Modelado de la Distribución de Espacios Físicos mediante Algoritmos Evolutivos

Paula A. Millado, Guillermo R. Coronado, M. Laura Ivanissevich
Universidad Nacional de la Patgonia Austral
Lisandro de la Torre 1070 - Te: 02966-442321
{pmillado; gcoronado; mivanissevich}@uarg.unpa.edu.ar

Claudio A. Delrieux
Universidad Nacional del Sur
claudio@acm.org

RESUMEN

En la unidad académica Río Gallegos de la Universidad Nacional de la Patagonia Austral, se realiza la distribución y asignación de cada cátedra a un aula, de forma manual, conformando así la grilla de horarios de la unidad. Son conocidas las frecuentes dificultades con las que se encuentra el personal a cargo al momento de realizar la tarea de reasignación de espacios ante eventos especiales, con resultados que, ante la premura, terminan desaprovechando el recurso edilicio que posee la citada unidad académica.
Nos hemos propuesto diseñar un modelo efectivo y flexible que realice esta tarea, minimizando la diferencia entre capacidad del aula y alumnos anotados por asignatura. Dicho modelo adopta las estrategias de un algoritmo evolutivo.

Palabras Claves: Algoritmos Evolutivos - Optimización.

1. Introducción

La asignación de la carga horaria de cada cátedra, a un aula (o espacio físico) disponible, es una tarea que se realiza manualmente en la Unidad Académica Río Gallegos de la UNPA. Las dificultades no se hacen esperar al momento de reasignar dichos espacios, por diversos motivos. Ni en las asignaciones originales ni en las reasignaciones, se tiene en cuenta la capacidad máxima de un espacio físico, desperdiciando este recurso. Esto sin mencionar los conflictos de asignaciones que sólo son detectados cuando llega el reclamo de los docentes al momento de descubrir la falla.

Nos hemos planteado el problema de obtener una distribución óptima de los espacios físicos disponibles minimizando (en esta primer etapa del proyecto) el número de bancos inutilizados. En comparación con otros trabajos realizados (ver [1] y [2]), en nuestro caso la tarea se ve simplificada por el hecho de disponer de todos los espacios físicos en un mismo

campus universitario, evitando así el problema de la movilidad de los alumnos. Así mismo todas las aulas disponen de las condiciones ambientales óptimas para su funcionamiento (calefacción, en nuestro caso).

Dentro de la amplia calificación de algoritmos evolutivos, nos hemos centrado en la modelización del problema haciendo uso de los Algoritmos Genéticos, sin descartar otros modelos para futuros enfoques.

2. Breve caracterización de un Algoritmo Genético

Un Algoritmo Genético (AG) consta de una representación esquemática a la que aplica una técnica de búsqueda (enfocada a problemas de optimización) basada en las teorías evolutivas Neodarwinianas. Es así que consta de procesos de selección de individuos más aptos de una población, que sobreviven al adaptarse, más eficientemente, a las exigencias del entorno. Este proceso se controla actuando sobre los genes de un individuo en los que está codificada cada una de sus características.

2.1. Bondades de un AG

Un AG difiere de los algoritmos tradicionales en los siguientes aspectos:

- Trabaja con la codificación de un conjunto de parámetros, no con los parámetros entre sí.
- Evalúa un grupo de soluciones en lugar de un solución por vez.
- Usa reglas de transiciones probabilísticas en lugar de reglas determinísticas.
- Evalúa las posibles soluciones sin aplicar ningún proceso de inferencia, salvando obstáculos vinculados a la definición de la función de supervivencia.

2.2. Representación del *individuo*

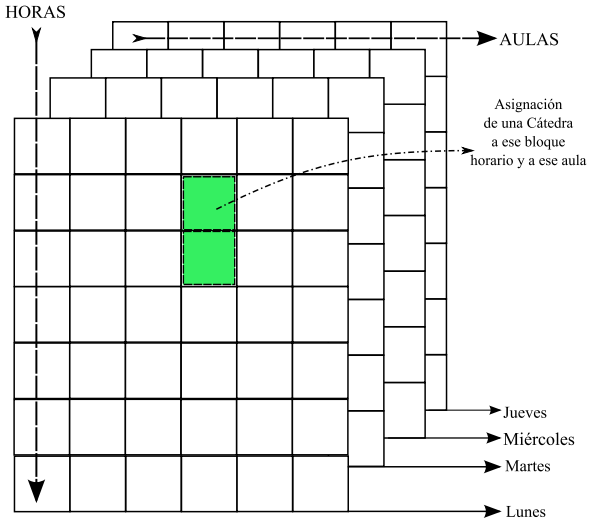
Un AG codifica las características del problema, representando los genes de cada individuo (o cromosoma) de la población de alguna manera. La forma tradicional de un AG (propuesta por John Holland) representa por ceros y unos los genes de cada cromosoma, aunque existen otros tipos de representaciones. La elección adecuada de dicha representación juega un papel importante en la eficacia del AG.

2.2.1. Nuestra elección de individuo

Hemos decidido modelar a nuestro *cromosoma* o *individuo* como una serie de matrices que representan la semana laboral, donde cada matriz conserva la información por dia, cada una de estas matrices está compuestas por filas que representan a las aulas y por columnas de horas, por lo que cada celda representa la asignación de un aula a una asignatura por una unidad de tiempo (generalmente una hora) debido a que las asignaturas pueden constar de 4,6,8,9 o 10 horas semanales en modulos de 2hs, 3hs o 4hs. Esta forma de ver al cromosoma evita la superposición de asignaciones.

2.3. La función *fitness* o función de aptitud

Representa el “ecosistema” donde el individuo vivirá; el que mejor se adapte, sobrevive. Matemáticamente hablando es la función-



Representación del indiviuo (cromosoma).

parámetro que evalúa una solución propuesta por el algoritmo.

2.3.1. Nuestro modelo de fitness

Estableciendo como premisa la intención de *minimizar* el espacio áulico desperdiciado, surge claramente que un primer intento de establecer la "regla de la vida" para los individuos de la población, se obtiene al monitorear la sumatoria de las diferencias entre el espacio físico del aula (en número de bancos) y el número de alumnos inscriptos a la cátedra asignada para esa aula. En síntesis:

$$\begin{aligned} Dif_p &= CapAula_i - AlCatedra_i \\ \Delta_p &= \Phi Dif_p \\ F_p &= \sum_{i=1}^{pob} (\Delta_p), \end{aligned}$$

El valor Φ representa una constante de penalización da tal manera que:

a) Si $Dif_p = 0$ tenemos la situación ideal, luego $\Phi = 1$.

b) Si $Dif_p < 0$ tenemos el caso donde la capacidad del aula es inferior al número de alumnos inscriptos en la cátedra (estado no deseado), luego se le asigna a Φ un valor lo suficientemente grande como para que el algoritmo le aplique una probabilidad baja en su proceso de supervivencia.

c) Si $Dif_p > 0$ tenemos el caso donde la capacidad del aula es superior al número de alumnos inscriptos. Esta es una situación deseada, sin embargo el algoritmo buscará la configuración que *minimice* este valor.

2.4. Generación de la población inicial o Birthpop

La primer etapa de un AG es la generación de la población inicial.

2.4.1. Nuestra *birthpop.*

Estamos evaluando dos alternativas, una de ellas es la generación clásica de una población inicial, es decir, totalmente aleatoria; como segunda opción se considera iniciar desde una asignación existente creada con una técnica tradicional (manual), buscando la optimización de dicha asignación. Para respetar la escencia de los algoritmos genéticos, la creacion de la poblacion inicial se realiza de forma aleatorea realizando un sorteo de las aulas entre las asignaturas a cubrir, si la celda sorteada se encuentra ocupada se sortea nuevamente.

2.5. Operadores de un AG

Un AG consta, escencialmente, de tres procesos conocidos como: *selección*, *crossover o cruzamiento* y *mutación*; los que, al aplicarse

garantizan la dinámica evolutiva que lo caracteriza.

2.5.1. Nuestro operador de selección

En nuestro modelo hemos elegido utilizar un proceso de selección basado en la rueda de la ruleta, es decir de la generación de hijos, se seleccionarán, probabilísticamente aquellos que mejor se ajustan a nuestra función *fitness*.

2.5.2. Nuestro operador de crossover

Este proceso consiste en entrecruzar subcadenas o partes del cromosoma (individuo) siguiendo algún método no determinístico de elección, tanto del número de subcadenas a intercambiar como del *lugar* del cromosoma donde se aplica la selección.
Se está evaluando la variante que mejor se adapte a nuestro modelo.

2.5.3. Nuestro operador de mutación

El operador mutación consiste en la alteración aleatoria de cada uno de los genes del individuo con cierta probabilidad. El objetivo de la mutación es el de producir diversidad en la población. Supongamos que al generar aleatoriamente la población inicial, en la misma posición de todos los cromosomas sólo aparece un único elemento del alfabeto utilizado, ésto supondrá que con los operadores reproducción y cruce nunca cambiará dicho elemento, por lo que puede ocurrir que jamás se alcance la solución óptima de nuestro problema. Es necesario un operador que pueda cambiar aleatoriamente cualquier elemento de cualquier individuo. Esta es la misión del operador mutación.

En nuestro caso, seduce la idea de intercambiar aquellos genes caracterizados por aulas de idéntica capacidad, aunque aún está en proceso de evaluación. Algunas ideas consisten en: a) la posiblilidad de utilizar Automatas Celulares a nivel del gen, aprovechando la estructura matricial pensada para el individuo. b) Realizar una mutación a nivel del cromosoma, implementado a traves del reordenamiento genetico mediante un intercambio de días.

3. Implementación

Se esta realizando las primeras pruebas de codificación en C++, debido a que posibilita el total control de los procesos y datos, siendo necesaria una amplia conciencia de estos. Se opto por tecnologías Orientadas a Objetos ante la posibilidad del uso de patrones de diseño como el Strategy del Catálogo de Patrones Gamma [5] que facilita la implementacion eficiente de familias de algoritmos, de forma encapsuladas e intercambiables, entre otros.

4. Referencias

[1] Karanik, Marcelo J. *Asignación DInámica de Aulas Utilizando Algoritmos Genéticos*. Trabajo presentado en VIII WICC. 2005.
[2] Karanik, Marcelo J. *Asignación DInámica de Aulas Utilizando Algoritmos Genéticos*. Trabajo presentado en IX WICC. 2006.
[3] Michalewicz, Z. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag, second edition, 1994.
[4] Rechenberg,I. *Evolution strategie: Optimierung technischer systeme nach prinzipien derbiologischen evolution*. Frommann-Holzboog, 1973.
[5] Gamma, Erich; Helm, Richard; Johnson, Ralph; Vlissides, John.*Design Patterns. Ele-*

ments of Reusable Object-Oriented Software. Editorial Addison-Wesley. 1995.

[6]Seed, Graham. *An Introduction to Object-Oriented Programming in C++*. Springer.1996

[7]Schildt, Herbert . *Borland C++. Manual de Referencia*. McGraw-Hill. 1997.

[8]Smith, A.E. and Tate, D.M. *Genetic Optimization Using a Penalty Function*. In Proceedings of the Fifth International Conference on Genetic Algorithms, 499-503. Urbana-Champaign, CA: Morgan Kaufmann. 1993.

[9]Burke, E.K. y J.P.Newall. *A multistage evolutionary algorithm for the timetable problem*. IEEE Transactions on Evolutionary Computation, vol.3, no.1, p.63-74 (abril de 1999).

[10] Goldbert, D. *Genetic Algorithms in Search Optimization and Machine Learning*. Adison-Wesley Publishing Co. 1989.