matemática: Una publicación del ICM – ESPOL
2010, Vol. 8, No. 1

UN ALGORITMO EVOLUTIVO PARA RESOLVER EL PROBLEMA DE CALENDARIZACIÓN PARA UNIVERSIDADES

[1]Cabezas Xavier, [2]Sandoya Fernando

Resumen. *En esta investigación se realizó el diseño y la implementación de un Algoritmo Evolutivo para resolver el problema de Calendarización de Materias Basado en Plan de Estudio (CB-CTT), el cual es un caso particular de la Calendarización Educacional para Universidades. Se consideran nueve restricciones, cuatro que no deben infringirse bajo ningún motivo (duras), como la asignación de dos recursos en un mismo lugar al mismo tiempo, y cinco cuyo no cumplimiento se obliga a minimizarse (suaves). Se diseña un Algoritmo Genético y se lo implementa en Matlab® con aportes en el diseño de la representación de una solución y en los procedimientos de Cruce, Mutación y Selección, obteniendo buenos resultados con los problemas de prueba propuesto por SaTT (Scheduling and Timetabling Group) de la Universidad de Udine, Italia.*

Palabras Claves: Calendarización, Metaheurísticas, Algoritmos Genéticos, Matlab.

Abstract. *In this research is designed and implemented an Evolutionary Algorithm to the Curriculum-Based Course Timetabling Problem (CB-CTT), which is a particular case of the Scheduling Education for Universities. Nine restrictions were considered, four have not infringed by any reason (hard), including the allocation of two resources in one place at the same time, and five that are required to be minimize (soft). A Genetic Algorithm is designed and implemented it in Matlab® with contributions into the design of the representation of a solution and in procedures for crossing, mutation and selection, obtaining good results with test problems proposed by SaTT (Scheduling and Timetabling Group), University of Udine, Italy.*

Keywords: Timetabling, Metaheuristics, Genetic Algorithm, Matlab.

Recibido: Febrero, 2010
Aceptado: Marzo, 2010

1. INTRODUCCIÓN

El Problema de Calendarización TTP (por sus siglas en inglés Timetabling Problem) es un tipo particular de la clase amplia de problemas simplemente de Programación (Scheduling) [5] y se puede definir de forma muy simple: **Definición 1.1** (Calendarización, Zhipeng Lu and Jin-Kao Hao [3]). Asignar un número de eventos, cada uno con ciertas características, a un número limitado de recursos sujeto a restricciones.

Para poder interpretar lo que en la definición 1.1 significan los términos: eventos, recursos y restricciones, se debe estudiar las características de un problema particular. Para este trabajo, se considera una clase de los TTP's denominada Educacional, que consiste en la programación de clases de un conjunto de materias con un número dado de aulas y periodos de tiempo disponibles que satisfacen varias restricciones [1]. Las restricciones pueden ser clasificadas en duras y suaves. Las restricciones de tipo duras son utilizadas para encontrar soluciones factibles al problema y no pueden ser infringidas, por otro lado las restricciones suaves pueden relajarse de tal forma de buscar soluciones cercanas al óptimo que serán cotas (inferiores o superiores) para buscar nuevas alternativas de solución.

El caso Educacional es solo una de las variaciones de los TTP's, también existen otras aplicaciones en áreas muy distintas, tales como transporte (programación de salidas de buses), TV (Calendario de Programas de TV) o deportes (programación de calendarios de juegos), un ejemplo de éste último tipo de problema se encuentra en [7], donde se muestra un método de solución que utiliza una aproximación con programación lineal entera (ILP, por las siglas en inglés de Integer Linear Programming) para la determinación de horarios de la Liga Italiana de Fútbol.

Para el caso Educacional se pueden clasificar en dos grupos: Calendarización para Exámenes (ETT, Exam Timetabling) y Calendarización para Materias (Clases Regulares) (CTT, Course Timetabling), este último a su vez puede ser subdividido en Calendarización por Materias Basado en Inscripciones (EB-CTT, Enrollment-Based Course Timetabling) y Calendarización por Materias Basado en Plan de Estudios (CB-CTT, Curriculum-Based Course Timetabling), ver Tabla I.

Cuando se intenta resolver el TTP, pueden presentarse una serie de conflictos entre las

[1]*Cabezas G. Xavier, MSc. en Investigación de Operaciones, Escuela Politécnica Nacional. Master en Control de Operaciones y Gestión Logística, ESPOL. (e_mail: xavier.cabezas@stat-eeio.com)*
[2]*Sandoya Fernando, M.Sc., Profesor de la Escuela Superior Politécnica del Litoral (ESPOL); Coordinador de la carrera Ingeniería en Logística y Transporte ICM – ESPOL. (e_mail: fsandoya@espol.edu.ec)*

restricciones que se plantean, pero éstas para el caso del EB-CTT se deberán a la decisión que tomen los estudiantes sobre las materias que desean tomar, un cruce podría ocurrir si la elección no ha sido buena; por otro lado los problemas que se podrán presentar en el CB-CTT resultarán de una programación no tan buena del plan de estudios por parte de la institución educativa. Los conflictos ocurren cuando una programación requiera que cualquiera de los recursos este en dos lugares al mismo tiempo. En realidad todo depende del tamaño de la institución o del departamento que ofrece sus cursos y del tiempo requerido para producir la solución.

TABLA I

Un algoritmo evolutivo para resolver el problema de calendarización para universidades

Clasificación de Problemas de Calendarización Educacional

Calendarización		
Educacional	Exámenes **ETT**	
	Materias **CTT**	Basado en Inscripciones **EB-CTT**
	Materias **CTT**	Basado en Plan de Estudios **CB-CTT**

2. RESTRICCIONES CONSIDERADAS PARA EL CB-CTT

Cuando se realiza una revisión de la literatura del CB-CTT se pueden encontrar una serie de documentos que presentan formulaciones novedosas pero que no siempre consideran las previamente definidas por otros autores, lo cual es debido a que no es posible escribir una que contenga todos los casos posibles que pueden presentarse en la vida real, cada institución educativa posee características diferentes que hacen que el problema se vuelva muy particular.

Las restricciones que se han considerado para esta investigación, son las mismas que se proponen en Di Gaspero [8].

Restricciones Duras (HARD constraints)

- **Sesiones de clases (lectures):** Todas las sesiones de clases de las materias deben programarse y asignarse a diferentes periodos. Si la sesión de clase no está programada, se considera una violación.
- **Aulas Ocupadas (room occupancy):** Dos lecturas no pueden ser programas en la misma aula en el mismo periodo. Se cuenta como una violación adicional cualquier sesión de clase extra en una misma aula y periodo.
- **Conflictos (conflicts):** Sesiones de clases de las materias del mismo curriculum o dictadas por un mismo profesor deben programarse en diferentes periodos. Una violación ocurre si dos sesiones de clases están en conflicto.
- **Disponibilidad (availabilities):** Si un profesor de una materia no está disponible para dictarla en un periodo dado, entonces ninguna sesión de clase de esta materia debe ubicarse en ese periodo. Cada sesión de clase en un periodo no disponible para esa materia se cuenta como una violación.

Restricciones Suaves (SOFT constraints)

- **Capacidad de Aulas (Room Capacity):** Para cada sesión de clase el número de estudiantes que asisten a las materias debe ser menor o igual al número de lugares disponibles de todas las aulas que acogen a las sesiones de clases. Se cuenta como una violación a cada alumno por encima de la capacidad del aula.
- **Días de Trabajo Mínimo (Minimum Working Days):** Las sesiones de clases de cada materia deben ser dictadas en un número mínimo de días. Cada día por debajo del mínimo se consideran 5 puntos de penalidad.
- **Curriculum Compacto (Curriculum Compactnees):** Sesiones de clase que pertenecen a un mismo curriculum deberán estar juntas unas con otras (periodos consecutivos). Para un curriculum dado se considera una violación cada vez que una sesión de clase no sea adyacente a otra en el mismo día.
- **Curriculum Compacto version 2 (Curriculum Compactnees ver 2):** Sesiones de clases programadas en un mismo día no deben tener sesiones de clases de otras materias entre ellas.
- **Estabilización de Aulas (Room Stability):** Todas las sesiones de clases de una materia deben ser dictadas en una sola aula.

2.1. UN EJEMPLO PARA RESOLVER

Unos datos de ejemplo que sirven de prueba para analizar la eficiencia de heurísticas y metaheurísticas para problemas de Calendarización (scheduling) y en particular para problemas de tipo CB-CTT se pueden encontrar en la página web del SaTT: *Scheduling and Timetabling Group*[3], aquí se define un formato de datos para inicio al que llaman ECTT (Extended Curriculum-based Course TimeTabling format), justamente para el CB-CTT, que contiene tablas con información específica que guardan relación con el formato requerido para la competición mundialmente conocida ITC

[3]Diegm-University of Udine (Italy), main contacts: Prof. Andrea Schaerf y Dr. Luca Di Gaspero; http://tabu.diegm.uniud.it/

(International Timetabling Competition).
De Cesco [1], donde los participantes deben remitirse a esta forma de ingreso, porque entre otras cosas, permite hacer ejecuciones de prueba para algunos ejemplos generados para este fin y compartir experiencia entre investigadores de forma estandarizada. Allí consta un pequeño ejemplo denominado Toy que tiene valor de la función de evaluación igual a cero. Toy se presenta en un archivo plano txt y se detalla en la Tabla II.

TABLA II
Un algoritmo evolutivo para resolver el problema de calendarización para universidades
Contenido del archivo toy.txt

```
Name: Toy
Courses: 4
Rooms: 3
Days: 5
Periods_per_day: 4
Curricula: 2
Constraints: 8

COURSES:
SceCosC Ocra 3 3 30
ArcTec Indaco 3 2 42
TecCos Rosa 5 4 40
Geotec Scarlatti 5 4 18

ROOMS:
rA 32
rB 50
rC 40

CURRICULA:
Cur1 3 SceCosC ArcTec TecCos
Cur2 2 TecCos Geotec

UNAVAILABILITY_CONSTRAINTS:
TecCos 2 0
TecCos 2 1
TecCos 3 2
TecCos 3 3
ArcTec 4 0
ArcTec 4 1
ArcTec 4 2
ArcTec 4 3

ROOM_CONSTRAINTS:
SceCosC rA
Geotec rB
TecCos rC

END.
```

Toy se resolverá con nuestro algoritmo. Se explica a continuación los componentes de este archivo txt.
Tabla: Datos generales
- Nombre para el problema a ejecutar.
- Número de aulas.
- Número de días para calendizar.
- Número de periodos por días.
- Número de curriculums.
- Min-Max clases diarias.
- Número de restricciones de no disponibilidad.
- Número de restricciones de aulas.

Tabla: Materias Nombre de identificación para las materias.
- Nombre de los profesores que dictan las materias.
- Número de clases de cada materia.
- Mínimo número de días para programar a las materias.
- Número de estudiantes que toman una materia. Valor binario 0-1 que indica si se requiere o no que dos lecturas programadas en el mismo día deban ser asignadas a periodos consecutivo (algo que ocurre en la mayoría de los casos).

Tabla: Aulas
- Nombre de identificación de las aulas.
- Capacidad de las aulas.
- Identificador que define la ubicación (en un bloque de edificio por ejemplo) de cada aula.

Tabla: Curriculums
- Nombre de identificación del curriculums (semestre).
- Número de cursos de los curriculums.
- Nombres de identificación de las materias en cada curriculum.

Tabla: Restricciones de no disponibilidad
- Nombre de identificación para las materias.
- Días en que las materias no pueden ser programadas.
- Periodos del día (Timeslots) en que las materias no pueden ser programadas.

Tabla: Restricciones de aulas
- Nombre de identificación para las materias.
- Nombre de identificación del aula donde no pueden ser programadas las materias.

El archivo "toy.txt" se llama directamente desde la implementación, y cada componente se guarda en matrices de números cuyos nombres se resumen en la Tabla III.

TABLA III
Un algoritmo evolutivo para resolver el problema de calendarización para universidades
Nombres y códigos de los archivos txt

Nombre de la tabla	Matrices
Datos generales	G
Materias (courses)	C
Aulas (rooms)	R
Curriculums (curricula)	CR
No disponibilidad (unavailability constrains)	UC
Restricciones de aulas (room constrains)	RC

3. DISEÑO DE UN ALGORITMO GENÉTICO PARA EL CB-CTT

3.1. EL ALGORITMO GENÉTICO

John Holland en su libro Adaptation in Natural and Aritificial Systems del año 1975 [4] fue el primero en utilizar el término Algoritmo Genético (al que abreviaremos AG), y que como su nombre lo indica, son procedimientos sistemáticos basados en la selección natural de los seres vivos y el paso de información genética generación a generación. Holland basó su trabajo en la evolución de las especies, propuesta por Darwin.

Definición 3.1 (Algoritmo Genético [2]). Un algoritmo Genético es una estructura de control que organiza o dirige un conjunto de transformaciones y operaciones diseñadas para simular los procesos de evolución.

Lógicamente los términos utilizados por los AG en sus implementaciones computacionales guardan relación con la evolución natural y por este motivo es necesario describir algunas características de este proceso, más aún si lo que se requiere es un proceso de evolución simulado, por medio de un algoritmo matemático implementado computacionalmente [2] (pág. 70).

Procesos de la evolución

Todos los cambios ocurridos a lo largo del cambio generacional ocurren en una unidad llamada cromosoma, que identifica a cada miembro de una población de individuos, de los cuales nos interesa la probabilidad que sus características genéticas sobrevivan en el futuro. La idea es que, al transcurrir el tiempo, en la población solo queden los individuos más fuertes, es decir aquellos que producen buenas soluciones al problema que se plantea. Los Genes constituyen la estructura más simple que forma un cromosoma, cada miembro de una población tiene el mismo número de genes. Los primeros algoritmos AG representaban a los genes por medio de valores binarios 1 - 0, sin embargo esto no es aplicable en la mayoría de los problemas prácticos.

La principal característica del proceso evolutivo es la supervivencia, la elección natural del más fuerte, lo que análogamente quiere decir en un problema de optimización un individuo que procure un mejor valor de una función objetivo (FO) a minimizar (o maximizar, de acuerdo al contexto de trabajo), es decir un individuo puede ser visto como una solución posible que puede mejorar o empeorar a FO. Esto lleva a considerar una función que permita medir que tan bueno o malo es un individuo respecto a otro. El cruce de los miembros de una población, producen hijos con características que heredan de sus padres, donde cada pareja se elige principalmente por su fortaleza dentro del grupo al que pertenecen. La búsqueda de la solución se consigue mediante la evaluación de la función objetivo f para cada cadena de la población. A la función de evaluación se la conoce con el nombre de fitness y el procedimiento requiere que la cadena de genes con mayor valor de fitness pueda ser identificado, para asignarle mayor probabilidad de reproducirse. La recombinación de soluciones es lo que hemos llamado cruce.

Otra situación que se vive en la naturaleza, es que de generación en generación siempre se producen cambios estructurales los cuales se esperan que sean para bien, sin embargo con alguna probabilidad un descendiente pudiera traer deformaciones genéticas que lo hagan o bien vulnerable al medio ambiente o bien lo conviertan en una entidad superior, a esto se lo suele llamar mutación.

Población Inicial

Antes de comenzar a buscar soluciones a un problema particular, es necesario contar con un conjunto de soluciones (Población Inicial) el cual pueda evolucionar generación tras generación. La Población Inicial puede ser construida de forma aleatoria o mediante algún procedimiento sistemático. Es muy importante considerar que este conjunto debe contener suficiente información para que su transformación procure una solución final óptima o al menos muy cerca del óptimo, una población muy homogénea podría tener como consecuencia una convergencia hacia una solución no necesariamente buena. Debería además tener un tamaño adecuado para evitar un bajo rendimiento del algoritmo, por otro lado un tamaño de la población muy grande podría llevarnos a un tiempo de proceso fuera de los límites de lo esperado.

Selección para una Nueva Generación

Cuando ya se tiene una población inicial diversa y de calidad se debe escoger los candidatos a quienes se aplicarán los procedimientos de cruce y/o mutación, es claro que se requiere que estas soluciones sean individuos fuertes, lo que simula la elección natural de las especies. Existen varios métodos para seleccionar soluciones entre los que se encuentran dos de las más conocidas: Torneo y Rueda de la Fortuna (Roulette Wheel).

En el método de selección por Torneo se eligen aleatoriamente desde la población dos individuos (soluciones) y de estos dos se selecciona aquel que al evaluarlo en la función objetivo obtenga el mejor resultado deseado (maximiza o minimiza).

En la Rueda de la Fortuna los elementos de la nueva generación se obtienen por medio de la generación de una variable aleatoria discreta que se define de la siguiente forma:

1. Se ordena la Población Actual de mayor a menor de acuerdo a la valoración de cada cromosoma en relación a la función objetivo.
2. Se suman las evaluaciones de la función objetiva de todos los cromosomas.
3. A cada cromosoma se le asigna una probabilidad de ser seleccionada de forma aleatoria, con:

$$Evaluación = \frac{Evaluación(i)}{\sum Evaluación(Población)}$$

Se selecciona un padre y una madre según el peso dado por la función de evaluación y la tabla de frecuencias relativas que se construye.

3.1.1. CRITERIO DE PARADA

Además de los procesos de cruce y mutación que ya se han explicado, es necesario contar con algún criterio de parada del algoritmo. Es claro, que mientras avanza el tiempo de ejecución del AG las nuevas generaciones serán cada vez más homogéneas, por este motivo un criterio natural de parada es detener el procedimiento cuando un gran número de cromosomas de la población sean iguales. Otro criterio podría ser definir un tiempo de ejecución fijo y luego de la última iteración elegir aquella solución representada por un cromosoma con mejor función de evaluación (fitness).
Un seudocódigo clásico y simple del AG se muestra a continuación:

FIGURA 1
Un algoritmo evolutivo para resolver el problema de calendarización para universidades
Simple Pseudocódigo para el AG

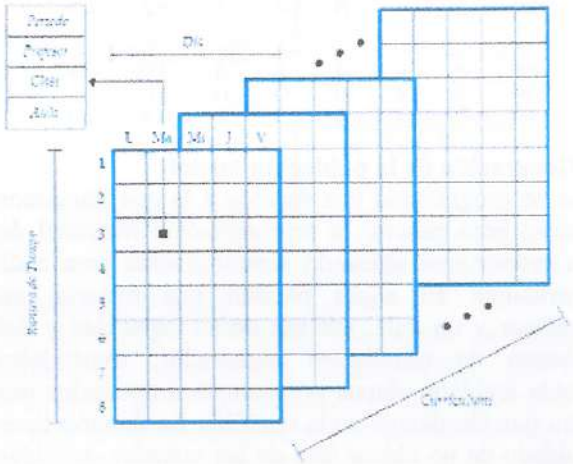
```
Procedimiento Algoritmo Genético
    Generar población inicial
    Evaluar población
While NO Criterio de Parada
    Selección_población
    Cruzar población
    Mutar población
    Evaluar población
end While
```

3.2. El AG EN EL CB-CTT Y SU IMPLEMENTACIÓN EN MATLAB ®

Representación del Cromosoma
Para el CB-CTT cada cromosoma, representa un calendario completo y en este caso cada gen es una estructura compuesta al menos de cinco componentes:
1. Día
2. Ranura de tiempo
3. Profesor
4. Clase (de alguna Materia)
5. Aula

El par (Día, Ranura de tiempo) forman un periodo. Un ejemplo se ilustra en la Figura 2.

FIGURA 2
Un algoritmo evolutivo para resolver el problema de calendarización para universidades
Representación de un cromosoma para Calendarización



Representación del Cromosoma en Matlab ®
Un calendario se ha representado en este trabajo como una matriz de dimensión [(días)(ranuras de tiempo por día)(curriculums) × 8]. En este arreglo, el número de filas corresponde a cada periodo disponible para calendarizar multiplicado por el número de curriculums considerados en un problema particular, esto quiere decir que los calendarios de cada curriculum se dispondrán uno debajo del otro, ver Figura 3, donde los diferentes colores rojo y azul determinan respectivamente un calendario para el curriculum 1 y curriculum 2.. Por otra parte, cada columna de la matriz representa respectivamente:
1. Indicador del número de la fila 1, 2, 3,. . ., etc.
2. Día de la semana (laborables).
3. Ranura de tiempo (horas de clases disponibles).
4. Materia que se dictará en un periodo.
5. Profesor de la materia.
6. Aula a utilizar.
7. Capacidad del aula a utilizar.
8. Número de estudiantes registrados en la materia.

FIGURA 3
Un algoritmo evolutivo para resolver el problema de calendarización para universidades
Representación matricial de un cromosoma (calendario)

Índice	Día	Ramura	Materia	Profesor	Aula	Capacidad	Estudiantes
1	1	1	3	3	2	50	40
2	2	1	0	0	0	0	0
3	3	1	1	1	3	40	30
:	:	:	:	:	:	:	:
19	4	4	0	0	0	0	2
20	5	4	2	2	3	40	42
21	1	1	0	0	0	0	0
22	2	1	3	3	1	32	40
:	:	:	:	:	:	:	:
38	3	4	4	4	3	40	18
39	4	4	0	0	0	0	0
40	5	4	3	3	1	32	40

Generación de la población inicial

Se ha programado una función a la que llamamos gcrom para generar la representación matricial de un cromosoma ubicando aleatoriamente para cada curriculum en algún periodo una materia, un profesor y un aula, además de su capacidad y del número de estudiantes registrados, tomándolas desde archivos planos previamente importados por otra función denominada readfiles. La función tiene cuidado de no ubicar más de las materias definidas para cada curriculum en el archivo CR.txt.

En este punto se aplica un procedimiento que trata de mejorar el cromosoma inicial cambiando aleatoriamente el aula antes asignada por otra, siempre y cuando la función objetivo mejore en su valor, para este caso esto quiere decir que la restricción suave llamada Capacidad de Aulas se supere. Este procedimiento se ha nombrado mejoraroom.

Evaluación de un Cromosoma y de la Población

Todas las restricciones consideradas en esta investigación se han programado dentro de la función fitnees, que mide el número de violaciones de cada restricción de acuerdo al criterio de la Sección 2. Un ejemplo de la aplicación de la función de evaluación en Matlab ® se presenta en la Figura 4. La suma del vector fitnessv es la medida de incumplimiento de restricciones.

FIGURA 4
Un algoritmo evolutivo para resolver el problema de calendarización para universidades
Ejemplo de aplicación de la función fitness

```
>> [ffitnessv ffitness]=fitness(cromosoma,G,R,UC,C)
ffitnessv =
     1     0     3    54     2     2     0     4
ffitness =
    66
```

La evaluación en la población se la realiza utilizando la función evaluapoblacion. Este procedimiento simplemente da el valor de fitness para cada cromosoma, presentándolos en un vector que se ha llamado fitnessp, ver Figura 5.

FIGURA 5
Un algoritmo evolutivo para resolver el problema de calendarización para universidades
Ejemplo del vector fitnessp para una población de tamaño 5

```
>> [fitnessp]=evaluapoblacion(poblacion,G,R,UC,C)
fitnessp =
    36
    68
    76
    71
    87
```

El proceso de selección

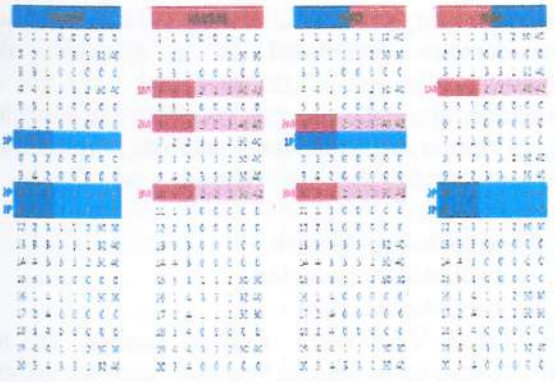
Para poder pasar de generación en generación se ha diseñado un procedimiento de selección de soluciones que permite elegir de entre todos los cromosomas de una población a los mejores, aquellos cuya función de evaluación sea la mejor posible. El método que se ha implementado es la Rueda de la Fortuna (Roulette Wheel).

El procedimiento de cruce

En la función cruce, se desarrolla un procedimiento sistemático, que transfiere información genética de un padre y una madre para generar un hijo y una hija.

El proceso comienza buscando en el calendario del padre y la madre el periodo donde cada una de las materias ya hayan sido programadas. Luego se genera un número aleatorio uniforme de parámetros 1 y 0 con el fin de determinar si es el gen del padre que se transferirá al hijo o el de la madre, y se ubicarán los datos del ascendiente en el mismo periodo encontrado, cuidando que esta ubicación esté libre en el descendiente, caso contrario se selecciona una ubicación libre de forma aleatoria. Cuando la elección se ha realizado, la información que se transfiere al otro descendiente es exactamente lo opuesto, siempre que sea posible. Todo esto se lo realiza para cada calendario de cada curriculum, ver Figura 6.

FIGURA 6
Un algoritmo evolutivo para resolver el problema de calendarización para universidades
Ejemplo de un proceso de cruce para la base de datos Toy



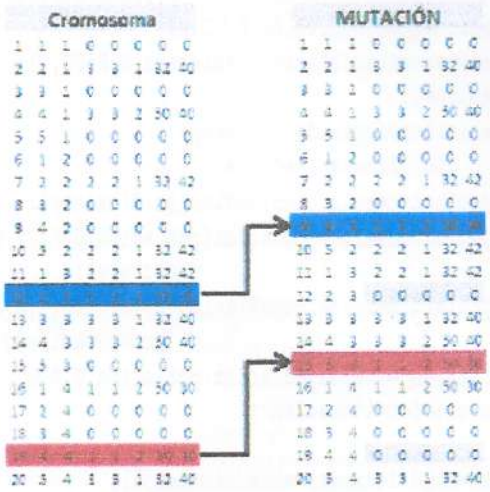
Con el propósito de asegurar que las futuras generaciones posean miembros más fuertes, se ha decidido que al momento de cruzar dos individuos, se elija entre el hijo y la hija, al de menor función de evaluación para que forme parte de la nueva población. La función fitness se aplica a ambos cromosomas y se verifica quien sobrevive y quién no. El padre y la madre se seleccionan de forma aleatoria desde la población luego del proceso de selección descrito anteriormente. La función que permite este procedimiento es crucepoblación. Esta función contiene el parámetro probcruce que representa la probabilidad de que dos individuos se crucen, y se ajusta luego de experimentos computacionales.

El procedimiento de mutación

La mutación es un procedimiento que actúa sobre los genes. En esta implementación se ha considerado para cada fila de la representación cromosómica no vacía, ubicar el gen (la fila) en un lugar disponible de forma aleatoria, pero en este caso, este cambio se produce siempre que la función de evaluación mejore en su medida. Esto asegura que si la mutación en un gen ocurre, esto sea para bien, ver Figura 7, donde el valor de la función de evaluación para Cromosoma es 97 y para MUTACIÓN es 93.

La función mutación produce estos cambios. Un parámetro de entrada es probmutacion el cual sirve para determinar si un gen es candidato a mutar. El proceso de mutación debe aplicarse a cada elemento de la población de acuerdo a una probabilidad que se define en la función mutaciónpoblación como probmutacion, esta es la misma probabilidad dada en la función mutación para decidir si un gen de un cromosoma mutará o no. Esto significa que la misma probabilidad servirá para definir si un cromosoma entra al proceso de mutación además de decidir si un gen cambiará de posición.

FIGURA 7
Un algoritmo evolutivo para resolver el problema de calendarización para universidades
Ejemplo de un proceso de mutación para la base de datos Toy.



Criterio de Parada

En teoría el AG podría ejecutarse en tiempo indefinido, teniendo en consideración que los elementos de la población luego de un tiempo t grande se parecerán entre sí y se espera además que la función de evaluación tenga un valor cercano a cero (o cero en el mejor de los casos). Debido a esto un criterio de parada del AG se ha definido en este trabajo como:

1. Para si al menos una proporción p de individuos miembros de la población tienen el mismo valor de fitness, ó
2. Si en alguna iteración algún elemento de la población tiene un valor de función de evaluación igual cero.

En el primer caso la solución será aquel cromosoma con mínimo fitness, si existe más de una solución encontrada, se elige una de forma aleatoria, y en el segundo caso aquel que tenga el valor de cero. En la función criterioparada se implementa el caso uno y en el caso dos se considera en el programa general del AG, ver Programa 1.

AG completo

Luego de tener todos los elementos necesarios para la implantación del AG se enlazan todas las rutinas descritas anteriormente en un solo código. A este programa general se denomina agcbctt, ver Programa 1. El cual guarda relación estricta con el código clásico de un AG, ver Figura 1.

Programa 1 "agcbctt.m": AG completo

```
function [poblacion tt]=agcbctt(G,C,R,CR,UC,RC,n,
                                probcruce,probmutacion)
poblacion=gpob(G,C,CR,RC,R,UC,n);
fitnessp=evaluapoblacion(poblacion,G,R,UC,C);
iteracion=0;
while (criterioparada(fitnessp) ||
       min(fitnessp)==0) ~=1
    poblacion=seleccion(poblacion,fitnessp);
    poblacion=crucepoblacion(poblacion,G,R,UC,C,
                             probcruce);
    poblacion=mutacionpoblacion(poblacion,G,R,UC,C,
                                probmutacion);
    fitnessp=evaluapoblacion(poblacion,G,R,UC,C);
    iteracion=iteracion+1
end
posmin=find(fitnessp==min(fitnessp));
tt=poblacion(:,:,posmin);
```

4. RESULTADOS NUMÉRICOS

4.1. RECURSOS COMPUTACIONALES EMPLEADOS

En Rodríguez [6] se define a Matlab ®[4] como un instrumento computacional simple, versátil y de gran poder para aplicaciones numéricas, simbólicas y gráficas, que contiene una gran cantidad de funciones predefinidas para aplicaciones en ciencias e ingeniería. Esta descripción de Matlab muestra muy bien las razones por la cual se ha elegido este lenguaje de programación para la implementación computacional de un AG diseñado específicamente para el CB-CTT.

La versión de Matlab® que se ha utilizado es Matlab 7.6.0.324 (R2008a), y los recursos de hardware y software se detallan en la Tabla IV.

TABLA IV

Un algoritmo evolutivo para resolver el problema de calendarización para universidades

Recursos para implementación computacional del AG

Sistema Operativo	Windows® XP Professional
Procesador	Intel® Core™2 Duo 1.06 GHz
Memoria RAM	2GB

[4] *Matlab® es una marca registrada de The MathWorks TM.*

4.2. CALIBRACIÓN DE PARÁMETROS

Los valores de los parámetros de entrada del AG se han ajustado mediante un proceso de ensayo y error, es decir, se probó diferentes valores hasta que consideramos estabilidad en la ejecución del algoritmo. Con estabilidad se quiere decir que el programa ejecutado alcanza los objetivos deseados en un tiempo razonablemente pequeño. Un resumen de los parámetros de entrada se detalla a continuación:

1. Tamaño de la población.
2. Probabilidad de mutación.
3. Probabilidad de cruce.

La Tabla V muestra valores medios de las repeticiones de la función "agcbctt" para los valores de n = 10,20, ... ,100. En la Figura 8 se puede observar que cuando se fijan los valores de probabilidad de *cruce* y *mutación* el AG parece encontrar soluciones óptimas. El número de iteraciones que se necesitan para que el AG encuentre una solución (no necesariamente la óptima) de acuerdo a la Figura 9 se vuelve estable, entre 22 y 26, para valores de n igual a 30, pero como se menciona antes, los *cromosomas* óptimos se empiezan a encontrar a partir de n = 50.

La Figura 10 da una tendencia lineal de crecimiento mientras el tamaño de la *población* crece. Las pruebas para este ejemplo han mostrado que tiempos aceptables del algoritmo podrían estar entre 30 y 50 segundos, que en el gráfico se muestran para n entre 40 y 60. Los valores fijos de las probabilidades de *cruce* y *mutación*, se obtuvieron luego de analizar de forma conjunta los gráficos para variaciones en estos parámetros. En la Tabla 6 se ha hecho variar la probabilidad de cruce en valores de probcruce = 0,1, 0,2, . . . , 1, y se puede observar en la Figura 11, que el óptimo se encuentra a partir del valor de probabilidad de cruce de 0,5, sin embargo, un análisis mas detallado mostró que el valor de este parámetro podría estar entre 0,6 y 0,8. Para estas probabilidades de cruce, el promedio del número de iteraciones que se necesitaron, está entre 23 y 27, ver Figura 12 y su tiempo de ejecución promedio entre 40 y 50 segundos como lo presenta la Figura 13.

TABLA V

Un algoritmo evolutivo para resolver el problema de calendarización para universidades

Experimento computacional (Tamaño de Población)

Parámetros	
Corridas c/pob	30
Prob. cruce	0.8
Prob. mutacion	0.4
% de igual fitness	0.8

n	Fitness	Iteraciones	Segundos
10	10	15	4
20	3	20	11
30	1	26	21
40	1	25	27
50	0	24	33
60	0	26	45
70	0	23	45
80	0	23	51
90	0	25	62
100	0	27	74

FIGURA 8

Un algoritmo evolutivo para resolver el problema de calendarización para universidades

Tamaño de Población vs Fitness, variación de población

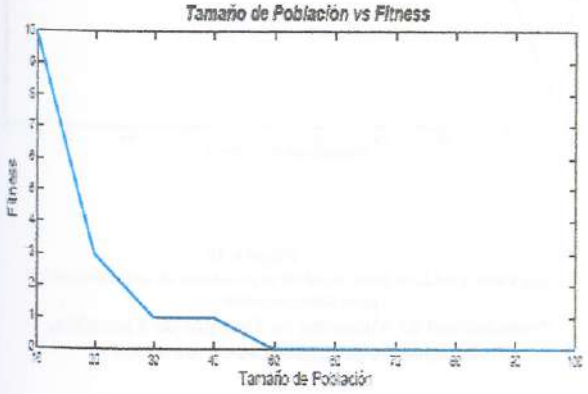


FIGURA 9

Un algoritmo evolutivo para resolver el problema de calendarización para universidades

Tamaño de Población vs Número de Iteraciones, variación de población

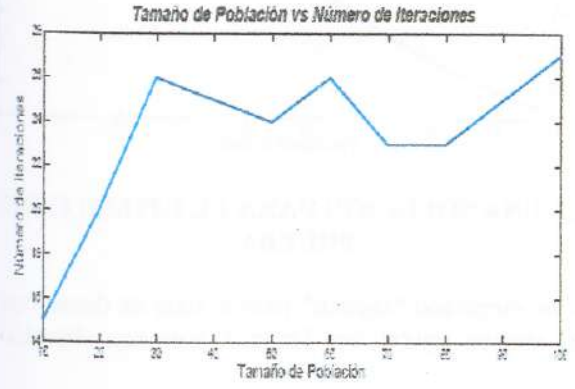


FIGURA 10

Un algoritmo evolutivo para resolver el problema de calendarización para universidades

Tamaño de Población vs Tiempo de Ejecución, variación de población

TABLA VI

Un algoritmo evolutivo para resolver el problema de calendarización para universidades

Experimento computacional (Probabilidad de Cruce)

Parámetros	
Corridas c/pob	30
n	60
Prob. mutación	0.4
% de igual fitness	0.8

probcruce	Fitness	Iteraciones	Segundos
0.1	8	23	16
0.2	3	26	21
0.3	1	27	27
0.4	1	26	31
0.5	0	26	36
0.6	0	27	46
0.7	0	24	45
0.8	0	23	47
0.9	0	24	56
1	0	24	62

FIGURA 11

Un algoritmo evolutivo para resolver el problema de calendarización para universidades

Probabilidad de Cruce vs Fitness, variación de probabilidad de cruce

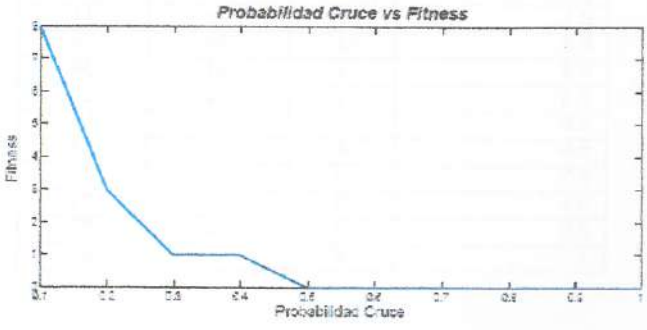


FIGURA 12
Un algoritmo evolutivo para resolver el problema de calendarización para universidades
Probabilidad de Cruce vs Número de Iteraciones, variación de probabilidad de cruce

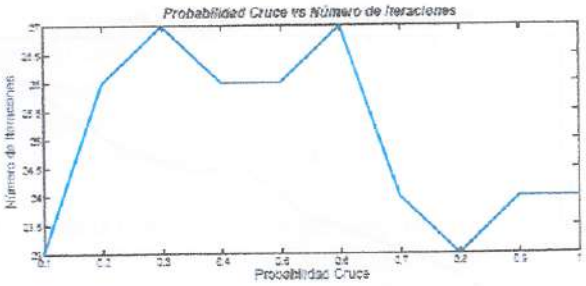
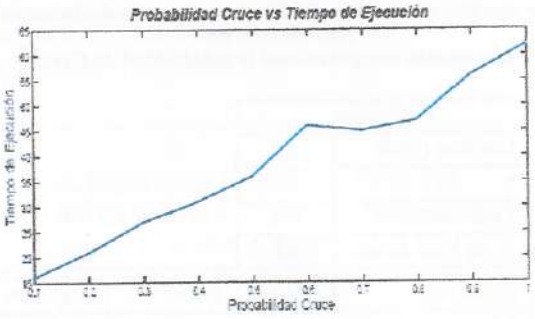


FIGURA 13
Un algoritmo evolutivo para resolver el problema de calendarización para universidades
Probabilidad de Cruce vs Tiempos de Ejecución, variación de probabilidad de cruce



Un análisis similar a los dos anteriores se realiza en la Tabla VII y en las Figuras 14, 15 y 16, donde se establecen valores de probabilidad de mutación mayores de 0,5 para un número de iteraciones promedio entre 0,3 y 0,7, y un tiempo de ejecución aproximado de 50 segundos.

TABLA VII
Un algoritmo evolutivo para resolver el problema de calendarización para universidades
Experimento computacional (Probabilidad de Mutación)

Parámetros	
Corridas c/pob	30
n	60
Prob. cruce	0.8
% de igual fitness	0.8

probmutacion	Fitness	Iteraciones	Segundos
0.1	7	19	11
0.2	4	22	18
0.3	1	24	26
0.4	1	24	37
0.5	0	23	50
0.6	0	24	63
0.7	0	22	72
0.8	0	23	73
0.9	0	24	77
1	0	25	83

FIGURA 14
Un algoritmo evolutivo para resolver el problema de calendarización para universidades
Probabilidad de Mutación vs Fitness, variación de probabilidad de mutación

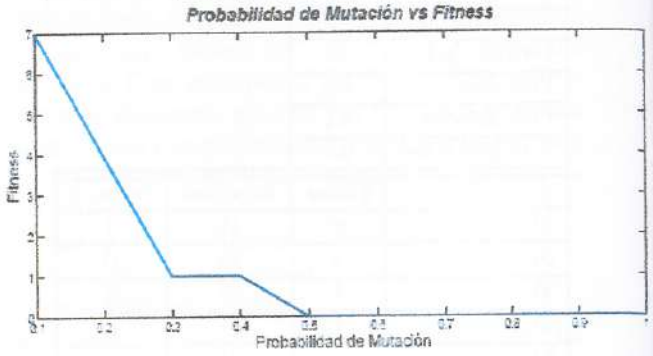


FIGURA 15
Un algoritmo evolutivo para resolver el problema de calendarización para universidades
Probabilidad de Mutación vs Número de Iteraciones, variación de probabilidad de mutación

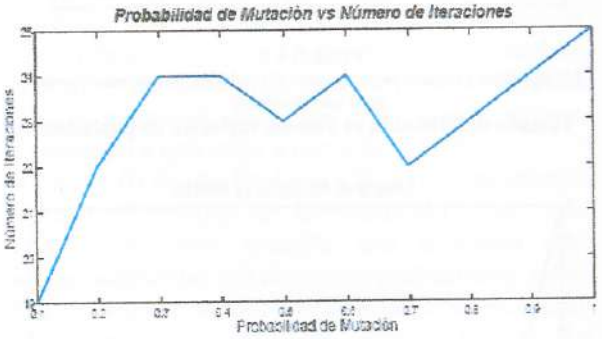
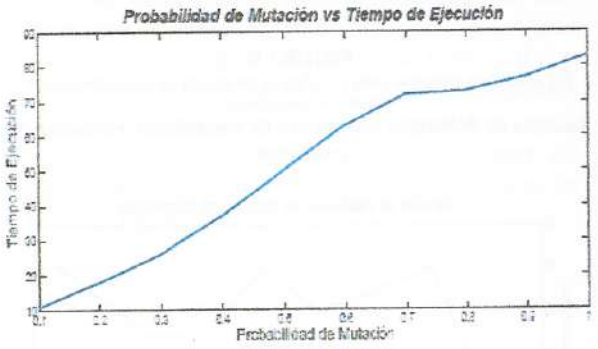


Figura 16
Un algoritmo evolutivo para resolver el problema de calendarización para universidades
Probabilidad de Mutación vs Tiempos de Ejecución, variación de probabilidad de mutación



4.3. UNA SOLUCIÓN PARA EL EJEMPLO DE PRUEBA

Se ha ejecutado "agcbctt" para la base de datos Toy del archivo *toy.txt*, ver Tabla 2, y se han obtenido

los calendarios de las Figuras 17 y 18 que cumplen con todas las restricciones impuestas.
Lo parámetros utilizados para este ejemplo fueron:

- Tamaño de la población: 60
- Probabilidad de Cruce: 0.7
- Probabilidad de Mutación: 0.4

FIGURA 17
Un algoritmo evolutivo para resolver el problema de calendarización para universidades
Calendario para el Curriculum 1, Toy

	Curriculum 1				
	Día 1	Día 2	Día 3	Día 4	Día 5
ranura 1	TecCos rB	ArcTec rB		TecCos rB	
ranura 2	TecCos rB			SceCosC rC	
ranura 3			SceCosC rC	ArcTec rB	TecCos rB
ranura 4			TecCos rB	ArcTec rB	SceCosC rC

FIGURA 18
Un algoritmo evolutivo para resolver el problema de calendarización para universidades
Calendario para el Curriculum 2, Toy

	Curriculum 2				
	Día 1	Día 2	Día 3	Día 4	Día 5
ranura 1					
ranura 2	Geotec rC	TecCos rB		TecCos rB	TecCos rB
ranura 3	TecCos rB	Geotec rC		Geotec rC	Geotec rC
ranura 4	TecCos rB			Geotec rC	

En la figura 19 se muestra la función de evaluación de la solución presentada.
El número de iteraciones que le tomó al AG encontrar una solución fue 22 y el tiempo de ejecución de 19 segundos.

FIGURA 19
Un algoritmo evolutivo para resolver el problema de calendarización para universidades
Aplicación de la función fitness a la base de datos Toy

```
>> [ffitnessv ffitness]=fitnessvector(tt,G,R,UC,C)
ffitnessv =
     0     0     0     0     0     0     0     0
ffitness =
     0
```

REFERENCIAS BIBLIOGRÁFICAS Y ELECTRÓNICAS

[1]. **DE CESCO, F.; SCHAERF, A.; DI GASPERO, L.** (2008). *"Benchmarking curriculum-based course timetabling: Formulations, data formats, instances, validation, and results"*.

[2]. **DÍAZ, A.** (1996). *"Optimización Heurística y Redes Neuronales"*. Addison Wesley.

[3]. **JIN-KAO HAO, ZHIPENG LU.** (2008). *"Adaptive tabú search for course timetabling"*. ELSEVIER.

[4]. **HOLLAND, J.H.** (1992), (1975). *"Adaptation in natural and artificial Systems"*. University of Michigan Press, Ann Arbor, Michigan; re-issued by MIT Press.

[5]. **Y-T LEUNG, J.** (2004). *"Handbook of Scheduling. Algorithms, Lodels and Perfomance Analysis"*. Chapman & Hall/CRC. Computer and Information Science Series.

[6]. **RODRÍGUEZ OJEDA, L.** (2007). *Matlab* R *Conceptos Básicos y Programación*. Instituto de Ciencias Matemáticas, Escuela Superior Politécnica del Litoral.

[7]. **OLIVERI, D.; DELLA CROCE, F.** (2004). *"Scheduling the italian football league: an ilp-based approach"*. ELSEVIER.

[8]. **SCHAERF, A.; DE CESCO, F.; MC- COLLUM, B.** (2007). *"The second international timetabling competition (itc-2007): Curriculum-based course timetabling (track 3)"*. Technical Report.