UNICIENCIA 22
pp. 115-121
2008

IMPLEMENTACIÓN DE UN ALGORITMO GENÉTICO PARA LA ASIGNACIÓN DE AULAS EN UN CENTRO DE ESTUDIO

***Yadira Solano Sabatier*[1], *Miguel Calvo Marín*[2], *Leonardo Trejos Picado*[3]**
Escuela de Ciencias de la Computación e Informática, Facultad de Ingeniería,
Universidad de Costa Rica, San Pedro, Montes de Oca, Costa Rica
[1] Tel.: (506) 2207-4020 Corel: ysolano@ecci.ucr.ac.cr
[2] Corel: calvomarin@ice.co.cr
[3] Corel: ltrejos@gmail.com

RESUMEN

Los algoritmos genéticos han demostrado ser una herramienta muy eficiente para resolver problemas de optimización. Por otra parte, la asignación de aulas en cualquier centro educativo, en particular, aquellos centros que no disponen de gran cantidad de aulas para hacer frente a la demanda periódica de cursos, se convierte en un problema de optimización. En la Escuela de Ciencias de la Computación e Informática de la Universidad de Costa Rica, esta asignación se realiza semestre a semestre, en forma manual, por lo que se hace necesaria la asignación de personal dedicado sólo a esta labor por varios días. El presente artículo presenta una solución automatizada que no sólo reduce el tiempo de respuesta a unos cuantos segundos, sino encuentra una solución óptima en la mayoría de las pruebas realizadas. Además ofrece facilidades adicionales como flexibilidad a la hora de definir horarios, cursos y tipos de aulas, así como la capacidad de interactuar con el sistema para probar formas diversas de asignación de aulas dependiendo de los requisitos de cada curso.

Palabras claves: Algoritmos genéticos, problemas de optimización, asignación de clases, solución automática.

ABSTRACT

The genetic algorithm is a very efficient tool to solve optimization problems. On the other hand, the classroom assignation in any education center, particularly those that does not have enough quantity of classrooms for the course's demand converts it in an optimization problem. In the Department of Computer Science (Universidad de Costa Rica) this work is carried out manually every six months. Besides, at least two persons of the department are dedicated full time to this labor for one week or more. The present article describes an automatic solution that not only reduces the response time to seconds but it also finds an optimal solution in the majority of the cases. In addition gives flexibility in using the program when the information involved with classroom assignation has to be updated. The interface is simple an easy to use.

Keywords: Genetic algorithm, optimization problems, classroom assignation, automatic solution.

1. ALGORITMOS GENÉTICOS

Como su nombre lo indica se inspiran en la teoría de la evolución de las especies de Charles Darwin, donde los individuos de una población se cruzan, se reproducen y sobreviven los más aptos (Holland, 1992; Koza, 1999).

Desde el punto de vista computacional, se establece una analogía entre una población de individuos y un espacio de soluciones para un problema particular. A grandes rasgos, el algoritmo consiste en cruzar las soluciones, al igual que los individuos de una población, producir nuevas soluciones, evaluarlas y seleccionar las más aptas para cruzarlas de nuevo; de manera que, después de varias generaciones se encuentren las mejores

Recibido 10/3/2006 / Aceptado 15/5/2006
Revisado por María Adilia García Vargas

soluciones para dicho problema. De lo anterior, es fácil deducir que los algoritmos genéticos representan una herramienta muy atractiva para resolver problemas de optimización.

Estos algoritmos se clasifican dentro de los métodos de búsqueda ciega, sin embargo, no realizan una búsqueda exhaustiva dentro del espacio de soluciones (Forrest, 1993). Esto significa que la complejidad computacional es de orden polinomial. Aunque teóricamente podrían no encontrar una solución óptima, ni siquiera una buena, en la práctica han demostrado ser muy eficaces, baste una revisión bibliográfica rápida para ver la cantidad creciente de artículos publicados sobre este tema en diversas revistas científicas de renombre internacional (Glen y Payne, 1995; Altshuler y Linden, 1997; Burke y Newall, 1999; Zitzler *et al*., 1999; Hughes y Leyland, 2000; Beasley *et al*., 2001; Au *et al*., 2003).

2. PROBLEMA: ASIGNACIÓN DE AULAS

En general, la asignación de aulas a los cursos que se imparten en cualquier centro educativo puede plantearse de manera fácil como un problema de optimización. Es decir, la demanda de aulas normalmente supera la cantidad disponible de éstas. Por esta razón, al inicio de cada período lectivo se hace necesario invertir una importante cantidad de recursos de personal y de tiempo, con el fin de hacer una adecuada distribución de aulas, acorde con las necesidades existentes en cada centro. La herramienta que describimos en este artículo ha sido desarrollada para el caso particular de la Escuela de Ciencias de la Computación e Informática de la Universidad de Costa Rica, sin embargo, es muy importante aclarar que cualquier centro educativo que enfrente el problema de asignación de aulas puede beneficiarse de manera significativa con esta herramienta. Los resultados obtenidos son muy satisfactorios, a continuación puntualizamos los más relevantes.

1. Automatización de una tarea que hasta ahora se ha realizado de forma manual.
2. Reducción del tiempo necesario para encontrar una solución, actualmente el tiempo requerido es de varios días.
3. La solución obtenida es muy cercana a la óptima, sin dejar de mencionar que en varios casos se obtuvo la óptima.
4. La herramienta es muy sencilla de comprender y de utilizar.
5. La herramienta es muy versátil: con pequeñas y sencillas modificaciones permite ajustarse a cambios en la descripción original del problema.

En las secciones siguientes se describe la aplicación con mayor detalle.

3. DESCRIPCIÓN DE LA HERRAMIENTA: ASIGNADOR DE AULAS

A continuación se describen los componentes de la herramienta, denominada de ahora en adelante: asignador de aulas.

Como primer componente se tienen los datos de entrada, la información que aquí se brinda es indispensable, debe ser completa y correcta, ningún dato puede omitirse, pues no permitiría el adecuado funcionamiento de la herramienta. Inicialmente debe indicarse el número de aulas, la respectiva identificación y el tipo de cada aula, así como la cantidad de tipos distintos que componen este grupo.

Posteriormente se debe editar, dentro de un formato determinado, el catálogo de cada uno de los cursos a los que se le desea asignar un aula. Es importante aclarar que la edición de este catálogo sólo se realiza la primera vez, en usos posteriores sólo será necesario actualizarlo si fuera el caso.

Como segunda parte se tiene la ejecución del algoritmo genético, que con base en los datos suministrados, encuentra una solución utilizando las operaciones que más adelante se describirán.

Finalmente, se tienen los resultados del programa, los cuales brindan la configuración completa de los horarios para cada una de las aulas especificadas en la entrada de datos. Es decir, para cada aula se configura un horario que indica las horas en que los distintos cursos serán impartidos. Además, permite ver todos aquellos cursos del catálogo a los cuales no les fue posible asignarle un aula.

Cabe mencionar que en cada una de estas partes destacadas se encuentran inmersas ciertas características de la aplicación que permiten una interfaz amigable, que sirve de guía en la utilización de esta herramienta.

En ella es posible editar y guardar los datos de entradas, con el fin de mantener un registro, para que en futuras aplicaciones que se requiera tomarlo como base, sólo se realicen las modificaciones necesarias.

Asimismo se brinda la opción de guardar las soluciones finales que el algoritmo encuentra para visualizar sus resultados cuando sea necesario y como último detalle se permite que los resultados de la herramienta que se centran en la información de los horarios, puedan ser divulgados por medio de la impresión en papel.

Una vez hecha la descripción general de la aplicación, a continuación se hará un mayor énfasis en cuanto a la composición del algoritmo genético en sí, explicando los objetos y operaciones que son parte en su desarrollo.

3.1 Objetos

Individuo

Este es el objeto o clase principal del algoritmo, representa la información de una posible solución, ya que posee toda la configuración de los horarios de las aulas. El método de representación utilizado permite una mayor precisión y complejidad que el método comparativamente restringido de utilizar sólo números binarios (Fleming y Purshouse, 2002).

Su composición es la siguiente:

I. Un conjunto de cromosomas representado por un arreglo bidimensional de hileras llamado Cromosoma, cada fila almacena la información propia y detallada de un curso de la siguiente manera:

   1. La sigla del curso. Ej.: "CI - 201".
   2. El grupo del curso. Ej.: "01".
   3. El primer día de su horario. Ej.: "L" (Lunes).
   4. Las horas del primer día. Ej.: "07:00-8:50".
   5. El segundo día de su horario. Ej.: "J" (Jueves).
   6. Las horas del segundo día. Ej.: "07:00-8:50".
   7. El requisito de tipo de aula. Ej.: "laboratorio".
   8. El aula asignada.

II. Un campo numérico entero que indica la cantidad de cromosomas (cursos) del individuo.

III. Un campo numérico entero que indica la cantidad de choques que se presentan a la hora de crear el horario; un choque se da cuando no es posible asignarle el aula designada que contiene en su cromosoma.

Padres

Este objeto es un arreglo de individuos, que contiene los padres iniciales que se han generado al azar y empezarán a cruzarse cuando dé inicio el algoritmo.

Élite

Es un arreglo de individuos, que almacena los dos individuos más aptos que se van generando a través del algoritmo.

Aulas

Este objeto manipula toda la información referente a las aulas. Entre los campos que maneja se encuentran:

- Un arreglo de hileras bidimensional que almacena todos los horarios de las aulas a las que se les desea asignar cursos. Cada horario de un aula abarca 7 columnas (días) y 14 filas (horas).
- Otro arreglo bidimensional que guarda las aulas según el tipo.
- Un campo numérico que almacena la cantidad de aulas.
- Un campo numérico que indica la cantidad

de tipos de aulas.

3.2 Operadores básicos

Aptitud

La aptitud del individuo está basada en el número de choques de horario que presente a la hora de realizar los horarios.

La función de aptitud en la implementación de este algoritmo genético se basa en contar el número de choques presentes en la asignación de las aulas con respecto a la hora. Dicho en otras palabras, se da cuando dos cursos con una misma hora y tipo de requisito, fueron asignados a la misma aula y, por lo tanto, se genera un choque.

Elitismo

La función de elitismo se basa en escoger los dos mejores padres de cada generación, o sea, aquéllos que poseen la mayor aptitud, y por lo tanto, representan buenas soluciones al problema. Esta élite podría no sufrir alteraciones si la nueva generación no produce individuos con una aptitud mayor que la que poseen los individuos de la élite.

Cruce

La función de cruce es la que permite la creación de una nueva generación de individuos; en nuestro caso se implementó el criterio del *cruce básico*. De esta manera, el punto de cruce se establece en cuatro distintas formas:

1. Se toma la primera mitad del cromosoma del padre y se toma la segunda mitad del cromosoma madre para generar un nuevo hijo.
2. Se toma la primera mitad del cromosoma de la madre y se toma la segunda mitad del cromosoma del padre para generar un nuevo hijo.
3. Se divide el cromosoma en 4 partes y se le asigna la sección 1.3 del padre y 2.4 de la madre.
4. Se divide el cromosoma en 4 partes y se le asigna la sección 2.4 del padre y 1.3 de la madre.

Mutación

El objetivo de esta operación es brindar mayor diversidad en la población. Tal y como se recomienda, se establece una tasa de mutación alta al inicio del algoritmo y una baja al final para permitir su convergencia hacia una solución óptima. Se implementó de la siguiente forma:

1. Se estable la cantidad de mutaciones a realizar según una variable que controla la tasa de mutación.
2. Se selecciona un nuevo número de forma aleatoria que indicará el gen a modificar, es decir, se cambia la asignación del aula de dicho gen.
3. Se repite el proceso anterior tantas veces como lo indique el número escogido en el punto 1.

Mezcla

Se refiere a la creación de nuevas generaciones. Las operaciones aplicadas en este proceso son: cruce, mutación y aptitud.

Primero se crean los padres de forma aleatoria, éstos crean la primera generación mediante los cruces. A partir de ellos se seleccionan los individuos más aptos por medio de las operaciones Aptitud y Elitismo. La Mutación interviene de una manera gradual en la conformación de los individuos añadiendo una mayor diversidad al conjunto de posibles soluciones.

Aplicando este proceso durante un determinado número de iteraciones, el algoritmo converge a una solución, que aunque es óptima, no está libre de presentar choques en los horarios.

3.3 Resultados obtenidos

La fase de prueba de esta aplicación se centró en la solución de la asignación de aulas a los cursos que se imparten en la Escuela de Ciencias de Computación e Informática de la Facultad de Ingeniería, Universidad de Costa Rica. Los datos para estas pruebas fueron proporcionados por el personal administrativo de esta unidad académica.

Estos datos corresponden tanto a la cantidad y características de las aulas que se tienen disponi-

Figura 1. Ventana para la actualización de la información sobre aulas.

bles para impartir las lecciones, como al catálogo de los cursos. En este caso se trabajó con la información correspondiente a los cursos impartidos en el segundo período del 2003.

El conjunto de aulas consta de 17 unidades y se simularon en un conjunto de 6 tipos distintos de aulas, ya que algunas poseen la característica de ser laboratorios especializados para impartir lecciones de cursos determinados o están destinadas a ciertos tipos de cursos como los de servicio que no pertenecen al plan de bachillerato o licenciatura de la carrera de Ciencias de la Computación e Informática.

Además, el catálogo o lista de cursos ofrecidos durante el período consta de 128 unidades, aquí se encuentra la información de cada uno de los cursos, a saber, su identificación (sigla y grupo), el horario de sus lecciones y, por último, el factor con una mayor relevancia como lo es el tipo de aula que requiere.

La información pertinente a aulas y cursos se introduce en las ventanas respectivas según se muestra a continuación en las figuras 1 y 2.

Una vez finalizada la ejecución de la apli-

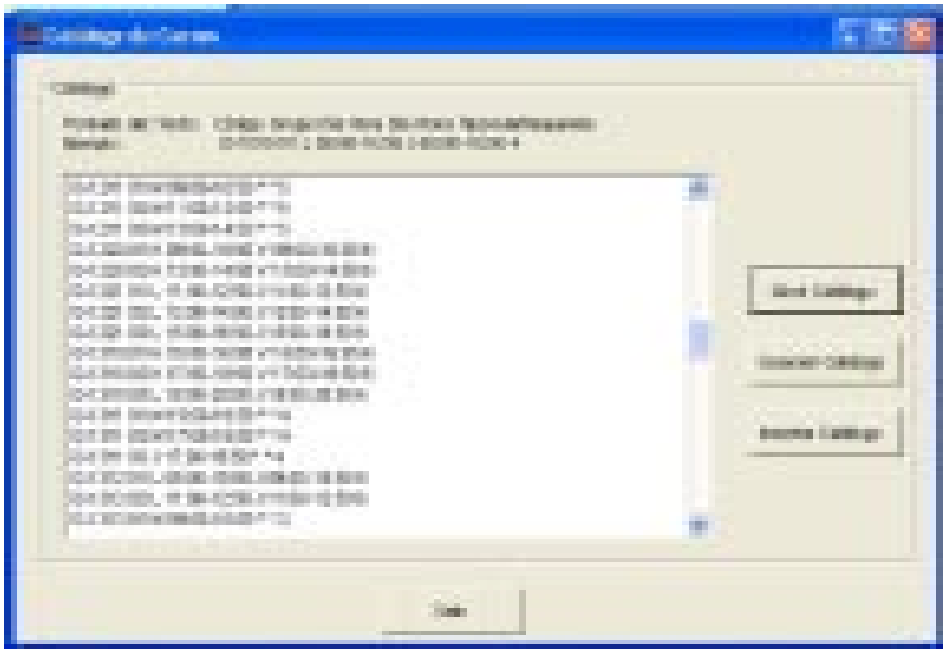

Figura 2. Ventana para la actualización de la información sobre cursos.

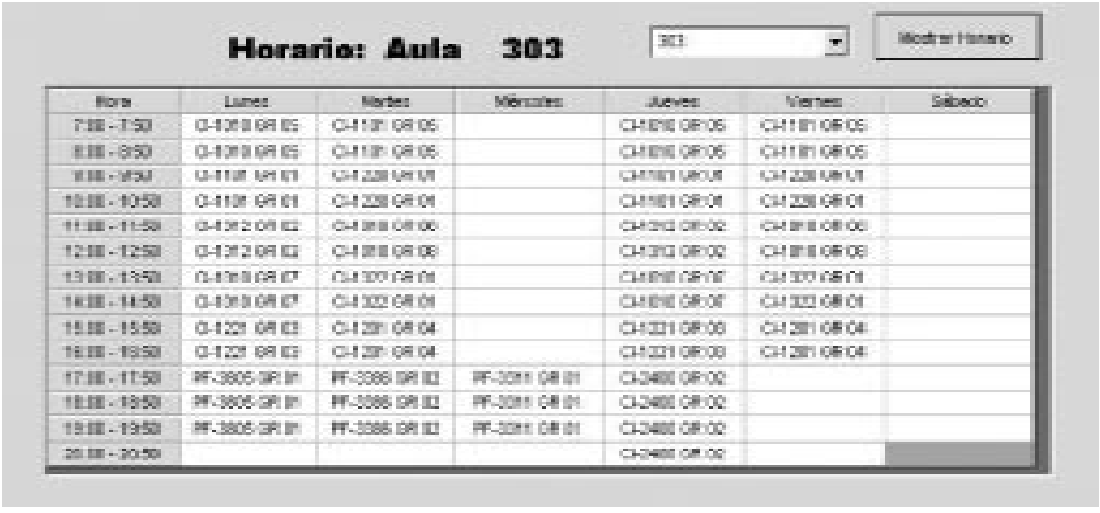

Figura 3. Ventana que muestra un ejemplo de asignación de horario para el aula 303.

cación fue posible analizar los resultados de una de las asignaciones óptimas encontradas.

En la figura 3 se observa un ejemplo del horario conformado para el aula 303 en particular.

Adicionalmente, en la figura 4 se muestran aquellos cursos a los que no fue posible asignarles lugar dentro de alguna aula, debido a la insuficiencia de aulas que satisfagan los requerimientos de dichos cursos.

Para el caso del ejemplo (Fig. 4), se nota que en varias ejecuciones se logró encontrar una solución que contabiliza solamente cuatro situaciones de choques, número muy bajo para la cantidad de cursos y aulas que la Escuela de Computación maneja. Estas soluciones se obtuvieron en un promedio de 4 segundos. El trabajo que resta es observar cómo quedan conformados los horarios de cada una de las aulas que participan en la asignación.

Dado que los cálculos de ciertos valores se hacen de forma aleatoria, el tiempo requerido para obtener una solución puede variar significativamente. De ahí la importancia de no quedarse con la primera solución, se recomienda guardarla y probar

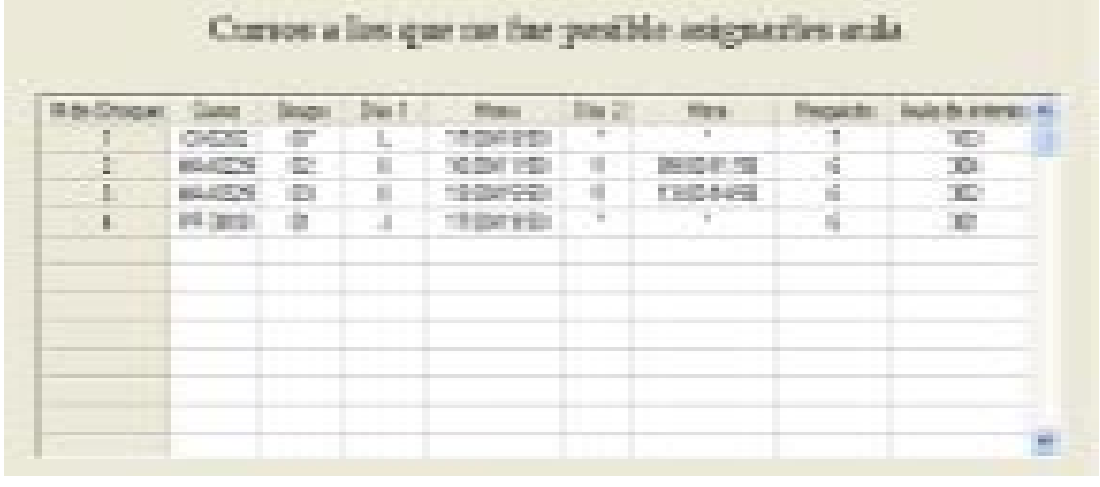

Figura 4. Ventana que muestra los casos fallidos o sin solución.

cierta cantidad de veces, con el fin de encontrar otras soluciones mejores en otros términos, en las que se puede llegar a obtener menor cantidad de cursos sin ser asignados.

4. CONCLUSIONES

Gracias al paralelismo implícito de los algoritmos genéticos no sólo es posible reducir el enorme número de soluciones que otros métodos obligan a evaluar, sino que también permite encontrar con éxito resultados óptimos o muy buenos en un período corto, tras muestrear directamente sólo regiones pequeñas del amplio espacio de búsqueda.

Los resultados obtenidos al aplicar algoritmos genéticos a un problema de optimización real, como lo es la asignación de aulas en un centro educativo, han sido muy satisfactorios como ha quedado demostrado en los resultados obtenidos.

El problema de la asignación de aulas está presente en toda institución de enseñanza y requiere de gran dedicación del personal asignado. Con el uso de la aplicación desarrollada se logra una gran reducción de tiempo, en este caso, se reduce la labor de una semana o más a tan sólo unos minutos, logrando una eficiencia en el uso de los recursos de personal y de las soluciones al problema planteado.

Los resultados de la figura 3 muestran que en la mayoría de los casos la solución obtenida se acerca a la óptima, a pesar de que en teoría puede no encontrarse ni siquiera una solución.

En una etapa posterior de este proyecto se piensa incluir otras variables igualmente restrictivas, entre ellas la asignación de profesores a cursos y horarios.

5. FUENTES BIBLIOGRÁFICAS

Altshuler, Edward & Derek Linden. 1997. "Design of a wire antenna using a genetic algorithm". *Journal of Electronic Defense*, vol. 20, nº 7, pp. 50-52.

Au, Wai-Ho, Keith Chan & Xin Yao. 2003. "A novel evolutionary data mining algorithm with applications to churn prediction". *IEEE Transactions on Evolutionary Computation*, vol. 7, nº 6, pp. 532-545.

Beasley, J.E., J. Sonander & P. Havelock. 2001. "Scheduling aircraft landings at London Heathrow using a population heuristic". *Journal of the Operational Research Society*, vol. 52, nº 5, pp. 483-493.

Burke, E.K. & J.P. Newall. 1999. "A multistage evolutionary algorithm for the timetable problem". *IEEE Transactions on Evolutionary Computation*, vol. 3, nº 1, pp. 63-74.

Fleming, Peter & R.C. Purshouse. 2002. "Evolutionary algorithms in control systems engineering: a survey". *Control Engineering Practice*, vol. 10, pp. 1223-1241.

Forrest, Stephanie. 1993. "Genetic algorithms: principles of natural selection applied to computation". *Science*, vol. 261, pp. 872-878.

Glen, R.C. y A.W.R. Payne. 1995. "A genetic algorithm for the automated generation of molecules within constraints". *Journal of Computer-Aided Molecular Design*, vol. 9, pp. 181-202.

Goldberg, David. 1989. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley.

Haupt, Randy & Sue Ellen Haupt. 1998. *Practical Genetic Algorithms*. John Wiley & Sons.

Holland, John. 1992. "Genetic algorithms". *Scientific American*. pp. 66-72.

Hughes, Evan & Maurice Leyland. 2000. "Using multiple genetic algorithms to generate radar point-scatterer models". *IEEE Transactions on Evolutionary Computation*, vol. 4, nº 2, pp. 147-163.

Koza, John, Forest Bennett, David Andre & Martin Keane. 1999. *Genetic Programming III: Darwinian Invention and Problem Solving*. Morgan Kaufmann Publishers.

Mitchell, Melanie. 1996. *An Introduction to Genetic Algorithms*. MIT Press.

Zitzler, Eckart & Lothar Thiele. 1999. "Multiobjective evolutionary algorithms: a comparative case study and the Strength Pareto approach". *IEEE Transactions on Evolutionary Computation*, vol. 3, nº 4, pp. 257-271.

6. AGRADECIMIENTOS

Deseamos expresar nuestro profundo agradeci-