RISI 7(1), 37 - 55 (2010)
Revista de Investigación de Sistemas e Informática
Facultad de Ingeniería de Sistemas e Informática
Universidad Nacional Mayor de San Marcos
ISSN 1815-0268 (versión impresa)
ISSN 1816-3823 (versión electrónica)

Sistema de apoyo a la generación de horarios basado en algoritmos genéticos

Augusto Cortez Vásquez[1,2], Gissela Rosales Gerónimo[1], Raúl Naupari Quiroz[1], Hugo Vega Huerta Z.[1,2]

[1]Universidad Nacional Mayor de San Marcos
Facultad de Ingeniería de Sistemas e Informática
[2]Universidad Ricardo Palma
Facultad de Ingeniería

cortez_august@yahoo.fr, hugovegahuerta@hotmail.com

RESUMEN

El artículo presenta un estudio para solucionar el problema de elaboración de horarios o timetabling en una facultad. Propone un modelo matemático en el cual se definen las restricciones del problema, y luego se establece el diseño de la solución y la adaptación del algoritmo a esta. Para resolver el problema planteado utiliza algoritmos genéticos los cuales pertenecen al grupo de técnicas meta heurísticas. Con este desarrollo se pretende obtener un modelo cuyo principal objetivo es el de reducir el tiempo y la ocurrencia de errores en la generación manual de los horarios de clase por parte de las autoridades responsables de la Facultad de Ingeniería de Sistemas e Informática de la Universidad Nacional Mayor de San Marcos, y satisfacer al máximo las necesidades de los usuarios. Esto significa un avance tecnológico y organizativo cuyo propósito es el de minimizar los cruces entre cursos y horas, evitando extensos intervalos entre clases y disminuyendo los niveles de insatisfacción entre los docentes y estudiantes.

Palabras claves: Algoritmos genéticos, Timetabling, Optimización, asignación de carga académica.

ABSTRACT

The paper presents a study to solve the problem of developing a schedule or in a school timetabling. It proposes a mathematical model which defines the constraints of the problem, and then establishing the solution design and adaptation of this algorithm. To resolve the problem using genetic algorithms which belong to the meta heuristic techniques. With this development is to obtain a model whose main objective is to reduce the time and the occurrence of errors in the manual generation of school hours by the authorities of the Faculty of Engineering and Computer Systems National University Mayor de San Marcos, and to fully satisfy the needs of users. This means a technological and organizational whose purpose is to minimize the crossings between courses and hours, avoiding large gaps between classes and lowering the levels of dissatisfaction among teachers and students.

Key words: Genetic algorithms, Timetabling, Optimization, academic load allocation.

1. INTRODUCCIÓN

La asignación de recursos limitados, sujetos a restricciones de distinto tipo, ha sido un problema de relevancia en áreas como la administración de las organizaciones e incluso la inteligencia artificial. Por lo general, la solución de este tipo de problemas trae consigo una serie de condiciones de eficiencia, tiempo y oportunidad que deben ser tomadas en cuenta, más allá de la forma correcta de realizar la asignación. En particular se trata de resolver el problema de generación de horarios de clases, tomando como dominio del problema los cursos que se dictan en la Facultad de Ingeniería de Sistemas e Informática de la Universidad Nacional Mayor de San Marcos. El objetivo es maximizar el uso de los recursos, minimizar el desperdicio del espacio físico y establecer adecuadamente la carga académica de los docentes para lograr la asignación completa de los horarios de clases.

Aunque existen varios métodos de resolución como puede ser la búsqueda nodo por nodo de forma sistemática [10], o la utilizacion de métodos de búsqueda heurística [10], la propuesta que se hace en este trabajo es utilizar algoritmos genéticos.

Problemática

En toda institución de educación superior, al inicio de cada período académico se presenta la necesidad de asignar y coordinar los recursos económicos, materiales y humanos en beneficio de los estudiantes. Actualmente, como caso particular, en la Facultad de Ingeniería de Sistemas e Informática la tarea de elaboración de horarios académicos se realiza de forma manual, estimando y estipulando los recursos como mejor parecen ajustarse según el criterio del responsable de elaborar dichos horarios en cada semestre académico. Esta tarea normalmente requiere varios días de trabajo; sin embargo, debido al volumen de datos y variables que intervienen en el proceso, la tarea resulta ser costosa, compleja y no siempre culmina en resultados satisfactorios; como por ejemplo, realizar una asignación de aulas que permanecen medianamente vacías ya que la cantidad de alumnos matriculados es menor a su capacidad o viceversa, es decir, tal vez se asignan recintos muy pequeños para una gran cantidad de estudiantes, etc. Además se puede producir insatisfacción en algunos aspectos, tales como, que un estudiante no podrá matricularse en todos los cursos que desea debido a que están programados en el mismo periodo de tiempo, o que a un docente se le programan dos cursos en el mismo horario o periodo, etc.

Causas del problema

Para realizar la asignación de horarios, los encargados no cuentan con una planeación exacta de la información a usar, como

- Cantidad de estudiantes que llevarían un curso (alumnos que aprobaron el curso prerrequisito, mas los repitentes
- Cantidad de grupos que se debe aperturar por cada curso;
- Disponibilidad horaria por parte de los docentes
- Inventario de aulas de acuerdo a capacidades

2. MATERIALES Y MÉTODOS

Restricciones para realizar horarios de clases

Larrosa [7] define el término restricciones como condiciones que debe satisfacer el horario generado; y las clasifica en dos tipos:

Restricciones obligatorias, son propiedades espaciales o temporales. Toda restricción obligatoria debe cumplirse, la violación de alguna origina un horario no válido.

Restricciones deseables son restricciones que en realidad denotan preferencias del usuario (políticas flexibles) y se desea que se cumplan en la medida de lo posible. La violación de algunas de ellas seguirá produciendo un horario válido, pero de menor calidad que si se cumplieran todas éstas.

Adicionalmente clasificaremos cada uno de estos tipos en **Restricciones generales** las cuales podrán aplicarse a la mayoría de problemas de timetabling universitario y las **Restricciones específicas** que son propias de una sola institución, en este caso la FISI.

Para la solución de nuestro problema sólo se considerarán las restricciones obligatorias.

Restricciones Obligatorias

Restricciones Obligatorias Generales

- ROG1. Un curso debe tener asignado a lo más un docente en un aula en un periodo específico.

- ROG2. Un docente debe tener asignado a lo más un curso en un aula en un periodo específico.
- ROG3. Un aula puede tener a lo más un curso asignado y un docente en un periodo específico.
- ROG4. No es posible asignar a un docente un curso por el cual no posea preferencia.
- ROG5. No se puede asignar un curso a un docente fuera de su disponibilidad de tiempo.
- ROG6. La cantidad de estudiantes en un curso no puede sobrepasar la capacidad del aula.
- ROG7. Un curso debe cumplir con una cantidad de horas semanales requeridas. Según sea el caso, si el número de horas académicas es par se asignará dos horas diarias hasta cubrir la totalidad, sino se asignará un día de tres horas y el resto de dos horas. A cada uno de estos grupos de horas se les llamará bloques.
- ROG8. Las horas de un bloque para un determinado curso en un mismo día deben de ser consecutivas además de ser asignados a un docente y aula.
- ROG9. Cada bloque de un determinado curso sólo puede asignarse a lo más una sola vez cada día.
- ROG10. Los cursos no deben de solaparse entre ellos, ni en duración ni ubicación física, una vez seleccionada la hora de inicio de éstos y el aula correspondiente.

Restricciones Obligatorias Específicas

- ROE1. Un docente debe de impartir una cantidad determinada de horas de dictado de clases como máximo y como mínimo según su categoría.
- ROE2. Los cursos de teoría y práctica deben de asignarse a aulas de tipo no laboratorio.
- ROE3. Los cursos de laboratorio deben de asignarse a aulas de tipo laboratorio.
- ROE4. No deben coincidir los horarios de los cursos que corresponden a un mismo semestre y grupo.
- ROE5. Pueden existir asignaciones previas de cursos a un determinado periodo y aula.
- ROE6. Pueden existir asignaciones previas de docentes a un determinado período y curso.
- ROE7. Los cursos son clasificados en los turnos de mañana, tarde y noche según el semestre al que pertenecen.

Restricciones Deseables

Restricciones Deseables Generales

I. RDG1. Dos cursos de semestres consecutivos no deben de ser asignados en un mismo periodo de tiempo.

II. RDG2. Los bloques de un curso deben de distribuirse lo mejor posible en la semana.

III. RDG3. Se debe minimizar el número de horas libres entre el fin de una clase y el inicio de otra.

IV. RDG4. Pueden existir prioridades o penalidades sobre cursos para que sean dictados en periodos determinados. Es decir, una solución será de mayor calidad mientras se cumplan adecuadamente las prioridades o se tenga la menor cantidad de asignaciones a períodos penalizados.

Restricciones Deseables Específicas

V. RDE1. No se deben de asignar cursos en horario de almuerzo.

Modelo General

Conjunto de Datos

VI. Sea C = {1,..., c} $\forall$ c $\in$ Z el vector de cursos académicos.

Ejemplo: C = {

Algorítmica I	Cálculo I	…	Base de Datos	…

}

VII. Sea G = {g_1,...,g_n} el vector de grupos aperturados para los cursos. Cada g $\in$ G contiene un conjunto de cursos $C_g \subseteq C$, que no pueden ser programados en un mismo periodo de tiempo. Dos grupos distintos pueden tener cursos iguales, por ejemplo sea C_{g1} y C_{g2} dos cursos pertenecientes a los grupos G_1 y G_2, con $G_1 \neq G_2$ puede ocurrir que $C_{g1} \cap C_{g2} \neq 0$.

Ejemplo: G = {

Grupo 1	Grupo 2	Grupo 3

}

VIII. Sea L= {l_1,...,l_n} el vector de tipos que indica la modalidad del dictado de los cursos. Cada l $\in$ L contiene un conjunto de cursos $C_l \subseteq C$, que no pueden ser programadas en un mismo periodo de tiempo. Dos tipos distintos pueden tener cursos iguales, por ejemplo sea C_{l1} y C_{l2} dos cursos pertenecientes a los tipos L_1 y L_2, con $L_1 \neq L_2$ puede ocurrir que $C_{l1} \cap C_{l2} \neq 0$.

Definimos de antemano L= {l_t, l_p, l_l}, cada elemento correspondiente a los cursos de tipo de teoría, practica y laboratorio.

Ejemplo: L= {

Teoría	Práctica	Laboratorio

}

IX. Sea T = {1,..., t} $\forall$ w $\in$ Z el vector de periodos de tiempo en el que se puede dictar un curso cualquiera c $\in$C.

Ejemplo: T = {

Periodo 1 Lunes 8am – 9am	Periodo 2 Lunes 9am – 10am	…	Periodo 84 Sábado 9pm – 10pm

}

X. Sea I= {i_1,...,i_n} el vector de días de la semana que agrupa diferentes periodos. Cada i $\in$ I contiene un conjunto de periodos $T_i \subseteq$T. Dos días distintos no pueden tener periodos iguales, por ejemplo sea T_{i1} y T_{i2} dos periodos pertenecientes a los tipos I_1 y I_2, con $I_1 \neq I_2$ debe ocurrir que $T_{i1} \cap T_{i2} = 0$.

Definimos de antemano I= {i_l, i_m, i_e, i_j, i_v, i_s}, cada elemento correspondiente a los días lunes, martes, miércoles, jueves, viernes y sábado.

Ejemplo: I= {

Lunes	Martes	Miércoles	Jueves	Viernes	Sábado

}

XI. Sea D = {1,...,d} $\forall$ d $\in$ Z el vector de docentes que pueden dictar un curso cualquiera c $\in$C.

Ejemplo: D= {

Jaime Pariona Quispe	Daniel Quinto Pasce	…

}

XII. Sea A = {1,...,a} $\forall$ a $\in$ Z el vector de aulas donde se puede dictar un curso cualquiera c $\in$C.

Ejemplo: A= {

Aula 101	Aula 102	Aula 103	Aula 104	…

}

XIII. Sea DC la matriz de docentes d $\in$ D que dictan el curso c $\in$ C.

Ejemplo: DC= {

	Jaime Pariona	Gilberto Salinas Azaña	…
Algorítmica I	0	0	
…	0	0	
Base de Datos	1	0	
…	0	0	
Análisis de Sistemas	0	1	
…	0	0	

}

XIV. Sea TC la matriz de periodos de tiempo t $\in$ T en que se puede dictar el curso c $\in$ C.

Ejemplo: TC = {

	Algorítmica I	…	Base de Datos	…	Inteligencia Artificial	…
Periodo 1	1	…	0	…	0	…
Periodo 2	1	…	0	…	0	…
…	1	…	0	…	0	…
Periodo 6	0	…	1	…	0	…
Periodo 7	0	…	1	…	0	…
…	0	…	1	…	0	…
Periodo 11	0	…	0	…	1	…
Periodo 12	0	…	0	…	1	…
…	0	…	0	…	1	…
Periodo 15	1	…	0	…	0	…
…	…	…	…	…	...	…

}

XV. Sea TD la matriz de periodos de tiempo t $\in$ T correspondiente a la disponibilidad horaria del docente para dictar un curso c $\in$ C.

Ejemplo: TD = {

	Luis Alarcón Loayza	César Augusto Alcántara Loayza	…
Periodo 1	0	0	…
Periodo 2	0	0	…
…	0	0	…
Periodo 6	1	0	…
Periodo 7	1	0	…
…	1	0	…
Periodo 11	0	1	…
Periodo 12	0	1	…
…	0	1	…
Periodo 15	0	0	…
…	…	…	…

}

XVI. Sea K = {k_1,...,k_a} $\forall$ k $\in$ Z el vector de capacidades de las diferentes aulas.

Ejemplo: K= {

50 alumnos (Aula 101)	50	40	30	…

}

XVII. Sea Q = {q_1,...,q_c} $\forall$ q $\in$ Z el vector de cantidad máxima de alumnos por curso.

Ejemplo: Q= {

50 alumnos (Algorítmica I)	50	50	40	…

}

XVIII. Sea H = {h_1,...,h_c} $\forall$ h $\in$ Z el vector de cantidad de horas semanales que se le asigna a cada uno de los cursos.

Ejemplo: H= {

6 horas (Algorítmica I)	5	5	3	…

}

XIX. Sea MHD = $\{p_1,...,p_d\}$ $\forall$ p $\in$ Z el vector de cantidad máxima de horas que debe dictar semanalmente un docente d $\in$D.

Ejemplo: MHD = {

7 horas (Jaime Pariona Quispe)	8	6	4	…

}

XX. Sea NHD = $\{n_1,...,n_d\}$ $\forall$ n $\in$ Z el vector de cantidad mínima de horas que debe dictar semanalmente un docente d $\in$D.

Ejemplo: NHD = {

5 horas (Lucecita del Pino)	4	3	3	…

}

XXI. Sea S = $\{s_1,...,s_n\}$ el vector de semestres (o ciclos académicos) que está asociado a los diferentes cursos. Cada s $\in$ S contiene un conjunto de cursos Cs $\subseteq$C, que no pueden ser programadas en un mismo periodo de tiempo. Dos semestres distintos no pueden tener cursos iguales, por ejemplo sea C_{s1} y C_{s2} dos cursos pertenecientes a los semestres S_1 y S_2, con $S_1 \neq S_2$ debe ocurrir que $C_{s1} \cap C_{s2} = 0$.

Definimos de antemano S= $\{s_1,s_2,s_3,s_4,s_5,s_6,s_7,s_8,s_9,s_{10}\}$, cada elemento correspondiente a los cursos de los semestres del 1 al 10.

Ejemplo: S = {

Semestre I	Semestre II	Semestre III	…	Semestre X

}

XXII. Sea M = $\{m_1..,m_n\}$ el vector de tipos para indicar las clases de aulas existentes. Cada m $\in$ M contiene un conjunto de aulas $A_m \subseteq$A. Dos tipos distintos pueden tener aulas iguales, por ejemplo sea A_{m1} y A_{m2} dos aulas pertenecientes a los tipos M_1 y M_2, con $M_1 \neq M_2$ puede ocurrir que $A_{m1} \cap A_{m2} \neq 0$.

Definimos de antemano M=$\{m_t,m_p,m_l\}$, cada elemento correspondiente a las aulas de tipo teoría, practica y laboratorio.

Ejemplo: M = {

Teoría	Práctica	Laboratorio

}

XXIII. Sea PrvCD el vector de asignaciones previas de docentes d $\in$D que dictan el curso c $\in$C.

Ejemplo: PrvCD= {

	Luis Alarcón Loayza	Augusto Cortez Vasquez	…
Algorítmica I	0	0	…
…	0	0	…
ALgoritmica III	1	0	…
…	0	0	…
Análisis de Sistemas	0	1	…
…	0	0	…
Proyecto de Tesis I	0	1	…
…	…	…	…

}

XXIV. Sea PrvTC el vector de asignaciones previas de un curso c $\in$C que se dicta en el periodo t $\in$T.

Ejemplo: PrvTC = {

	Algorítmica I	…	Base de Datos	…	Inteligencia Artificial	…
Periodo 1	1	…	0	…	0	…
Periodo 2	0	…	0	…	0	…
…	0	…	0	…	0	…
Periodo 6	0	…	0	…	0	…
Periodo 7	0	…	0	…	0	…
…	0	…	0	…	0	…
Periodo 11	0	…	0	…	1	…
Periodo 12	0	…	1	…	0	…
…	0	…	0	…	0	…

}

XXV. Sea DTC el vector de periodos deseables t $\in$T donde un curso c $\in$C se pueda dictar.

Ejemplo: DTC = {

	Algorítmica I	…	Base de Datos	…	Estructura de datos	…
Periodo 1	1	…	0	…	0	…
Periodo 2	1	…	0	…	0	…
…	0	…	0	…	0	…
Periodo 20	0	…	1	…	0	…
Periodo 21	0	…	1	…	0	…
…	0	…	0	…	0	…
Periodo 39	0	…	0	…	1	…
Periodo 40	0	…	0	…	1	…
…	0	…	0	…	0	…

}

Parámetros

Definimos los siguientes parámetros que corresponden a cada uno de los recursos involucrados:

y = cantidad de periodos semanales

z = cantidad de aulas,

n = cantidad de cursos,

p = cantidad de docentes, $\forall_{y, z, n, p} \in Z$

Variables de decisión

Tenemos X_{ijkl} como variables binario-valoradas: 1, si asignan un aula i, al curso j, un docente k, en el periodo l.

X_{ijkl} = 1 si el aula i es asignada al curso j con el docente k en el periodo l.
0 si no.

$$\forall i \in \{1,..,z\}, \forall j, \in \{1,..,n\}, \forall k \in \{1,..,p\}, \forall l \in \{1,..,y\}$$

Variable de duración

Tenemos W_{ijkl} tiempo de ocupación del aula i por el curso j con el docente k en el periodo l.

$$R \in Z$$

$$W_{ijkl} =$$

$$\forall i \in \{1,..,z\}, \forall j, \in \{1,..,n\}, \forall k \in \{1,..,p\}, \forall l \in \{1,..,y\}$$

Modelamiento de Restricciones Obligatorias

Este modelamiento se dará en función de las restricciones obligatorias mencionadas anteriormente:

Todo curso j que se dicta en un periodo específico l debe tener asignado a lo más un docente k en un aula i.

$$\sum_{i=1}^{z}\sum_{k=1}^{p} X_{i\,jkl} \le 1$$

$$\forall j \in C, \forall l \in T \quad \text{... (ROG1)}$$

Todo docente k que enseña en un periodo específico l debe tener asignado a lo más un curso j en un aula i.

$$\sum_{i=1}^{z}\sum_{j=1}^{n} X_{i\,jkl} \le 1$$

$$\forall k \in D, \forall l \in T \quad \text{... (ROG2)}$$

Toda aula i en un periodo específico l debe tener asignado a lo más un curso j y un docente k.

$$\sum_{j=1}^{n}\sum_{k=1}^{p} X_{i\,jkl} \le 1$$

$$\forall i \in A, \forall l \in T \quad \text{... (ROG3)}$$

Todo docente k debe tener asignado un curso j que corresponda a su preferencia.

[DC = Matriz de docentes y cursos]

$$X_{i\,jkl} = DC_{jk}$$

$$\forall X_{i\,jkl} = 1, \forall i \in A, \forall l \in T, \forall j \in C, \forall k \in D$$

... (ROG4)

Todo docente k debe tener asignado un curso j dentro de su disponibilidad de tiempo.

[TD = Matriz de docentes y periodos de tiempo]

$$X_{i\,jkl} = TD_{jk}$$

$$\forall X_{i\,jkl} = 1, \forall i \in A, \forall l \in T, \forall j \in C, \forall k \in D$$

... (ROG5)

Todo curso j debe tener una cantidad de alumnos Q matriculados que no sobrepase la capacidad K del aula i.

$$X_{ijkl} <= 1 + (K_i - Q_j)$$

$$\forall X_{i\,jkl} = 1, \forall i \in A, \forall l \in T, \forall j \in C, \forall k \in D$$

... (ROG6)

Todo curso j debe cumplir con una cantidad de horas semanales H.

$$\sum_{i=1}^{z}\sum_{k=1}^{p}\sum_{l=1}^{y} X_{i\,jkl} * W_{i\,jkl} = H_j$$

$$\forall j \in C \quad \text{... (ROG7.a)}$$

Todo curso j cuya cantidad de horas académicas sea par se le asignará bloques de dos horas diarias hasta cubrir la totalidad, sino se le asignará un día de tres horas y el resto de dos horas.

$$W_{ijkl} = 2 + H_j \bmod 2$$

$$\forall i \in A, \forall l \in T, \forall j \in C, \forall k \in D \quad \text{... (ROG7.b)}$$

Todo curso j debe tener asignado un bloque de horas a lo más una vez dentro de los periodos [a,b] que pertenecen a un día de la semana l.

$$\sum_{i=1}^{z}\sum_{k=1}^{p}\sum_{l=a}^{b} X_{i\,jkl} \le 1$$

$$\forall j \in C, \forall a,b \in T \wedge \forall [a,b] \subseteq I \quad \text{... (ROG9)}$$

Todo curso j debe ser únicamente asignado en una aula i con un docente k desde que inicia en un periodo a hasta la finalización de su tiempo de duración W.

$$\sum_{l=a}^{a+W_{ijka}} X_{i\,jkl} \leq 1$$

$$\forall \mathrm{X}_{\mathrm{i\,jka}} = 1, \forall i \in A, \forall l,a \in T, \forall j \in C, \forall k \in D$$

... (ROG10)

Todo docente k tiene una cantidad mínima de horas NHD y una cantidad máxima de horas MHD de dictado de clases. [W = Vector de ocupación del aula i por el curso j con el docente k en el periodo l]

$$\sum_{i=1}^{z}\sum_{j=1}^{p}\sum_{l=1}^{y} X_{i\,jkl} * W_{i\,jkl} \leq MHD_k \wedge \sum_{i=1}^{z}\sum_{j=1}^{p}\sum_{l=1}^{y} X_{i\,jkl} * W_{i\,jkl} \geq NHD_k$$

$$\forall k \in D \text{ ... (ROE1)}$$

Todo curso j de tipo teoría Lt debe ser asignado a un aula i de tipo no laboratorio Mt.

$$\mathrm{L_{jT}} = \mathrm{M_{iT}}$$

$$\forall \mathrm{X_{i\,jkl}} = 1, \forall i \in A, \forall l \in T, \forall j \in C, \forall k \in D$$

... (ROE2.a)

Todo curso j de tipo práctica Lp debe ser asignado a un aula i de tipo no laboratorio Mp.

$$\mathrm{L_{jP}} = \mathrm{M_{iP}}$$

$$\forall \mathrm{X_{i\,jkl}} = 1, \forall i \in A, \forall l \in T, \forall j \in C, \forall k \in D$$

... (ROE2.b)

Todo curso j de tipo laboratorio Ll debe ser asignado a un aula i de tipo laboratorio Ml.

$$\mathrm{L_{jL}} = \mathrm{M_{iL}}$$

$$\forall \mathrm{X_{i\,jkl}} = 1, \forall i \in A, \forall l \in T, \forall j \in C, \forall k \in D$$

... (ROE3)

Todo curso j dentro de un rango con valor inicial a y un valor final b que pertenece a un semestre S y a un Grupo G debe ser asignado a un aula i y un docente k y no debe coincidir con otro rango de cursos de otro semestre y con ningún otro grupo.

$$\sum_{k=1}^{p}\sum_{j=a}^{b}\sum_{i=1}^{z} X_{i\,jkl} \leq 1$$

$$\forall l \in T, \forall a,b \in C \wedge \forall [a,b] \subseteq S \wedge \forall [a,b] \subseteq G$$

... (ROE4)

Existen asignaciones previas de un curso j a un determinado período l y un aula i.

$$\mathrm{PrvCD_{i\,jkl}} = \mathrm{X_{ijkl}}$$

$$\forall \mathrm{X_{i\,jkl}} = 1, \forall i \in A, \forall l \in T, \forall j \in C, \forall k \in D$$

... (ROE5)

Existen asignaciones previas de un docente k a un determinado período l y un curso j.

$$\mathrm{PrvTC_{i\,jkl}} = \mathrm{X_{ijkl}}$$

$$\forall \mathrm{X_{i\,jkl}} = 1, \forall i \in A, \forall l \in T, \forall j \in C, \forall k \in D$$

... (ROE6)

Los cursos son clasificados en los turnos de mañana, tarde y noche según el semestre al que pertenecen

$$\mathrm{X_{i\,jkl}} = \mathrm{TTC_{jl}}$$

$$\forall \mathrm{X_{i\,jkl}} = 1, \forall i \in A, \forall l \in T, \forall j \in C, \forall k \in D$$

... (ROE7)

Modelamiento de Restricciones Deseables

Este modelamiento se dará en función de las restricciones deseables mencionadas anteriormente:

Todo curso j que pertenece a un rango inicial a y rango final b no debe ser asignado en un mismo periodo l con respecto a otro rango inicial c y un rango final d que corresponden a semestres S consecutivos.

$$\sum_{k=1}^{p}\sum_{j=a}^{b}\sum_{i=1}^{z} X_{i\,jkl} + \sum_{k=1}^{p}\sum_{j=c}^{d}\sum_{i=1}^{z} X_{i\,jkl} \leq 1$$

$$\forall l \in T, \forall a,b,c,d \in C \wedge \forall [a,b],[c,d] \subseteq S \quad \text{... (RDG1)}$$

Los bloques de un curso deben de distribuirse adecuadamente durante la semana.

Vi ≤días que ocupa el curso en la semana / (ultimo día de la semana que ocupa el curso-primer día de la se-

mana usado para el curso) ≤ Vs $\forall j \in C$ *... (RDG2)*

Minimizar el número de horas libres entre el fin de una clase y el inicio de otra.

Bi ≤ Total de horas ocupadas por algún curso/ (ultima hora ocupada del día por algún curso - primera hora ocupada del día por algún curso) ≤ Bs $\forall i \in I$ *... (RDG3)*

Todo curso j debe ser dictado en un periodo l determinado.

$$\mathrm{D}TC_{\mathrm{jl}} = \mathrm{X}_{\mathrm{ijkl}} \quad \forall \mathrm{X}_{\mathrm{i\,jkl}} = 1 \quad \text{... (RDG4) y (RDE1)}$$

Función objetivo

Función objetivo global

$$fog(x) = \sum_{x=1}^{N} fol(x)$$

$$\forall \mathrm{x} \in = \{1,..,N\}$$

Donde x representa cada elemento de la matriz, N representa el número total de elementos en la matriz de cuatro dimensiones.

Función objetivo local

$$fol(x) = (\ chsl(x) + chsg(x)\ /2) * (\ chpl(x) + chpg(x)\ /2)$$

En donde, chsl y chpl representa los conflictos de aulas y docentes en un mismo semestre, respectivamente, mientras que chsg y chpg representan los conflictos de aulas y docentes entre semestres.

Criterio de optimización

$Min(fog(x)) = w$, donde w es una constante cualquiera.

Modelo Particular

Una vez definido el modelo general debemos adaptarlo a la FISI, dando como resultado un modelo particular el cual acota aún más el dominio del problema.

Conjunto de Datos

Se cuenta con un total de 68 cursos a programar.

Existen a lo máximo 3 grupos por cada curso a programar.

Los cursos a programar se pueden dictar en las diferentes modalidades de tipo teoría, practica y laboratorio

Existen un total de 84 periodos de tiempo posibles que van desde el lunes a las 8 am. hasta el sábado a las 10:00 pm. definiendo la duración por periodo de 60 minutos.

Los diferentes periodos de tiempo son contenidos en los días lunes, martes, miércoles, jueves, viernes y sábado.

Se cuenta con un total de una plana de 101 docentes.

Se cuenta con un total de 25 aulas hábiles para dictar las clases

Son 10 semestres (ciclos académicos) en total.

Existen tres clasificaciones diferentes de aulas, de teoría, práctica y laboratorio.

Parámetros

84 = cantidad de periodos semanales

25 = cantidad de aulas,

68 = cantidad de cursos,

101 = cantidad de docentes, $\forall_{\mathrm{x,y,n,p}} \in \mathrm{Z}$

Variables de decisión

Tenemos X_{ijkl} como variables binario-valoradas: 1, si asignan una aula i, al curso j, un docente k, en el periodo l.

1 si el aula i es asignada al curso j con el docente k en el periodo l.

X_{ijkl} = 0 si no.

$$\forall \mathrm{i} \in = \{1,..,25\}, \forall j, \in = \{1,..,68\}, \forall \mathrm{k} \in = \{1,..,101\}, \forall \mathrm{l} \in = \{1,..,84\}$$

Variable de duración

Tenemos W_{ijkl} tiempo de ocupación del aula i por el curso j con el docente k en el periodo l.

$$R \in Z$$

$$\mathrm{W}_{\mathrm{ijkl}} = \quad 0$$

$$\forall \mathrm{i} \in = \{1,..,25\}, \forall j, \in = \{1,..,68\}, \forall \mathrm{k} \in = \{1,..,101\}, \forall \mathrm{l} \in = \{1,..,84$$

Modelamiento de Restricciones Obligatorias

Según las restricciones obligatorias mencionadas anteriormente:

$$\sum_{i=1}^{25}\sum_{k=1}^{101} X_{i\,jkl} \le 1$$

$$\forall j \in C, \forall l \in T \text{ ... (ROG1)}$$

$$\sum_{i=1}^{25}\sum_{j=1}^{68} X_{i\,jkl} \le 1$$

$$\forall k \in D, \forall l \in T \text{ ... (ROG2)}$$

$$\sum_{j=1}^{68}\sum_{k=1}^{101} X_{i\,jkl} \le 1$$

$$\forall i \in A, \forall l \in T \text{ ... (ROG3)}$$

$$\forall \mathrm{X}_{\mathrm{i\,jkl}} = 1, \forall i \in A, \forall l \in T, \forall j \in C, \forall k \in D$$

... (ROG4)

$$\mathrm{X}_{\mathrm{i\,jkl}} = \mathrm{TD}_{\mathrm{jk}}$$

$$\forall \mathrm{X}_{\mathrm{i\,jkl}} = 1, \forall i \in A, \forall l \in T, \forall j \in C, \forall k \in D$$

... (ROG5)

$$\forall \mathrm{X}_{\mathrm{i\,jkl}} = 1, \forall i \in A, \forall l \in T, \forall j \in C, \forall k \in D$$

... (ROG6)

$$\sum_{i=1}^{25}\sum_{k=1}^{101}\sum_{l=1}^{84} X_{i\,jkl} * T_{i\,jkl} = H_j$$

$$\forall j \in C \text{ ... (ROG7.a)}$$

$$\mathrm{T}_{\mathrm{ijkl}} = 2 + \mathrm{H}_{\mathrm{j}} \mathrm{mod} 2$$

$$\forall i \in A, \forall l \in T, \forall j \in C, \forall k \in D \text{ ...}$$

(ROG7.b)

$$\forall j \in C, \forall a, b \in T \wedge \forall [a,b] \subseteq I \text{ ... (ROG9)}$$

$$\sum_{l=a}^{a+T_{jlka}} X_{i\,jkl} \le 1$$

$$\sum_{i=1}^{25}\sum_{k=1}^{101}\sum_{l=a}^{b} X_{i\,jkl} \le 1$$

$$\forall \mathrm{X}_{\mathrm{i\,jka}} = 1, \forall i \in A, \forall l, a \in T, \forall j \in C, \forall k \in D$$

... (ROG10)

$$\sum_{i=1}^{25}\sum_{j=1}^{68}\sum_{l=1}^{84} X_{i\,jkl} * W_{i\,jkl} \le MHD_k \wedge \sum_{i=1}^{25}\sum_{j=1}^{68}\sum_{l=1}^{84} X_{i\,jkl} * W_{i\,jkl} \ge NHD_k$$

$$\forall k \in D \text{ ...(ROE1)}$$

$$\mathrm{L}_{\mathrm{jT}} = \mathrm{M}_{\mathrm{iT}}$$

$$\forall \mathrm{X}_{\mathrm{i\,jkl}} = 1, \forall i \in A, \forall l \in T, \forall j \in C, \forall k \in D$$

... (ROE2.a)

$$\mathrm{L}_{\mathrm{jP}} = \mathrm{M}_{\mathrm{P}}$$

$$\forall \mathrm{X}_{\mathrm{i\,jkl}} = 1, \forall i \in A, \forall l \in T, \forall j \in C, \forall k \in D$$

... (ROE2.b)

$$\mathrm{L}_{\mathrm{jL}} = \mathrm{M}_{\mathrm{L}}$$

$$\forall \mathrm{X}_{\mathrm{i\,jkl}} = 1, \forall i \in A, \forall l \in T, \forall j \in C, \forall k \in D$$

... (ROE3)

$$\sum_{i=1}^{25}\sum_{j=a}^{b}\sum_{j=1}^{101} X_{i\,jkl} \le 1$$

$$\forall l \in T, \forall a, b \in C \wedge \forall [a,b] \subseteq S \wedge \forall [a,b] \subseteq L$$

... (ROE4)

$$\mathrm{PrvCD}_{\mathrm{i\,jkl}} = \mathrm{X}_{\mathrm{ijkl}} \quad \mathrm{PrvTC}_{\mathrm{i\,jkl}} = \mathrm{X}_{\mathrm{ijkl}}$$

$$\forall \mathrm{X}_{\mathrm{i\,jkl}} = 1, \forall i \in A, \forall l \in T, \forall j \in C, \forall k \in D$$

... (ROE5)

$$\forall X_{i\,jkl} = 1, \forall i \in A, \forall l \in T, \forall j \in C, \forall k \in D$$

... *(ROE6)*

$$X_{i\,jkl} = TTC_j$$

$$\forall X_{i\,jkl} = 1, \forall i \in A, \forall l \in T, \forall j \in C, \forall k \in D$$

... *(ROE7)*

Modelamiento de Restricciones Deseables

Según las restricciones deseables mencionadas anteriormente:

$$\sum_{i=1}^{25}\sum_{j=a}^{b}\sum_{k=1}^{101} X_{i\,jkl} + \sum_{i=1}^{25}\sum_{j=c}^{d}\sum_{k=1}^{101} X_{i\,jkl} \leq 1$$

$$\forall l \in T, \forall a,b,c,d \in C \wedge \forall [a,b], [c,d] \subseteq S$$ ... (RDG1)

Vi ≤ días que ocupa el curso en la semana/ (ultimo día de la semana que ocupa el curso-primer día de la semana usado para el curso) ≤ Vs $\forall j \in C$ ... (RDG2)

Bi ≤ Total de horas ocupadas por algún curso/ (ultima hora ocupada del día por algún curso - primera hora ocupada del día por algún curso) ≤ Bs $\forall i \in I$ ... (RDG3)

$$DTC_{jl} = X_{ijkl}$$

$$\forall X_{i\,jkl} = 1$$

... (RDG4) y (RDE1)

Función Objetivo

Función objetivo global

$$fog(x) = \sum_{x=1}^{14422800} fol(x)$$

$$\forall x \in = \{1,..,14422800\}$$

Donde x representa cada elemento de la matriz, N representa el número total de Elementos en la matriz de cuatro dimensiones. El valor 14422800 es producto de multiplicar las 25 aulas por los 68 cursos por los 101 docentes por los 84 periodos.

Función objetivo local

$$fol(x) = (\ chsl(x) + chsg(x)\ /2) * (\ chpl(x) + chpg(x)\ /2)$$

En donde, chsl y chpl representa los conflictos de aulas y docentes en un mismo semestre, respectivamente, mientras que chsg y chpg representan los conflictos de salones y profesores entre semestres.

Criterio de Optimización

$$Min(fog(x) = 0$$

Hipótesis General

La automatización del proceso de generación de horarios de clases con un margen de error de 2%, mediante el uso de algoritmos genéticos, reducirá el tiempo y los recursos empleados, además de aumentar la calidad de los mismos y elevar el grado de satisfacción por parte del alumnado y el docente.

Variables e Indicadores

A continuación detallaremos la variable independiente y las variables dependientes que forman parte de la hipótesis planteada, así como también el indicador asociado a dichas variables:

Variable independiente 1: La automatización del proceso de generación de horarios de clases.

→ **Indicador:** Número de tareas manuales realizadas en el proceso de generación de horarios.

Variable Dependiente 1: Margen de error del 2%.

→ **Indicador:** Numero de clases del horario académico semestral que infrinjan una o más restricciones del problema.

Variable Dependiente 2: Reducir el tiempo de generación de horarios.

→ **Indicador:** Tiempo en la generación de horarios de clases.

Variable Dependiente 3: Reducir el uso de recursos implicados en la elaboración de los horarios de clases

→ **Indicador:** Costo de los recursos involucrados en la elaboración de horarios de clases.

Variable Dependiente 4: Aumentar la calidad de la solución.

→ **Indicador:** Numero de clases del horario académico semestral que infrinjan una o más restricciones del problema.

Variable Dependiente 5: Aumentar la satisfacción del

alumnado.

→ **Indicador:** Número de reclamos presentados por los alumnos.

Variable Dependiente 6: Aumentar la satisfacción del docente

→ **Indicador:** Número de reclamos presentados por los docentes.

Marco Teórico

En este capítulo se presentarán los conceptos básicos que se requieren para abordar los algoritmos genéticos, así como un ejemplo que permita comprender cómo aplicarlo a un problema de elección.

Introducción a los algoritmos genéticos

Esta técnica se basa en los mecanismos de selección que utiliza la naturaleza, de acuerdo a los cuales los individuos más aptos de una población son los que sobreviven, al adaptarse más fácilmente a los cambios que se producen en su entorno. La técnica de los Algoritmos Genéticos se encuentra dentro de un marco más amplio que viene a ser la Inteligencia Artificial, y más aún en la rama que se ha denominado Computación Evolutiva.

Definición de algoritmos genéticos

John Koza [12] define algoritmo genético :"Es un algoritmo matemático altamente paralelo que transforma un conjunto de objetos matemáticos individuales con respecto al tiempo usando operaciones modeladas de acuerdo al principio Darwiniano de reproducción y supervivencia del más apto, y tras haberse presentado de forma natural una serie de operaciones genéticas de entre las que destaca la recombinación sexual.

Un algoritmo genético está compuesto por [22]:

Módulo evolutivo: Presenta un mecanismo de decodificación que se encarga de interpretar la información de un individuo y una función de evaluación que mide la calidad del mismo.

Módulo poblacional: Posee una representación poblacional y las técnicas necesarias para poder manipularla como son la técnica de representación, el criterio de selección y de reemplazo. Aquí también se define el tamaño de la población y la condición de terminación.

Módulo reproductivo: Contiene los operadores genéticos.

Componentes de un algoritmo genético

Un Algoritmo genético tiene cinco componentes básicos:

1. Una **representación** de las soluciones potenciales del problema.

   Un procedimiento para crear una **población inicial** de posibles soluciones.
2. Una **función de evaluación** que permite clasificar las soluciones en términos de su aptitud.
3. Un conjunto de **operadores de evolución** que alteran la composición de los individuos de la población a través de las generaciones.

   Una **configuración paramétrica** de elementos tales como el tamaño de la población, probabilidad de cruzamiento, probabilidad de mutación, criterio de parada, etc.

Algoritmo genético simple

Existen múltiples propuestas y variantes de algoritmos genéticos. En este parte estudiaremos la propuesta original de Goldberg (1989), conocida como ALGORITMO GENÉTICO SIMPLE.La representación tradicionalmente utilizada es una cadena binaria. A la cadena general se le llama cromosoma. A cada subcadena (posición en la cadena general) se le denomina gen y al valor dentro de esta posición se le llama alelo. La representación es el genotipo que se corresponde con una solución al problema (fenotipo). Existe un proceso de codificación (y su inverso de decodificación) que permite pasar de fenotipo a genotipo y viceversa.

La codificación especifica una función de correspondencia *fC*: S ® {0,1}*, siendo S el espacio de soluciones del problema. La función inversa es la decodificación *fD*: {0,1}* ® S, puede ser una función parcial. La complejidad de *fC* y de *fD* dependerá de las características del problema y de las variables a codificar.

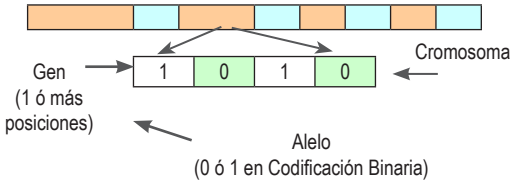

Figura 1. Individuo genético binario

Funcionamiento

El proceso algorítmico básico del AG es como sigue:

1. [Inicio] se genera la población aleatoria de *n* cromosomas (soluciones posibles para el problema).
2. [Aptitud] se evalúa la aptitud f (x) de cada cromosoma x de la población.
3. [Prueba] si la condición de término está satisfecha, se para el algoritmo, se devuelve la mejor solución de la población actual y se va al paso 7.
4. [Nueva población] se crea una nueva población repitiendo los siguientes pasos, hasta que se cumpla la condición de parada.
   a. [Selección] se selecciona dos cromosomas padres, de una población, según su aptitud (cuanto mejor es la aptitud, mayor es la probabilidad de ser seleccionado).
   b. [Emparejamiento] con una probabilidad de emparejamiento, los padres se emparejan para formar a un nuevo descendiente (hijos). Si no se realiza emparejamiento alguno, el descendiente es la copia exacta de los padres.
   c. [Mutación] con una probabilidad de mutación, el nuevo descendiente muta (en alguna posición de su cromosoma).
5. [Sustituir] la nueva población generada es aplicada para otra iteración del algoritmo.
6. [Bucle] se va al paso 2.
7. Fin del algoritmo.

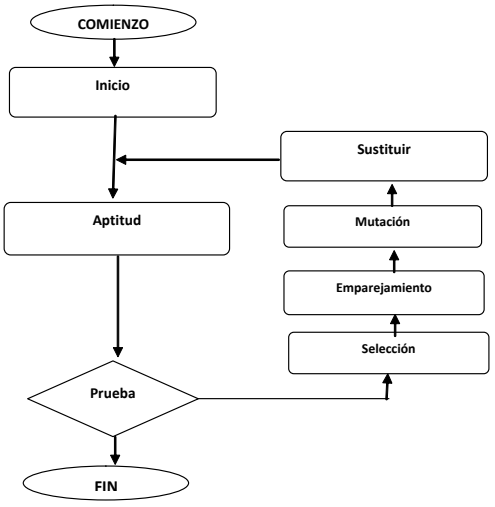

Figura 2. Funcionamiento de un algoritmo genético

Algoritmo básico

El Algoritmo Genético Simple, también denominado Canónico, presenta un conjunto de pasos u operaciones que permiten hallar la solución a un problema. Dado un estado inicial y una entrada, a través de pasos sucesivos y bien definidos se llega a un estado final, obteniendo una solución.

A continuación mostraremos el procedimiento básico del AG:

```
Begin
    t : = 0
    Inicializar P(t) // Generar una población inicial
    Evaluar P(t) // Computar la función de evaluación
    Mientras no sea condición de término, repetir
    t : = t+ 1
    Seleccionar P(t+1) a partir de P(t)
    Emparejamiento sobre P(t) // Obtener descendientes
    Aplicar Mutación sobre P(t) // Según probabilidad
    Sustituir P(t) por P(t+1) // Insertar la nueva generación
    Evaluar P(t) // Computar la función de evaluación

    Fin Mientras
End
```

Figura 3. Pseudocódigo de un algoritmo genético

P(t) es la población de individuos en la generación t; Emparejamiento y Mutación son operadores genéticos que permiten recombinar la información contenida en los cromosomas.

En la aplicación de algoritmos genéticos, se necesita una codificación o representación del problema que resulte adecuada al mismo. Además se requiere una función de evaluación o de adaptación al problema, la cual asigna un valor a cada posible solución codificada indicando la bondad de la solución. Durante la ejecución del algoritmo, los padres deben ser seleccionados para la reproducción o intercambio genético lo que producirá nuevos hijos o soluciones, a los cuales, con cierta probabilidad, se les aplicará una mutación. El resultado de la combinación de los pasos anteriores será un conjunto de individuos, posibles soluciones al problema, los cuales pasarán a formar parte de la siguiente generación.

Operadores Genéticos

Los operadores genéticos consisten en los métodos u operaciones que se pueden ejecutar sobre una población y se dividen en 4 categorías: Selección, Emparejamiento, Mutación y Reemplazo.

Selección: Proceso que escoge los miembros de la

población que serán utilizados para la reproducción. Su meta es dar más oportunidades de selección a los miembros más aptos de la población.

Emparejamiento: Consiste en unir de alguna forma los cromosomas de dos padres para formar dos descendientes. Existen diversas variaciones, dependiendo del número de puntos de división a emplear, la forma de ver el cromosoma, etc.

Mutación: Se encarga de modificar en forma aleatoria uno o más genes del cromosoma de un descendiente.

Reemplazo: Es el método por el cual se insertan los hijos en la población; por ejemplo, mediante la eliminación del individuo más débil o al azar.

3. RESULTADOS Y DISCUSIÓN

Adaptación del algoritmo

Tomando en cuenta las bases teóricas mostradas en los capítulos 2 y 3, la estructura del problema observado en el capitulo 1 y considerando los motivos por los cuales se seleccionó los Algoritmos Genéticos, la parte de este capítulo tiene como objetivo diseñar un algoritmo que trate con eficacia y eficiencia el problema. Las modificaciones que se realicen al algoritmo se tratarán en los siguientes puntos.

Unidad de asignación

En nuestro caso el elemento al cual le vamos a asignar los recursos (docentes, horarios y aulas) es la clase; la cual se define como la combinación de un curso, el tipo de éste, la cantidad de horas a dictar y el grupo.

CURSO	TIPO DE CURSO	HORAS DE DICTADO	GRUPO

Fiigura 4. Estructura de la unidad de asignación (clase)

Una vez definido esto, nuestro algoritmo se encargará de asignar un docente, un horario y un aula determinada a cada una de las clases, los cuales son necesarios para llevarla a cabo.

El proceso que generará el universo de clases, que servirán de elementos de entrada al algoritmo, será un procedimiento lineal. Este tomará en cuenta los tipos de curso (teoría, práctica y laboratorio), la cantidad de horas de dictado por cada tipo, los grupos que se aperturen para el semestre en curso y la siguiente restricción obligatoria:

"**ROG7**. Un curso debe cumplir con una cantidad de horas semanales requeridas. Según sea el caso si el número de horas académicas es par se asignará dos horas diarias hasta cubrir la totalidad, sino se asignará un día de tres horas y el resto de dos horas. A cada uno de estos grupos de horas se les llamará bloques".

Definiendo de este modo la unidad a la cual se asignará los recursos y asegurando que el algoritmo asignará recursos a todas las clases sin excepción con la finalidad de generar un horario factible, entonces se tendrán cumplidas de antemano la restricción obligatoria anterior así como la siguiente:

"**ROG8**. Las horas de un bloque para un determinado curso en un mismo día deben de ser consecutivas además de ser asignados a un mismo docente y aula"

Esto nos permitirá ahorrar tiempo computacional debido a que se evitará la evaluación del cumplimiento de estas restricciones, y además se podrá simplificar la implementación del algoritmo.

Mediante un ejemplo sencillo veremos la generación de las clases y su correspondiente estructura: Consideremos el curso de Algorítmica I, el cual tiene 3 tipos definidos y se le aperturarán 3 grupos en el presente semestre, además en la siguiente tabla tenemos las horas semanales de dictado por cada tipo definido para el curso:

Tipo de Curso	Horas de Dictado
Teoría	5
Practica	3
Laboratorio	4

Tabla 1. Horas de Dictado de Algorítmica I

Ahora tomaremos en cuenta lo expresado en la restricción **ROG7** : Para el curso de Algorítmica I de tipo teoría se tiene asociado una cantidad de horas de dictado impar, por lo tanto se tendría una clase de teoría de tres horas y otra de dos. Por consiguiente el total de clases generadas serían:

Curso	Tipo	Horas	Grupo
Algorítmica I	Teoría	3	1
Algorítmica I	Teoría	2	1
Algorítmica I	Practica	3	1
Algorítmica I	Laboratorio	2	1
Algorítmica I	Laboratorio	2	1
Algorítmica I	Teoría	2	2
Algorítmica I	Teoría	3	2
Algorítmica I	Practica	3	2

Algorítmica I	Laboratorio	2	2
Algorítmica I	Laboratorio	2	2
Algorítmica I	Teoría	2	3
Algorítmica I	Teoría	3	3
Algorítmica I	Práctica	3	3
Algorítmica I	Laboratorio	2	3
Algorítmica I	Laboratorio	2	3

Tabla 2. Clases generadas para el curso de Algorítmica I

Separación en Sub-fases

En este punto veremos cómo el problema, que como ya hemos mencionado anteriormente corresponde a la asignación de recursos (docentes, aulas, horarios) a las clases, será simplificado debido a que no se asignarán todos en un mismo momento. Dicha asignación se realizará uno a uno a través de diferentes fases delimiándose el espacio de soluciones cada vez que se van asignando dichos recursos.

A continuación definimos 3 fases según los recursos a asignar:

Fase 1: Asignación de docentes a las clases

Fase 2: Asignación de horarios a las clases

Fase 3: Asignación de aulas a las clases

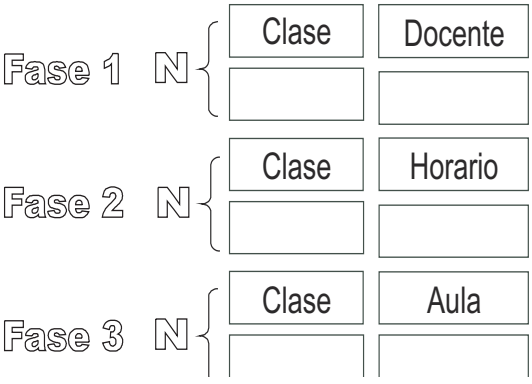

Figura 5. Fases para la asignación de clases a los recursos

A continuación veremos con más detalle cada una de las fases y la forma como éstas interactúan:

Primera fase asigna los docentes a las clases, tomando en cuenta que todas las restricciones obligatorias correspondientes al docente y al curso se cumplan. La solución resultante pasará a la

Segunda fase, la cual se encargará de asignar diversos horarios a las clases cerciorándose de que se cumplan las restricciones relacionadas a los periodos de tiempo, cursos y docentes; además se debe cerciorar que cada horario no tenga más asignaciones que aulas disponibles, dado de que si esto no se verifica entonces no existiría solución posible para la siguiente fase.

Tercera fase asigna aulas a las clases, las cuales ya poseen horarios y docentes asignados como resultado de la primera y segunda fase.

Puesto que existe un mayor número de restricciones implicadas con los horarios, resulta mejor considerar esta fase en segundo lugar debido a que controlaremos las restricciones de los docentes con respecto a los horarios que no se pudieron controlar en la primera fase. Y no se ha considerado como última fase debido a que no hay restricciones en el uso de las aulas con respecto a los horarios considerados.

En la fase uno encontramos restricciones que están relacionadas con la preferencia de dictado de cursos por parte de los docentes y la disponibilidad de éstos. En la fase dos, referida a los horarios, hay restricciones tales como la determinación de la facultad a programar algunos cursos solamente por la mañana ó por la tarde, la no posibilidad de programarlos en ciertas horas, entre otras. En la fase tres podemos manejar restricciones relacionadas a la necesidad de ciertas clases por hacer uso de aulas específicas. Esto puede ser ejemplificado con la programación de aulas para las clases de laboratorio de ciertos cursos. Sin embargo, debemos observar que hay relativamente pocas restricciones referidas a las asignaciones de las aulas porque el número de aulas en la facultad es grande. De hecho, la asignación de aulas a las clases que tienen ya horarios asignados es una tarea de menor complejidad comparada a la de asignar horarios a las clases, igualmente ocurre con la primera fase correspondiente a la asignación de docentes la cual también es de menor complejidad.

Si quisiéramos atacar el problema en una sola fase implicaría validar todas las restricciones obligatorias por la asignación de horarios, docentes y aulas a las clases, tendríamos entonces restricciones obligatorias tales como que no se debe asignar dos o más clases en el mismo horario y aula, ó asignar a un docente a más de una clase en el mismo horario. Puesto que un algoritmo genético es una técnica metaheurística, esto es absolutamente posible a lo largo del curso de su evolución. Dado que penalizamos tal situación, los algoritmos genéticos se esfuerzan eventualmente a

quitar tales asignaciones, pero esto está a expensas de, posiblemente, cientos de iteraciones para encontrar la solución factible. Para ilustrar esta idea supongamos que la asignación del docente y horario ya están definidos y ocurriese un conflicto de dos clases en un mismo horario y aula, simplemente deberíamos cambiar el aula de clases a otra aula con el mismo horario; esto es cierto especialmente para esas clases que no tienen preferencia alguna de lugar. Este cambio es siempre posible, y se hace a menudo, siempre y cuando el número de las clases asignadas a un horario no exceda el número de las aulas disponibles en ese horario. Lo mismo ocurre con los docentes, podemos cambiar el horario asignado a otro horario con el mismo docente siempre y cuando los horarios disponibles para el docente no sean sobrepasados.

Actualmente, el número de aulas disponibles es el mismo en cualquier horario, no cambia con respecto al tiempo, en otras palabras las aulas se encuentran disponibles en su totalidad para todos los horarios considerados; mientras que para el caso de los docentes y su disponibilidad, éstos son considerados datos más volátiles y complejos. Por lo tanto, podemos ahorrar centenares de iteraciones adicionales de heurística, separando el problema en fases.

La simplificación y la solución del problema, dividiéndolo en estas tres fases fomentan el hallazgo de soluciones de una manera más rápida, puesto que la combinación de los conflictos de los docentes, de los horarios y de las aulas no se considera simultáneamente, sino en diversas fases.

Representación del problema

La solución se realiza en tres fases, por tanto, tendremos que definir tres cromosomas, uno para cada fase. Cada uno de los cromosomas tendrá una cantidad de genes igual al número de clases y el valor de estos genes (alelos) será igual al recurso asignado a la clase. Una ilustración áspera de un cromosoma es como sigue:

Clase 1	Clase 2	Clase 3	…	Clase N
Recurso i	Recurso j	Recurso k	…	Recurso l

Figura 4.4. Representación vectorial del cromosoma

Es importante observar que los tres cromosomas son iguales a nivel de estructura, pero aquello que los diferenciará será el significado que tengan los alelos (valores de los genes) en cada fase; es decir, los alfabetos usados en cada una de las fases serán distintos.

Codificación mediante alfabetos duros

Como ya se ha mostrado teóricamente existe una codificación directa y una codificación indirecta, siendo ésta última la más común. En la codificación indirecta se traza la gama de valores posibles en una dimensión más simple, a menudo un espacio binario. La representación binaria tiene ventajas de cómputo y de programación debido a que presenta una estructura más simple; sin embargo, cuando tratamos casos más complejos es difícil tratarlo a través de una codificación indirecta debido a la complejidad de cómputo que implicaría tener que traducir cada representación.

Para nuestro caso, notamos que la utilización de una representación directa, debido a su facilidad de evaluación, permite el diseño de un sistema más complejo tanto en sus condiciones y sus objetivos. Es por ello que para nuestro modelo consideraremos una codificación directa en lugar de una codificación indirecta.

Alfabeto utilizado

En base a los requisitos particulares, y luego de establecida la arquitectura y codificación a utilizar, mostraremos el alfabeto que se empleará en cada una de las fases:

Fase 1	Fase 2	Fase 3
el alfabeto constará de enteros que representarán a cada docente, los cuales serán asignados a cada una de las clases en el semestre actual. Con esto tendremos un alfabeto específico asociado a cada clase a la cual le vayamos asignar un docente	Se usara la misma forma de representación para los horarios	Se usara la misma forma de representación para las aulas

De esta manera logramos que el valor que vaya a tomar cada gen no se encuentre fuera del dominio establecido para cada recurso, ya que con una representación binaria podríamos obtener un conjunto de valores que no sean correctos y que no correspondan al rango

de valores de los recursos. Por ejemplo, tenemos un conjunto de 20 docentes cuyo alfabeto estará definido desde el valor 1 al 20 y si para ello usamos una combinación binaria para representar dichos valores, entonces podríamos obtener valores que se encuentren fuera de este rango.

También puede suceder que a las clases no se les pueda asignar ciertas aulas o ciertos horarios o ciertos docentes por algún motivo, entonces nosotros podemos quitar esas combinaciones del alfabeto correspondiente. Por ejemplo, en caso que queramos asignar un docente a una clase, deberíamos primero seleccionar al docente entre los que ya están definidos; pero imaginemos que sólo dos de ellos tienen preferencia para el dictado de dicha clase, entonces si quitamos ciertos valores del alfabeto evitaríamos que se asigne un docente que no tenga preferencia por dicha clase.

Esta clase de codificación dura protege al modelo contra la violación de restricciones obligatorias, ahorrando de esta manera recursos computacionales y minimizando el tiempo. Bajo este criterio es imposible que los segmentos del cromosoma tomen valores fuera del espacio factible de soluciones. En nuestro problema timetabling, esto significa que si un docente desea enseñar solamente en los horarios de la mañana, el algoritmo genético nunca le asignará una clase de la tarde en toda su población, desde la primera iteración hasta la última. La probabilidad de que no se cumpla esta restricción es nula; sin embargo, esto también hace que al sistema le resulte difícil encontrar una solución, puesto que se ocupará de generar los diversos espacios de alfabetos que estarán asociados a las clases en cada una de sus fases.

Una vez detallada la conceptualización de nuestra idea veamos las restricciones que se emplearán en cada una de las fases, las cuales limitarán el uso de alfabetos en los diferentes genes de la población y a su vez no tendrán que ser tomadas en cuenta en la evaluación de la solución.

En la fase uno (docente - clase), acotar la lista de docentes que puedan dictar una clase determinada, nos permitirá dejar de lado la evaluación de las siguientes restricciones: ROG4, ROG5, ROE6, ROE7 y ROE5	En la fase dos (horario - clase), análogamente a lo indicado en la fase uno, los horarios en los que se pueda dictar una clase, nos permitirá dejar de lado la evaluación de las siguientes restricciones: ROE5, ROE7, ROG5,	en la fase tres (aula - clase), la limitación de las aulas en las que se pueda dictar una clase, nos permitirá dejar de lado la evaluación de las siguientes restricciones: ROE2, ROE3, ROG6

Inicialización de la población

La primera etapa en un Algoritmo Genético es la generación de la población inicial, que como ya se mencionó se suele realizar de manera aleatoria. Aquí se darán valores a todos los genes de los N cromosomas, donde N representa el tamaño de la población inicial, es decir, la cantidad de soluciones que se considerarán en cada etapa.

Como en nuestro caso se abordará el problema en tres fases diferentes teniendo para cada una de éstas un algoritmo genético asociado, el cual se encargará de la asignación de un recurso en particular (docentes, horarios y aulas) a las clases, necesitaremos entonces también tres fases de inicialización.

Con respecto a la estructura del cromosoma, en nuestro caso estará determinada por un arreglo de enteros para cada una de las tres fases consideradas. Una primera apreciación podría considerar la asignación, a cada índice del arreglo, de valores aleatorios dentro del conjunto de los enteros, desde 1 a un límite superior que vendría a ser el número total del recurso según la fase. Por ejemplo consideremos que deseamos inicializar diez clases con docentes y contamos con una plana de veinte docentes, entonces tendremos un arreglo de longitud diez donde cada índice del arreglo corresponderá a una clase, y el arreglo como sabemos representará al cromosoma. Además codificaremos cada docente del uno al veinte de manera consecutiva. Por lo tanto la inicialización comenzaría seleccionando un valor aleatorio entre uno y veinte para asignarlo al índice 0 del arreglo, luego escogeríamos otro valor aleatorio en el mismo rango y lo asignaríamos al valor del índice 1, y así sucesivamente hasta completar el cromosoma con valores y posteriormente formar todos los cromosomas de la población inicial.

Con esto tendríamos la población lista para comenzar a ejecutar el algoritmo, sin embargo vemos ciertas deficiencias en este método. Imaginemos que queremos asignar un docente a la clase de Algorítmica I y según el método de inicialización descrito, deberíamos entonces asignarle aleatoriamente un docente entre los veinte ya definidos, Resulta que sólo tres de ellos tienen preferencia para el dictado de Algorítmica I, entonces podría suceder que se asigne en la población inicial un docente que no tenga preferencia por el curso. Las opciones para el manejo de este error serían: desecharlas, asignándoles una baja aptitud en el momento de la

evaluación, lo cual implicaría su desaparición en términos estadísticos al cabo de unas pocas generaciones junto a la pérdida de información valiosa que pudieran contener; o Repararlas, lo cual aparece como una tarea costosa pues habrá de ser realizada en una gran cantidad de ocasiones a lo largo del proceso evolutivo. Ninguna de las alternativas parece óptima en sí misma.

La opción que proponemos consiste en evitar la generación de este tipo de individuos no factibles. Este problema podría evitarse si al momento de generar la población inicial no se tomarán en cuenta ciertos recursos que por las restricciones previamente nombradas, sería imposible que se use para una determinada clase. Usando el mismo ejemplo anterior para la clase de Algorítmica I debería asignarse un docente aleatorio pero no del conjunto total de veinte docentes sino solamente de los tres posibles que pueden dictar dicho curso.

Con esto estamos generando una población inicial que de antemano ya cumple ciertas restricciones, lo cual aliviará la tarea del método que evaluará las soluciones, ya que no serán tomados en cuenta durante la evaluación. Dado que si los cromosomas iniciales no consideran estos valores, las poblaciones subsecuentes generadas nunca los tendrán presentes, con ello se pretende ahorrar tiempo en la ejecución del algoritmo. Esta solución tiene la ventaja añadida de permitir eliminar directamente soluciones de muy mala calidad sin necesidad de evaluarlas.

Dado que la estructura del cromosoma es igual en cada fase, esto es, un arreglo de longitud igual al número de clases a las cuales se les va asignar un recurso, el método utilizado (mostrado en el ejemplo anterior) para la inicialización de la población será el mismo para cada una de las fases. Lo que variará por cada fase, aparte del recurso en particular, serán los alfabetos generados para cada gen de los cromosomas; esto se realizará a través de un procedimiento lineal que acotará dichos alfabetos.

El hacer uso de un alfabeto en particular para cada gen del cromosoma, en cualquiera de las tres fases, nos permitirá en gran medida ahorrar la evaluación de un gran número de restricciones, dado que las soluciones iniciales elaboradas no presentarán la infracción de estas restricciones. Con lo cual podríamos considerar ésta como una mejor solución que simplemente el hecho de asignar valores aleatorios.

Otro aspecto a tener en cuenta, es la posibilidad de introducir en la población inicial soluciones de buena calidad, por ejemplo, procedentes de la experiencia acumulada en procesos anteriores. Esta forma de introducir conocimiento específico es muy útil pues permite al algoritmo contar con información valiosa inicialmente. No obstante debe ser realizado con cuidado, pues sembrar la población inicial con soluciones no aleatorias puede alterar el proceso evolutivo al dirigir la búsqueda hacia un determinado espacio.

4. CONCLUSIONES

La Facultad de Ingeniería de Sistemas e Informática de la UNMSM no utiliza herramientas computacionales para generar los horarios de clase, los cuales manejan un alto grado de complejidad durante su elaboración. Es por esto que se planteó desarrollar una herramienta basada en algoritmos genéticos, que permita cubrir las necesidades existentes y en cuyo proceso de investigación se llegó a las conclusiones que se presentan a continuación:

1. Se construyó un algoritmo que permite encontrar soluciones buenas y/o consideradas aceptables dentro del margen de error definido y restricciones planteadas. Con lo cual se deduce que el objetivo general de esta investigación, que es encontrar una solución mediante el uso de algoritmos genéticos al problema de asignación de docentes, periodos y aulas a los cursos, se ha cumplido a cabalidad.
2. Otra conclusión, desprendida del proceso de investigación, está relacionada con la división del problema en distintas fases. Inicialmente no se pudo apreciar el impacto total de su aplicación sobre el problema, pero una vez concluida la investigación se observó lo siguiente:
   2.1. La división por fases permite reducir el tiempo de procesamiento y el uso de recursos; debido a que en la medida que se van encontrando soluciones en cada fase, las cuales resultan ser información de entrada de la fase subsecuente, el espacio de soluciones se va acotando cada vez más. La aplicación de la técnica “divide y vencerás” resultó de gran ayuda al momento de tratar de resolver el problema a partir de la solución de subproblemas.

   Otro logro conseguido gracias a la división en fases fue la simplicidad obtenida en la implementación del algoritmo, debido a que las res-

tricciones relacionadas a los docentes, periodos y aulas fueron tomadas en cuenta de manera separada unas de otras, de acuerdo al recurso asociado a cada fase.

2.2. Se pudo observar una debilidad al método. Pues tal vez al momento de acotar el espacio de soluciones mediante la generación de una solución parcial por parte de una fase anterior, se perdían ciertas soluciones factibles que ya no podrían ser exploradas por las fases subsecuentes y en la cual podría estar la solución que estemos buscando. Para solucionar esto se planteó maximizar la diversidad de individuos dentro de la población, de tal manera que nuestra población se encuentre lo más distribuido posible dentro del espacio de búsqueda acotado donde el algoritmo podía desplazarse. Esto significa hacer que los individuos sean lo más diferentes posibles en la población sin interferir con los procesos de evolución normal.

TRABAJOS A FUTURO

1. Se pueden combinar algoritmos genéticos con otra metaheurística para mejorar el tiempo de generación y calidad de la solución. Por ejemplo, la función que se encarga de calcular la aptitud de cada individuo de la población en las diferentes fases podría ser remplazada por una red neuronal, la cual se encargaría de realizar dicho cálculo. Con lo cual se dotaría a este nuevo algoritmo híbrido una característica de aprendizaje adicional a una búsqueda guiada.

   A su vez podría utilizarse una heurística de búsqueda, tal como un algoritmo voraz, que se encargue de generar la población inicial con cierto grado de preprocesamiento y que por ende sea mejor que simplemente inicializar la población de manera aleatoria.
2. La segunda mejora sería la de implementar un servicio Web con este algoritmo para que se pueda tener acceso a él, de tal manera que no sea usado solamente por la facultad sino también abrir la posibilidad de brindar este servicio a otras facultades y/o instituciones que lo requieran. Otra ventaja de implementar el servicio sería que se estaría destinando la carga del procesamiento a un servidor dedicado para liberar al cliente de dicha tarea.
3. Una vez establecido el procesamiento del algoritmo en un servidor, podrá ser organizado sobre una arquitectura de procesamiento distribuida. En este punto se presentan varias opciones, una es armar una arquitectura de computación paralela (computación grid) ya que proporciona una gran cantidad de recursos según se necesite, claro que este modelo sería independiente al algoritmo y tendría que realizarse un trabajo previo para implementar esta arquitectura. Otra opción más ligada a la implementación de la solución sería elaborar un modelo de procesamiento maestro-esclavo, donde un servidor maestro controle el proceso evolutivo, concentre la información resultante y asigne carga de trabajo a sus servidores esclavos. Dicha carga de trabajo podría ser la evaluación de una parte de los individuos de la población, dado que esto es lo que más tiempo toma en procesar.
4. Con la finalidad de hacer que el software sea más configurable, se recomienda permitir a los futuros usuarios poder escoger entre todas las restricciones que maneja el sistema y decidir cuáles se acoplan a su realidad ó problema específico.
5. Como última recomendación, se sugiere aumentar el potencial del software al momento de la generación de horarios integrando esta herramienta con la información contenido en el SUM de la UNMSM con la cual se podría calcular de manera automática la cantidad de grupos que se deberían aperturar para los diferentes cursos de manera automática o por lo menos sugerirlos para su aprobación.

5. FUENTES BIBLIOGRÁFICAS

Libros

Dawkins R. (1976). “The Selfish Gene”. USA, New York:Oxford University Press.

Díaz, Adenso (1996). Optimización heurística y redes neuronales. España: Editorial Paraninfo.

Fogel, David B (1995). Evolutionary Computation: Toward a New Philosophy of Machine Intelligence. USA: IEEE Press.

Garey, Michael R. (1979). Computers and intractability. A guide to the Theory of NP- Completeness. USA:Bell Telephone Laboratories.

Gil Londoño, Natyhelem (2006). Algoritmos Genéticos. Medellín.

Gil Tallavo, Marcos y Antonio Martínez, Amadís (2006). "Algoritmo basado en Tabú Search para el Problema de Asignación de Horarios de Clases". Departamento de Computación Facultad de Ciencias y Tecnología Universidad de Carabobo Valencia, Estado Carabobo, Venezuela.

Glover, F. y Laguna, M. (1993). Tabú Search. Modern Heuristic Techniques for Combinatorial Problems, USA: C. Reeves, ed. Blackwell Scientific Publishing.

Goldberg, D. y Voessne, S (1999). "Optimizing global-local search hybrids". In GECCO'99: Proceedings of the Genetic and Evolutionary Computation Conference. pp. 220-228.

Goldberg, David E. (1989). "Genetic Algorithms in Search, Optimization, and Machine Learning", Addison-Wesley Publishing Company.

Johnsonbaugh, Richard (1999). Matemáticas Discretas. Cuarta edición, Prentice Hall.

Kaufmann, A. y Faure, R. (1977). "Invitación a la Investigación de Operaciones". Segunda edición, México: CECSA.

Koza, John R. (1992). Genetic Programming. On the Programming of Computers by Means of Natural Selection. USA: The MIT Press.

Krasnogor, N. y Smith, J. (2005). "A Tutorial for Competent Memetic Algorithms: Model, Taxonomy and Design Issues". IEEE Transactions on Evolutionary Computation.

Leiserson, C. Rivest, R. y Stein, C. Cormen, T. (2001). Introduction to algorithms. USA:The MIT Press McGrawHill.

López Ceballo, Paulina Danae (2000). "El Método de búsqueda Tabú para la programación de horarios". México: Editorial Universidad de Sonora.

Macías, E. A. (2004). "Algoritmos de Búsqueda Local de GRASP para Solución de 3-SAT". pp. 219-225.

Merz, P. y Freisleben, B.(1999) "A comparison of memetic algorithms, tabú search, and ant colonies for the quadratic assignment problem". In Proceedings of the 1999 International Congress of Evolutionary Computation (CEC' 99). Washington DC, USA.

Michalewicz, Zbigniew (1996). "Genetic algorithms + data structures = evolution programs". Alemania: Editorial Springer.

Moscazo, P. (1989). "On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms". California Institute of Technology, Pasadena, California, USA, Tech. Rep. Caltech Concurrent Computation Program, Reporte 826.

Saleh M.A, Elmohamed y Fox, Geoffrey (1998) "A comparison of Annealing Techniques for Academic Course Scheduling". Practice and Theory of Automated Timetabling II, Selected Papers from the 2nd International Conference, PATAT'97.

Vázquez Espi, Mariano (1994). "Recocido simulado: un nuevo algoritmo para la optimización de estructuras". Madrid.

Whitley, Darrell (1997). A genetic algorithm tutorial. Holanda: Editorial Springer.

Revistas

Larrosa, J. y Meseguer (2003). Restricciones blandas: modelos y algoritmos. Revista Iberoamericana de Inteligencia Artificial, Vol. Otoño/2003, N.º 20, pp. 69-81.

Alkan, Alpay y Özcan,Ender (2003). "Memetic Algorithms for Timetabling". Evolutionary Computation, 2003. CEC '03. The 2003 Congress on, Vol. 3, pp. 1796-1802.

Berretta R,Cotta C, y Moscato P. (2003) "Enhancing the performance of memetic algorithms by using a matching-based recombination algorithm: Results on the number partitioning problem". In M. Resende and J. Pinho de Sousa, editors, Metaheuristics: Computer Decision Making. Kluwer Academic Publishers, Boston MA.

Casado Yusta, Silvia y Pacheco Bonrostro, Joaquín (2003). "Estudio Comparativo de diferentes estrategias meta heurísticas para la resolución del Labor Scheduling Problem". Estudios de Economía Aplicada, Vol. Diciembre/2003, N.º 21, pp. 537-554.

Franco Baquero, John Fredy, Eliana Mirledy Toro Ocampo y Ramón Alfonso Gallego Rendón (2008). Course timetabling problem resolved using Tabu Search. Ingeniería y Desarrollo N.º 24.

Glover, F (1989). Tabú search - part I. ORSA Journal on Computing, Vol. Verano/2003, N.º 3.

Kirkpatrick S., Gelatt C.D. y Vecchi M.P, (1983) "Optimization by Simulated Annealing". Science, Vol. 220, Nº 4598.

Melián, B., Moreno Pérez, J.A., Moreno Vega, J.M. (2003) Meta heurísticas: Una visión global. Revista Iberoamericana de Inteligencia Artificial 19, 2, 7-28.

Mushi, A. R. (2007). "Simulated Annealing Algorithm for the Examinations Timetabling Problem". African Journal of Science and Technology (AJST) Science

and Engineering Series Vol. 8, Nº. 2, pp. 24- 32.

Naji Azimi,Zahra (2004). Comparison of metaheuristic algorithms for examination timetabling problem. J. Appl. Math. & Computing Vol. 16(2004), Nº 1-2, pp. 337-354.

Fuentes Electrónicas

Algoritmos genéticos y computación evolutiva. Abril 2004 (Referido 01/10/2009). Disponible en URL: http://the-geek.org/docs/algen/

Algoritmo voraz. Julio 2006 (Referido 01/12/2009). Disponible en URL: http://es.wikipedia.org/wiki/Algoritmo_voraz

Inteligencia artificial. Octubre 2009 (Referido 06/11/2009). Disponible en URL: http://es.wikipedia.org/wiki/Inteligencia_artificial

Java Docs. Octubre 2009 (Referido 15/12/2009). Disponible en URL:http://jgap.sourceforge.net/javadoc/3.4.4/

Matriz (matemática). Septiembre 2009 (Referido 01/11/2009). Disponible en URL: http://http://es.wikipedia.org/wiki/Matriz_(matem%C3%A1tica)

Programación Lineal. Octubre 2007 (Referido 01/12/2009). Disponible en URL: http://es.wikipedia.org/wiki/Programaci%C3%B3n_lineal

Red neuronal artificial. Agosto 2008 (Referido 10/11/2009). Disponible en URL: http://es.wikipedia.org/wiki/Red_neuronal_artificial

Timetable. Octubre 2007 (Referido 01/10/2009). Disponible en URL: http://en.wikipedia.org/wiki/Timetable

Vector (informática). Diciembre 2008 (Referido 01/11/2009). Disponible en URL: http://es.wikipedia.org/wiki/Matriz_(programaci%C3%B3n)

Vuelta Atrás. Febrero 2006 (Referido 20/11/2009). Disponible en URL: http://es.wikipedia.org/wiki/Vuelta_Atr%C3%A1s