del conjunto de la derecha (R) y se incluye en el conjunto de la izquierda (L). Posteriormente se actualizan los adyacentes hacia la derecha AdyToR disminuyendo en uno a todos los vértices que son adyacentes a u. Se continúa con este proceso hasta que se hayan movido, al conjunto de la izquierda, la mitad de vértices del grafo.

```
Elem = 0
AdyToR_v = Grado_v                    ∀v ∈ V
R ← {v}                               ∀v ∈ V
L = ∅
u = ArgMin_v(Grado_v)
L ← {u}
R = R \ {u}
AdyToR_v--                            ∀v ∈ V|∃(u, v) ∈ E
Elem++
do{
  LC = R
  Limite = Min(AdyToR_v) + β(Max(AdyToR_v) −
Min(AdyToR_v))      ∀v ∈ LC
  LCR = LC \ v ∈ LC | AdyToR_v > Limite
  u = SelectAleatoria(LCR)
  L ← {u}
  R = R \ {u}
  AdyToR_v--                          ∀v ∈ V|∃(u, v) ∈ E
  Elem++
}while(Elem < ⌊|V|/2⌋)
```

Fig. 3. Algoritmo constructivo tipo GRASP.

3.2. Controlador difuso del parámetro β

Al algoritmo anterior se le incorporó un controlador difuso para ajustar automáticamente el parámetro β. El controlador difuso recibe como variable de entrada no difusa la β y el $\%Err$ de la solución actual con respecto a la mejor observada en el proceso de búsqueda y como variable de salida no difusa $\Delta\beta$, el cual especifica un incremento en la β.

En cada iteración el algoritmo genera la solución actual, se actualiza la mejor solución encontrada y se determina el porcentaje de error ($\%Err$). Con la β y $\%Err$ se invoca el controlador difuso, para determinar el valor de $\Delta\beta$, actualizar el valor de β y realizar la siguiente iteración.

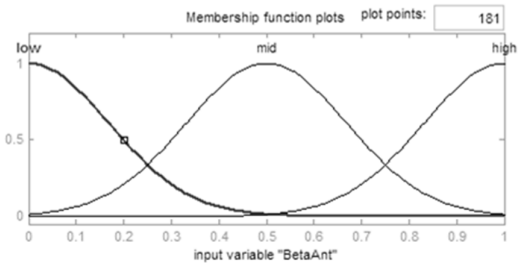

Fig. 4. Variable de entrada "β"

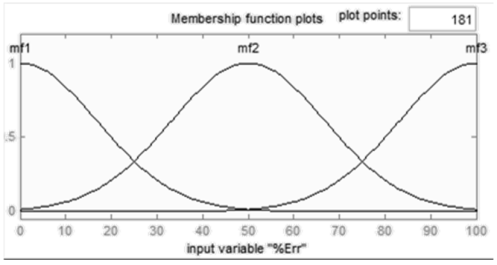

Fig. 5. Variable de entrada "$\%Err$"

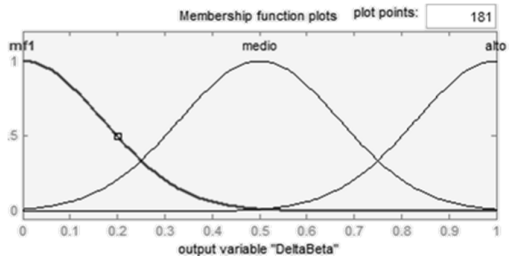

Fig. 6. Variable de salida "$\Delta\beta$"

Las entradas del controlador son β y $\%Err$ y la salida es $\Delta\beta$. El controlador usa 3 funciones de membresía gaussianas (bajo, medio, alto) tanto para el parámetro de

entrada como el de salida. Estas funciones de membresía tienen la característica de ser simétricas y con traslape. Las siguientes figuras (Figura 4, Figura 5 y Figura 6) muestran las funciones de membresía utilizadas.

Las reglas que utiliza el controlador son las siguientes:

1. Sí, β es bajo y $\%Err$ es bajo entonces $\Delta\beta$ será alto;
2. Sí, β es bajo y $\%Err$ es medio entonces $\Delta\beta$ será medio;
3. Sí, β es bajo y $\%Err$ es alto entonces $\Delta\beta$ será alto;
4. Sí, β es medio y $\%Err$ es bajo entonces $\Delta\beta$ será bajo;
5. Sí, β es medio y $\%Err$ es medio entonces $\Delta\beta$ será bajo;
6. Sí, β es medio y $\%Err$ es alto entonces $\Delta\beta$ será alto;
7. Sí, β es alto y $\%Err$ es bajo entonces $\Delta\beta$ será medio;
8. Sí, β es alto y $\%Err$ es medio entonces $\Delta\beta$ será alto;
9. Sí, β es alto y $\%Err$ es alto entonces $\Delta\beta$ será medio;

El motor de inferencia que utiliza el controlador es de tipo Mamdani.

4. Resultados experimentales

Toda la experimentación se realizó en una computadora portátil con las siguientes características: sistema operativo Windows 7 de 64 bits con service pack 1, procesador Intel core i7 a 3.1 GHz y memoria RAM de 8 GB.

El algoritmo constructivo tipo GRASP fue implementado en C++, y se implementaron dos versiones del controlador difuso. Una utiliza una librería desarrollada por los autores en Java y la otra utiliza la librería disponible en Matlab.

En los experimentos realizados se comparan los siguientes algoritmos: uno en el que el parámetro β tiene un valor fijo de .5, otro en el que el parámetro β se ajusta con el controlador implementado en Java, y otro en el que el parámetro β se ajusta con el controlador implementado en Matlab. Para comparar el desempeño de estos algoritmos se resolvieron con cada uno de ellos 87 instancias Harwell-Boeing [11]. Estas constituyen el conjunto de instancias más difíciles con las qué se evalúa el desempeño de algoritmos heurísticos para problemas de ordenamiento lineal.

Tabla 1. Resultado de las instancias Harwell-boeing.

Instancias	Sin controlador difuso	Java	Matlab
494_bus.mtx.rnd	194	161	154
662_bus.mtx.rnd	289	226	230
685_bus.mtx.rnd	328	257	259
arc130.mtx.rnd	65	33	29
ash292.mtx.rnd	145	129	127
ash85.mtx.rnd	41	35	33
bcspwr01.mtx.rnd	16	10	7

Instancias	Sin controlador difuso	Java	Matlab
bcspwr02.mtx.rnd	19	13	10
bcspwr03.mtx.rnd	55	36	37
bcspwr04.mtx.rnd	119	102	99
bcspwr05.mtx.rnd	192	149	148
bcsstk01.mtx.rnd	24	23	20
bcsstk02.mtx.rnd	33	33	33
bcsstk04.mtx.rnd	66	66	66
bcsstk05.mtx.rnd	75	73	74
bcsstk06.mtx.rnd	197	199	200
bcsstk20.mtx.rnd	190	187	195
bcsstk22.mtx.rnd	49	43	37
bcsstm07.mtx.rnd	199	201	204
can__144.mtx.rnd	72	63	59
can__161.mtx.rnd	80	71	75
can__292.mtx.rnd	144	126	124
can__445.mtx.rnd	220	206	209
curtis54.mtx.rnd	26	20	17
dwt__209.mtx.rnd	103	92	91
dwt__221.mtx.rnd	110	98	98
dwt__234.mtx.rnd	56	34	31
dwt__245.mtx.rnd	113	98	97
dwt__310.mtx.rnd	154	143	138
dwt__361.mtx.rnd	179	166	162
dwt__419.mtx.rnd	209	197	196
dwt__503.mtx.rnd	248	240	241
dwt__592.mtx.rnd	295	276	276
fs_183_1.mtx.rnd	83	67	66
fs_541_1.mtx.rnd	262	263	257
fs_680_1.mtx.rnd	308	240	234
gent113.mtx.rnd	50	43	47
gre_216a.mtx.rnd	104	98	99
gre__115.mtx.rnd	54	43	44
gre__185.mtx.rnd	91	86	82
gre__343.mtx.rnd	164	159	156
gre__512.mtx.rnd	247	240	241
hor__131.mtx.rnd	215	204	206
ibm32.mtx.rnd	15	13	13

Instancias	Sin controlador difuso	Java	Matlab
impcol_a.mtx.rnd	100	86	88
impcol_b.mtx.rnd	25	26	24
impcol_c.mtx.rnd	67	55	52
impcol_d.mtx.rnd	209	188	185
impcol_e.mtx.rnd	106	104	100
lns__131.mtx.rnd	54	42	40
lns__511.mtx.rnd	228	206	201
lund_a.mtx.rnd	70	70	71
lund_b.mtx.rnd	70	70	72
mbeacxc.mtx.rnd	243	243	242
mcca.mtx.rnd	81	75	75
nnc261.mtx.rnd	119	106	111
nnc666.mtx.rnd	313	282	295
nos1.mtx.rnd	76	61	58
nos2.mtx.rnd	314	268	275
nos4.mtx.rnd	45	38	41
nos5.mtx.rnd	233	230	229
nos6.mtx.rnd	330	279	286
plat362.mtx.rnd	181	178	174
plskz362.mtx.rnd	179	149	149
pores_1.mtx.rnd	13	13	13
pores_3.mtx.rnd	217	211	212
saylr1.mtx.rnd	116	97	98
saylr3.mtx.rnd	333	275	277
sherman4.mtx.rnd	266	236	244
shl__200.mtx.rnd	288	292	285
shl__400.mtx.rnd	286	291	285
shl____0.mtx.rnd	288	287	280
steam1.mtx.rnd	120	117	117
steam2.mtx.rnd	300	300	300
steam3.mtx.rnd	40	36	36
str__200.mtx.rnd	167	172	173
str__600.mtx.rnd	170	171	175
str____0.mtx.rnd	166	164	167
west0132.mtx.rnd	65	52	56
west0156.mtx.rnd	76	67	65
west0167.mtx.rnd	81	71	73
west0381.mtx.rnd	189	184	183

Instancias	Sin controlador difuso	Java	Matlab
west0479.mtx.rnd	227	222	221
west0497.mtx.rnd	238	214	219
west0655.mtx.rnd	313	311	307
will199.mtx.rnd	96	91	91
will57.mtx.rnd	24	17	18
Promedios	149.6551724	136.88505	136.597701

En estos resultados se observa que en casi todos los casos se reduce o se empata el valor objetivo al emplear un controlador difuso, ya sea en Java o Matlab. También podemos ver que se reduce el valor promedio de la función objetivo al emplear cualquier variante de los controladores difusos.

Los resultados de la prueba no paramétrica de Wilcoxon muestran que las mejoras observadas al utilizar los controladores difusos son estadísticamente significativas (con una certidumbre superior al 99%). Por otra parte, la prueba de Wilcoxon muestra resultados de equivalencia estadística entre los resultados obtenidos con la aplicación en Java y los resultados de Matlab. Estos resultados muestran que la implementación en Java posee la misma calidad que Matlab, y por lo tanto es una alternativa viable para realizar pruebas de sistemas difusos.

5. Conclusiones

En éste artículo se describe una heurística constructiva de tipo GRASP, en la que el parámetro β se ajusta automáticamente usando un sistema de control difuso. La evidencia experimental muestra que el control difuso mejora el desempeño del algoritmo y se verifica que las diferencias observadas son estadísticamente significativas. Se considera que este enfoque puede ser aplicado para controlar el parámetro β de heurísticas tipo GRASP aplicadas a otros problemas.

Referencias

1. Díaz, J., Petit, J., & Serna, M.: A Survey on Graph Lay-out Problems. CM Computing Surveys (CSUR), 34(3), pp. 313–356 (2000)
2. F. D. Brandes Ulrik: Vertex Bisection is Hard, too. Journal of Graph Algorithms and Applications (2009)
3. Fraire, Héctor, et al.: Exact Methods for the Vertex Bisection Problem. Recent Advances on Hybrid Approaches for Designing Intelligent Systems. Springer International Publishing. pp. 567–577 (2014)
4. Hans-Joachim, B.: Two open problems in communi-cation in edge-disjoint paths modes. Acta Mathematica et Informatica Universitatis Ostraviensis, Vol. 7, No. 1 (1999)
5. Karisch, S. E., Rendl, F., Clausen, J.: Solving graph bisection problems with semidefinite programming. INFORMS Journal on Computing, 12(3), pp. 177–191 (2000)

6. Feo, T. A., Resende, M. G.: A probabilistic heuristic for a computationally difficult set covering problem. Operations research letters, 8(2), pp. 67–71 (1989)
7. Feo, T. A., Resende, M. G.: Greedy randomized adaptive search procedures. Journal of global optimization, 6(2), pp. 109–133 (1995)
8. Zadeh, L.: Outline of a new approach to the analysis of complex systems and decision processes. IEEE Transactions on Systems, Man, and Cybernetics, Vol. 3, No. 1, pp. 28–44 (1973)
9. Mamdani, E. H., & Assilian, S. An experiment in linguistic synthesis with a fuzzy logic controller. International journal of man-machine studies, 7(1), pp. 1–13 (1975)
10. Sugeno, M.: Fuzzy measures and fuzzy integrals: a survey. Fuzzy Automata and Decision Processes, pp. 89–102 (1977)
11. Rojas, R.: Neural Networks: A systematic introduction. Springer, pp. 289–291 (1996)