Algoritmo memético aplicado a la solución del problema de asignación generalizada

Memetic algorithm applied to the solution of generalized assignment problem

Eliana Mirledy Toro Ocampo
Ingeniera Industrial y candidata a Magíster en Ingeniería Eléctrica en la línea de Investigación operativa de la Universidad Tecnológica de Pereira. Docente Facultad de Ingeniería Industrial, Universidad Tecnológica de Pereira.
eliana@ohm.utp.edu.co

Mauricio Granada Echeverry
Ingeniero Electricista y Magíster en Ingeniería Eléctrica en la línea de planeamiento eléctrico de la Universidad Tecnológica de Pereira. Docente Facultad de Ingeniería Eléctrica, Universidad Tecnológica de Pereira.
magra@utp.edu.co

Rubén Romero
Ingeniero Electricista de la Universidad de Perú (UNI-Perú). Magíster y Ph.D. en Ingeniería Eléctrica, Universidad Unicamp-Brasil. Docente Universidad UNESP-Brasil.
ruben@dsee.fee.unicamp.br

Fecha de recepción: abril 15 de 2005
Clasificación del artículo: investigación
Fecha de aceptación: junio 27 de 2005

Palabras clave: algoritmos genéticos, asignación generalizada, optimización, combinatorial.
Key words: genetic algorithm, generalized assignment, combinatorial, optimization.

Resumen

Este artículo presenta una modificación al algoritmo genético desarrollado por Chu-Beasley aplicado al problema de asignación generalizada. Se utiliza una propuesta basada en factores de sensibilidad para la generación de la población inicial y se implementa un modelo matemático que incorpora factores de penalización.

Abstract

This paper presents a modification to the genetic algorithm developed by Chu-Beasley, applied to the problem of generalized assignment. A proposal based on factors of sensitivity for the generation of the inicial conditions is used and a mathematical model is implemented that incorporates penalty factors.

1. Introducción

El problema de asignación generalizada (*PAG*) es uno de los clásicos de la investigación de operaciones. Consiste en encontrar una adecuada asignación de *m* tareas a *n* agentes, de manera que se minimice el costo total de asignación. La metodología de solución empleada en este trabajo se basa en la adecuación del algoritmo genético (*AG*) planteado por Chu-Beasley, modificando el modelo matemático de forma que incorpore factores de penalización y realizando la construcción de una población inicial conveniente (Beasley, 1997 y Granada, 2004).

En la literatura especializada se encuentran soluciones al PAG utilizando metaheurísticas como Búsqueda Tabú (Granada, 2004), las cuales son utilizadas para encontrar soluciones de buena calidad en problemas de gran tamaño. En el campo de la optimización clásica, se destacan los algoritmos basados en *Branch and Bound,* los cuales resultan eficaces en problemas de pequeño y mediano tamaño.

2. Problema de asignación generalizada

El modelo matemático modificado usando factores de penalización es:

$$\min \sum_{i=1}^{n}\sum_{j=1}^{m} C_{ij}X_{ij} + \lambda \sum_{i\in\{\text{restricciones violadas}\}} \left(\alpha_i \underbrace{\left(\sum_{j=1}^{n} A_{ij}X_{ij} - b_i \right)}_{\text{cantidad de recursos excedidos}} \right)$$

s.a.

$$\sum_{i=1}^{n} X_{ij} = 1 \quad X \in \{0,1\} \qquad (1)$$

Una descripción detallada del modelo matemático utilizado en este trabajo fue hecha en (Granada, 2004), en donde *Cij* es el costo de designar la tarea *j* al agente *i*; *m* es el número de tareas y *n* el número de agentes; X_{ij} corresponde a los elementos de la matriz de alternativas; A_{ij} representa los recursos consumidos por el agente *i* al realizar la tarea *j*; representan la capacidad total de recursos del agente *i*; λ es el parámetro encargado de controlar el peso o la importancia relativa de la restricción con respecto a la solución de costo actual y α es el factor que relaciona costo por unidad de recurso del agente *i*. Este último parámetro se calcula como:

$$\alpha_i = \frac{\sum_{j=1}^{m} C_{ij}X_{ij}}{b_i} \qquad (2)$$

La población inicial de alternativas puede generarse de manera aleatoria o usando factores de sensibilidad (Granada, 2004). Es importante notar que el segundo factor de la función objetivo planteada en (1) corresponde a la penalización; este factor es fundamental en el momento de cuantificar la infactibilidad de una alternativa en términos de costos.

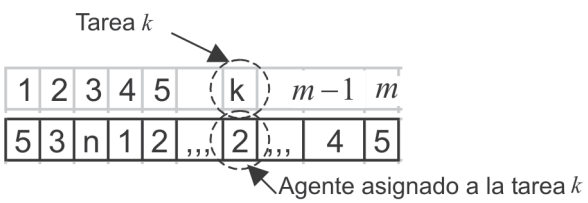
3. Algoritmo genético

Las principales características del AG de Chu-Beasley, consisten en mantener constante el tamaño de la población de alternativas de solución; de esta manera, en cada iteración se reemplaza una alternativa de la población usando un mecanismo eficiente de modificación de la misma. Este mecanismo busca beneficiar las alternativas menos infactibles y de mejor calidad; en cada iteración, la población es reemplazada sistemáticamente por un único descendiente generado. La estrategia tiene la ventaja que permite encontrar múltiples soluciones.

3.1 Cromosoma

La codificación utilizada para representar cada una de las alternativas de solución del PAG consiste en construir un vector de tamaño *m*. La *k-ésima* posición del vector representa la *k-ésima* tarea, y el elemento almacenado en esa posición corresponde al agente que realiza dicha tarea. La población de alternativas de solución se conforma por un número determinado de cromosomas, como se muestra en la figura 1.

Figura 1. Codificación de alternativas



3.2. Selección

En el proceso de selección es necesario definir el número de alternativas que serán escogidas de la población de manera aleatoria; posteriormente, estas alternativas compiten a fin de seleccionar una alternativa padre. El proceso de selección utilizado debe hacerse dos veces, de manera que se obtengan dos padres. Se proponen dos algoritmos de competencia para la selección de estos padres: el primero, denominado selección por ruleta, consiste en una escogencia basada en el inverso de valor de la función objetivo de cada alternativa ($1/Fobj_k$), donde las mejores tienen mayor probabilidad de ser seleccionadas como padre. Dicha probabilidad, asociada a cada alternativa, se calcula encontrando el porcentaje del valor anterior con respecto al valor que representa la función objetivo total (*FobjTotal*), el cual se calcula como:

$$FobjTotal = \sum_{i=1}^{k} \left(\frac{1}{Fobj_k} \right) \qquad (3)$$

Se consideran los inversos, dado que se trata de un problema de minimización, en el cual se requiere asignar mayor probabilidad a las alternativas con menor costo de asignación.

El proceso de ruleta consiste en asignar a cada alternativa el porcentaje correspondiente de la función objetivo total. Posteriormente se escoge un porcentaje aleatorio entre 0 y 100%, el cual determina el intervalo de la alternativa $padre_j$. Para ilustrar el proceso de ruleta con un ejemplo, supóngase que se tienen los siguientes parámetros:

$k = 5; Fobj_1 = 1.702; Fobj_2 = 1.705; Fobj_3 = 1.700; Fobj_4 = 1.706; Fobj_5 = 1.902$

Empleando (3) se tiene que *FobjTotal* = 0,0029. Así, los porcentajes de cada alternativa son:

$$\%Fobj_1 = \frac{\frac{1}{1.702} \times 100}{0.0029} \approx 20,2601\%$$

$$\%Fobj_2 \approx 20,2245\%; \%Fobj_2 \approx 20,2840\%;$$

$$\%Fobj_2 \approx 20,2126\%; \%Fobj_2 \approx 19,0188\%$$

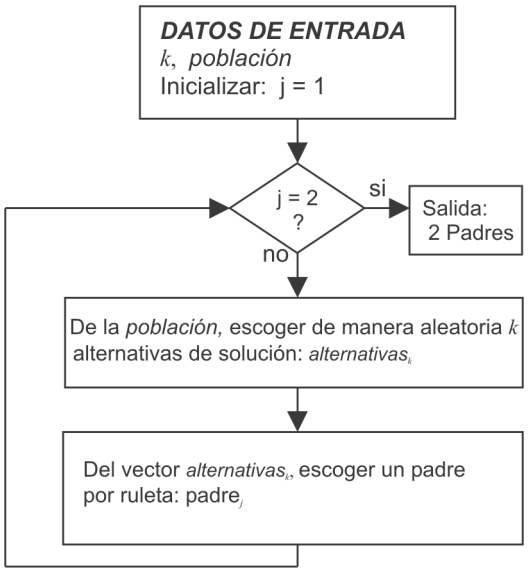
Finalmente, se genera un número aleatorio que sirve para seleccionar uno de los padres. Si por ejemplo, el número aleatorio generado es 62,8271 entonces, como se muestra en la tabla 1, la alternativa seleccionada como padre es la 4.

Tabla 1. Selección por ruleta

Alternativa	Porcentaje	Intervalo	
1	20,2601	0 - 20,2601	
2	20,2245	20,2602 - 40,4846	
3	20,2840	40,4846 - 60,7686	
4	20,2126	60,7687 - 80,9812	← Intervalo seleccionado
5	19,0188	80,9813 - 100	

En la figura 2 se muestra el proceso completo de selección de dos padres.

Figura 2. Proceso de selección



El segundo algoritmo, denominado selección por torneo es bastante simple: consiste en escoger la mejor de las *k* alternativas seleccionadas en forma aleatoria. El éxito de aplicar esta metodología radica en escoger adecuadamente un valor de *k* que se ajuste a cada problema en particular, teniendo en cuenta el tamaño y complejidad del problema y la

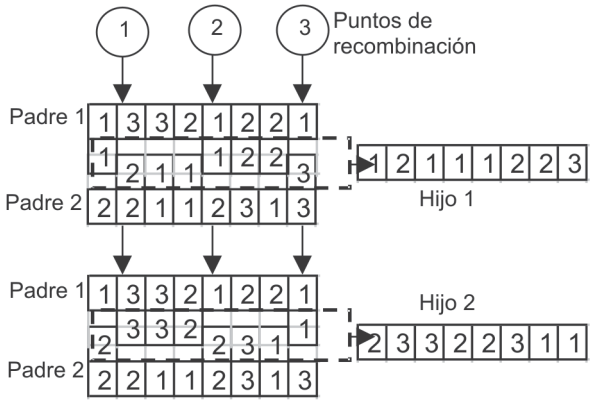
magnitud de la población inicial. Un valor de *k* muy alto y la generación de una población inicial pequeña puede, eventualmente, hacer elitista el proceso de selección y ocasionar convergencias locales. Un valor muy bajo de *k* puede ocasionar un mayor esfuerzo computacional.

3.3 Recombinación

Con los dos padres seleccionados, el siguiente paso consiste en combinarlos de forma tal que se obtenga solo un descendiente. Lo anterior representa una diferencia significativa con respecto al método tradicional de AG, y se traduce, en este caso, en una estrategia eficiente de búsqueda local.

En el proceso de recombinación es necesario definir el número *p* de puntos de recombinación. Estos puntos se escogen de manera aleatoria sobre el cromosoma; posteriormente se combinan las características de los padres realizando un cruzamiento de las porciones de cromosoma existentes entre cada punto de recombinación, como se muestra en la figura 3, para un cromosoma de ocho elementos y tres puntos de recombinación.

Figura 3. Ejemplo de recombinación de 3 puntos



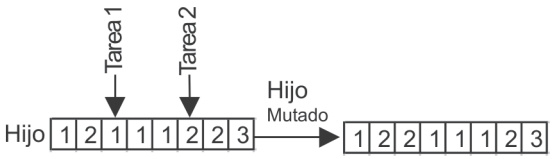
El resultado de la recombinación produce dos hijos, de los cuales uno es desechado de manera aleatoria. Otra estrategia es hacerlos competir por torneo o por ruleta.

3.4 Mutación

El proceso de mutación se encuentra fuertemente ligado al concepto de vecindad. Así, una mutación drástica puede alejar el proceso de la zona de búsqueda actual y producir oscilaciones alrededor de soluciones de buena calidad; de otra parte, una mutación débil, eventualmente puede ocasionar convergencias a soluciones locales de mala calidad. Por tanto, el proceso de mutación permite una amplia gama de propuestas.

En este trabajo, el cromosoma hijo obtenido en el proceso de recombinación es sometido al proceso de mutación. La mutación aplicada consiste en escoger, de manera aleatoria, dos tareas e intercambiar los agentes asociados a cada una, como se muestra en la figura 4.

Figura 4. Ejemplo de mutación



Para efectuar mutaciones más fuertes es posible aplicar dos o más veces el proceso descrito al mismo hijo. Las pruebas realizadas muestran que el mejor desempeño se obtiene aplicando la mutación por probabilidad, de manera que se defina aleatoriamente si no se aplica mutación, se aplica una (mutación suave) o dos veces (mutación agresiva). La propuesta anterior se plantea considerando una probabilidad igual para cada una de las tres estrategias.

3.5 Disminuir infactibilidad

El modelo planteado en este trabajo tiene en cuenta la infactibilidad de manera implícita, al considerar factores de penalización; además, el algoritmo que se propone incluye un proceso que busca disminuirla gradualmente. Para alternativas infactibles, el proceso consiste en reasignar una tarea perteneciente a un agente violado a otro agente con

capacidad para realizarla. Una alternativa es infactible cuando uno o más de sus agentes violan los limites máximos de consumo de recursos; así, es importante que el agente que asume la nueva tarea no pase a violar el límite máximo de recursos. La primera reasignación que disminuya la infactibilidad y no viole ninguna restricción, es la que se toma como definitiva. La infactibilidad se cuantifica mediante la siguiente expresión:

$$\sum_{i\in\left\{\begin{matrix}\text{restricciones}\\ \text{violadas}\end{matrix}\right\}}\left(\alpha_i \underbrace{\left(\sum_{j=1}^{n} A_{ij}X_{ij} - b_i\right)}_{\text{cantidad de recursos excedidos}}\right) \qquad (4)$$

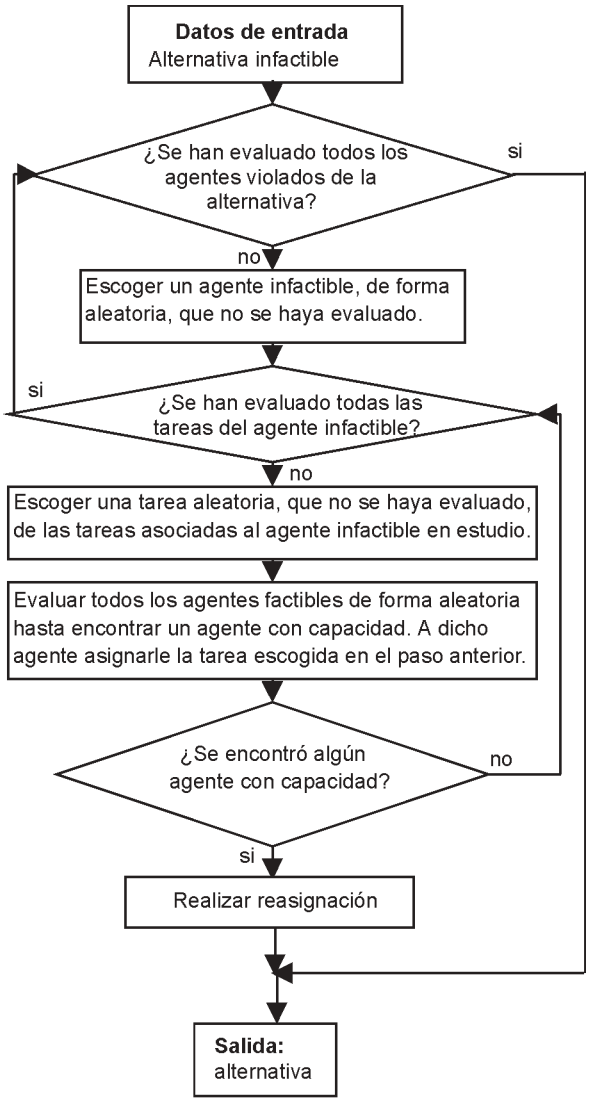
Este proceso puede llevarse a cabo consecutivamente dos o más veces para eliminar más rápido las configuraciones infactibles; los mejores resultados se obtuvieron aplicando una sola vez el proceso. En la figura 5 se describe el proceso completo.

3.6 Mejorar optimalidad

Este proceso busca mejorar el valor de la función objetivo y, al igual que el anterior, puede hacerse consecutivamente dos o más veces para disminuir con rapidez el valor de la función objetivo. Lo anterior puede traer como consecuencia convergencias a soluciones de mala calidad, por lo que deben elegirse valores adecuados para cada caso específico.

El proceso consiste en evaluar la alternativa actual haciendo que cada tarea sea realizada por cada agente diferente, teniendo cuidado de realizar una reasignación a la vez. Las reasignaciones que mejoren la función objetivo y no violen la capacidad del agente en estudio son almacenadas; finalmente, de todas las reasignaciones almacenadas se escoge la que más favorezca la función objetivo. Es importante notar que la alternativa resultante se diferencia de la alternativa original solo en un elemento; el proceso completo se describe en el diagrama de flujo mostrado en la figura 6.

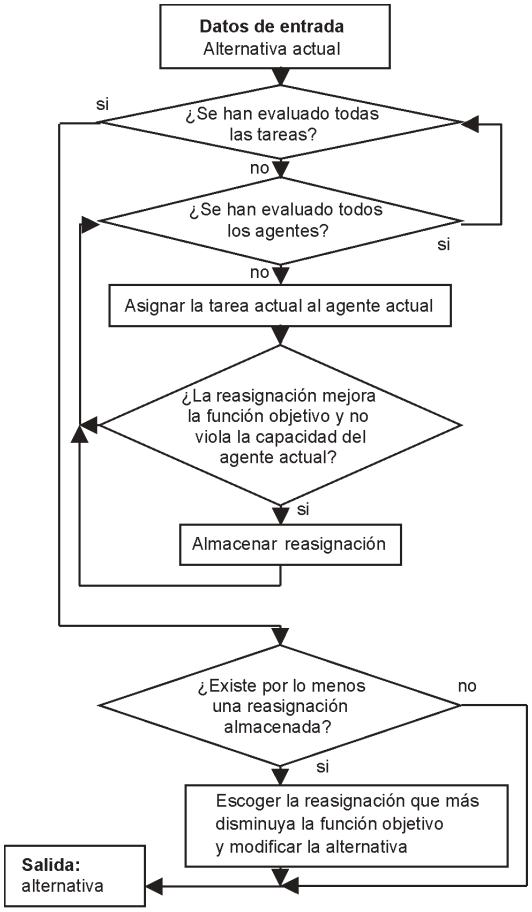
Figura 5. Proceso de mejoramiento de optimalidad



3.7 Modificar la población actual

Las principales características del algoritmo memético que aquí se presenta están asociadas a la modificación de la población actual. El algoritmo memético completo, después de generada la población inicial, consiste en repetir un número determinado de iteraciones los pasos siguientes del 1 al 6.

Figura 6. Proceso de disminución de infactibilidad



1. Se obtienen dos alternativas *padre* por *selección* de la población actual.
2. Se obtiene una alternativa *hijo* aplicando *recombinación* a los *padres* obtenidos en el paso anterior.
3. Se obtiene una *alternativa modificada* aplicando *mutación*.
4. Si la configuración es infactible se *mejora la infactibilidad* y se obtiene una alternativa menos infactible. De lo contrario ir al paso 5.
5. Se *mejora la optimalidad* de la alternativa en estudio.
6. Si la *alternativa resultante* de aplicar los pasos anteriores no se encuentra en la población, entonces aplicar estrategia de *modificación de la población*; de lo contrario volver al paso 1.

Para modificar la población se propone la siguiente estrategia:

1. Si la alternativa actual es infactible y a su vez es menos infactible que la peor infactible de la población, entonces reemplazar la peor infactible por la alternativa actual.
2. Si la configuración es factible y existe por lo menos una infactible en la población actual, entonces reemplazar la peor infactible por la alternativa actual.
3. Si la configuración es factible y toda las alternativas de la población actual son factibles, entonces reemplazar la alternativa con peor función objetivo por la alternativa actual. Lo anterior se hace sólo si la alternativa actual es de mejor calidad que la peor de la población.

La estrategia de modificación de la población actual se realiza cambiando sólo una alternativa por cada iteración, teniendo en cuenta que no se admiten alternativas repetidas. Lo anterior evita convergencias prematuras y asegura una exploración detallada de la región de soluciones. Además, pueden obtenerse múltiples soluciones de un mismo problema.

Esta estrategia busca preservar las mejores alternativas, asegurando factibilidad y optimalidad; estas características constituyen la principal diferencia con respecto al algoritmo propuesto por Chu-Beasley, en el cual la alternativa más infactible es reemplazada. En esta solución, a diferencia de los algoritmos genéticos tradicionales, no se modifica la población de manera aleatoria.

4. Población inicial

En este trabajo se propone e implementa un algoritmo que permite la obtención de una población inicial de alternativas de asignación que tiene en cuenta el estudio de sensibilidades. La matriz de sensibilidades S relaciona, elemento a elemento, la matriz de costos C con la matriz de recursos A, de manera que mide la aproximación de la relación

costo/recurso actual del agente *i* para realizar la tarea *j* con respecto a una relación costo/recurso ideal para realizar la misma tarea *j*, es decir:

$$S_{ij} = \frac{C_{ij}}{A_{ij}} - CA_{IDEAL\ j} \quad (5)$$

En (5), $CA_{IDEAL\,j}$ es la relación entre el menor costo de asignación asociado a la tarea *j* y el menor consumo de recurso asociado a la misma tarea *j*, esto es:

$$CA_{IDEAL\ j} = \frac{\min_{i=1}^{n}\left(C_{ij}\right)}{\min_{i=1}^{n}\left(A_{ij}\right)} \quad (6)$$

De esta manera, los elementos de la matriz más aproximados a cero, serán los que cumplen con una mejor relación costo/recurso y los agentes asociados a este elemento tendrán una probabilidad mayor al momento de decidir la asignación de la tarea *j*, la cual se realiza por ruleta. Un segundo criterio consiste en realizar una asignación de tareas de manera totalmente aleatoria.

5. Resultados

El algoritmo fue probado por medio de problemas bien conocidos en la literatura[1]. Los problemas típicos fueron clasificados en cinco grupos (*A-B-C-D-E*) según su grado de dificultad. Cada uno de estos grupos contiene seis problemas, de acuerdo con su tamaño. Así, existen tres problemas con $m = 100$ y $n \in \{5,10,20\}$ y otros tres problemas con $m = 200$ y $n \in \{5,10,20\}$. Los problemas tipo *A* son considerados los más sencillos; para los problemas *B* y *C*, los valores a_{ij} son enteros generados por una distribución uniforme $U(5, 25)$ y los valores c_{ij} son enteros generados por $U(10, 50)$. La capacidad de recurso b_j se encuentra usando $0{,}8/m \times \sum_i a_{ij}$ Para los problemas tipo *D*, a_{ij} son enteros generados usando $U(1, 100)$ y c_{ij} son enteros calculados usando $c_{ij} = 111 - a_{ij} + s$, donde *s* es un entero generado usando $U(-10, 10)$. Finalmente, para problemas tipo *E*, $a_{ij} = 1 - 10\ln(x)$, donde $x = U(0, 1)$, $c_{ij} = 1.000/a_{ij}$, donde $y = U(0, 1)$ y $b_j = \max\left\{0{,}8/m \times \sum_i a_{ij}, \max_j a_{ij}\right\}$. Los proble-

Tabla 2. Resultados obtenidos

Tipo	*n*	*m*	Mejor solución Chu-Beasley	Mejor solución AG	Número de soluciones encontradas	Tamaño de la población (alternativas)	*K*	*P*	Tiempo consumido (s)
A	5	100	1.698	1.698	32	100	5	3	15
A	20	200	2.339	2.339	18	100	5	3	48
B	5	100	1. 843	1.843	10	150	5	3	55
B	5	200	3.553	3.553	15	100	4	2	485
C	5	100	1.931	1.937	5	200	4	2	125
C	10	200	2.814	2.814	10	200	2	2	315
D	5	100	6.373	6.360	2	500	2	2	1.250
D	10	200	12.601	12.563	1	500	2	1	3.850
E	5	100	12.680	12.690	1	800	2	1	5.220
E	10	200	23.307	23.315	1	800	2	1	8.520

[1] Tales problemas pueden ser descargados de
http://mscmga.ms.ic.ac.uk/jeb/orlib/gapinfo.html
http://www.or.amp.i.kyoto-u.ac.jp/»yagiura/gap/

mas *B* y *C* son considerados más sencillos de resolver que los tipo D y E. En la tabla 2 se relacionan los resultados obtenidos en algunos de estos problemas, así como los parámetros que presentaron mejor desempeño.

En la tabla 2 se comparan los resultados obtenidos con las mejores soluciones conocidas y publicadas en la literatura especializada. Se encontró que para los casos tipo A, un valor de *K* alto (*K* = 5), sin aplicar mutación, una población inicial pequeña (100 individuos) y múltiples puntos de recombinación (*P* = 3), hacen que el algoritmo alcance la mejor solución conocida en pocas iteraciones; para casos de mayor tamaño y dificultad, estos parámetros deben ser calibrados de manera particular. Sin embargo, la tendencia es que en este tipo de problemas, los valores disminuyan para garantizar una mejor exploración del espacio de soluciones.

En problemas de baja complejidad, el proceso de selección por torneo mostró mejor desempeño que cuando se aplica selección por ruleta; en los otros casos el comportamiento del algoritmo fue similar. Cada caso fue sometido a 10 procesos de optimización, registrándose en la tabla la mejor solución encontrada.

El resultado más relevante se obtuvo para el caso tipo D, con $n = 10$ y $m = 200$. En este caso, la función de costo pasó de 12.601 a 12.563; el cromosoma de solución obtenido se muestra en la tabla 3.

En todos los casos evaluados, el proceso de mutación selectivo, en el que en forma probabilística se puede escoger no mutar, hacer una mutación suave o una agresiva, presentó los mejores resultados. De igual forma, las pruebas realizadas usando una población inicial basada en sensibilidades mostraron una disminución importante en el esfuerzo computacional al compararse con los resultados obtenidos usando población aleatoria. Las pruebas fueron hechas en un computador PC con procesador Pentium III de 700 MHz.

6. Conclusiones y comentarios

El algoritmo propuesto presenta resultados interesantes si se considera que permite obtener múltiples soluciones para un mismo problema. Asimismo, la estrategia planteada para modificar la población actual presenta un alto desempeño: se obtuvieron respuestas mejores que las obtenidas por Chu-Beasley y en un caso se disminuyó la mejor solución conocida.

La cuantificación de la infactibilidad se realiza de manera diferente a lo planteado por Chu-Beasley. En este trabajo se consideran factores de penalización y el proceso de disminución de infactibilidad; por tanto, es posible hacer perdurar por más o menos tiempo las alternativas infactibles si se da un valor adecuado al parámetro λ. Lo anterior puede mejorar la exploración del método y adquiere mayor importancia para problemas de alta complejidad, en los cuales el espacio de soluciones es reducido.

Para otorgar mayor flexibilidad al método es posible implementar estrategias adicionales de diversificación, si la solución no mejora durante determinado número de iteraciones. Una propuesta es adoptar la diversificación usada típicamente en el algoritmo de optimización por colonia de hormigas, en donde se genera una nueva población a la cual se le adiciona la mejor solución conocida hasta el momento.

Tabla 3. Cromosoma de solución

8	7	8	8	1	5	9	2	9	4	4	1	9	6	10	7	5	8	6	3	4	4	7	1	8	4	4	6	9	1	2	5	8	...		
4	2	6	9	3	3	9	1	4	9	1	3	8	4	6	8	1	9	5	6	4	4	1	2	6	8	8	3	4	2	10	10	3	...		
5	9	2	6	5	2	9	10	7	7	9	5	1	7	2	9	3	1	2	6	1	4	10	6	1	2	6	3	5	8	2	1	5	...		
3	6	7	8	3	4	2	4	9	6	5	3	9	10	5	6	6	6	1	9	9	4	10	9	4	4	6	2	4	2	4	10	5	...		
2	4	7	3	3	7	2	2	4	7	2	3	4	10	7	5	10	7	10	6	10	3	9	2	10	7	10	5	3	9	2	10	3	...		
1	9	7	1	5	5	9	8	8	3	5	7	8	1	6	6	1	8	7	5	5	7	6	7	9	5	3	5	1	2	7	1	4	...		
9	7																																		

AGRADECIMIENTOS

Los autores expresan su agradecimiento a la Universidad Distrital Francisco José de Caldas por facilitar los medios para esta publicación, y a la Universidad Tecnológica de Pereira (U.T.P.), por el apoyo brindado al grupo de desarrollo en investigación operativa (DINOP) de la U.T.P.

Referencias bibliográficas

[1] AARTS E.H.L and VAN LAARHOVEN P.J.M. (1985). *A New Polynomial Time Cooling schedule*, Proc. IEEE Int. Conf. On Computer-Aided Design, Santa Clara.

[2] BACK T. (1996). *Evolutionary Algorithms in Theory and Practice*. Oxford University Press.

[3] BEASLEY, J.E. E CHU, P. C. (1995) *A Genetic Algorithm for the Generalized Assignment Problem*. In: Computers Operations Research, 24(1), pp 17-23, 1997.

[4] CATTRYSSE D.G., VAN WASSENHOVE L.N. (1992) *A Survey of Algorithm for the Generalized Assignment Problem*. In: European Journal of Operational Research. 60, pp 260-272.

[5] CATTRYSSE D.G., VAN WASSENHOVE L.N., SALOMON M. (1994). *A Set Partitioning Heuristic for the Generalized Assignment Problem.* In: European Journal of Operational Research 72, pp 167-174.

[6] GOLDBERG D.E. (1989). *Genetic Algorithms in Search, Optimization an Machine Learning*. Addison Wesley, Reading, Mass.

[7] GRANADA M. y TORO E.M. (2004). *Solución al problema de la asignación generalizada usando el método de búsqueda tabú.* En: Revista Scientia et Technica (24), 61-67, U.T.P., Colombia.

[8] GRANADA M., TORO E.M., TABARES P. (2004). *Método de colonia de hormigas aplicado a la solución del problema de asignación generalizada*. En: Revista Tecnura No. 15, Universidad Distrital F.J.C., Colombia.

[9] HILLIER F., LIEBERMAN G.J. (1990). *Introduction to Operations Research*, McGraw Hill, Publishing Company.

[10] KELLY J.P., LAGUNA M. and GLOVER F. (1994). *A Study on Diversification Strategies for the Quadratic Assignment Problem.* In: Computers and Operations Research, vol. 22, no. 8, pp. 885-893.

[11] LAGUNA M., MARTÍ R. and CAMPOS V. (1999). *Intensification and Diversification with Elite Tabú Search Solutions for the Linear Ordering Problem*. In: Computers and Operations Research, vol. 26, pp. 1217-1230.

[12] MICHALEWICZ Z. (1992). *Genetic Algorithms + Data Structures = Evolution Programs*. In: Artificial Intelligence.

[13] OSORIO M.A. and LAGUNA M. (2003). *Logic Cuts for Multilevel Generalized Assignment Problems.* In: European Journal of Operational Research, vol. 151, pp. 238-246.