Recibido: 28 de Febrero de 2011/Aceptado: 29 de Junio
de 2011 Publicado en línea: 06 de septiembre de 2011

Revisión de Algoritmos Genéticos Aplicados al Problema de la Programación de Cursos Universitarios

Mireya Flores Pichardo

CIICAp, Universidad Autónoma del Estado de Morelos
Av. Universidad 1001, Chamilpa, 62209, Cuernavaca Morelos, MÉXICO

Resumen. La programación de horarios académicos es un problema particular que se encuentra dentro del problema general de asignación de recursos. Este problema de horarios, se conoce en la comunidad científica como Problema de Programación de Horarios Universitarios. Los problemas de programación de horarios consisten en generar horarios para tareas definidas, buscando cumplir de la mejor manera con condiciones o requerimientos específicos.
Este problema ha sido tratado con diferentes métodos, por ejemplo Colonia de Hormigas, Búsqueda Tabú, Coloreo de grafos y Algoritmos Genéticos. En éste trabajo se hace una revisión de algunos algoritmos evolutivos que han abordado el problema de horarios académicos aplicando diferentes modelos.

Palabras Clave: Programación de Horarios, Optimización Combinatoria, Heurísticas, Algoritmos Genéticos.

Abstract. The academic scheduling is a particular problem within the general problem of resource allocation. This scheduling problem is known as the University Timetabling Problem in the scientific community. The scheduling problems consist on generating schedules for defined tasks by pursuing the best way to specific conditions or requirements.
This problem has been addressed with different methods such as Ant Colony, Tabu Search, Graph Coloring and Genetic Algorithms. In this paper we review some evolutionary algorithms that have addressed the problem of academic schedules applying different models.

Keywords: Timetabling, Combinatorial Optimization, Heuristics, Genetic Algorithms.

1. Introducción

Los problemas de programación de horarios consisten en generar horarios para tareas definidas, buscando cumplir de la mejor manera con condiciones o requerimientos específicos. Estos problemas son muy comunes y se encuentran en distintos tipos de actividades tales como: Actividades Educacionales, Universidades, Colegios, Institutos, Facultades, Departamentos, Actividades Deportivas, Actividades de Transporte y otras actividades que involucran a personas y el uso de equipos [2]. En el área educativa, se requiere la creación y planeación de horarios de la forma más eficiente posible en cuanto a la designación y organización del personal, horarios, salones y equipos para conseguir un mejor aprovechamiento de los recursos y tendiente a lograr un mejor desempeño de los alumnos y el personal.

La programación de horarios académicos es un problema particular que se encuentra dentro del problema general de asignación de recursos, problema clasificado dentro de la teoría de la complejidad como NP-completo [3] y cuya importancia estriba en su aplicación práctica puesto que se observa en la asignación de equipo en la industria de la manufactura, en el transporte público o en la calendarización de horarios de escuelas. El problema de horarios, se conoce en la comunidad científica como Problema de Programación de Horarios Universitarios [1] El personal que tiene a su cargo la organización de los horarios de alumnos y profesores se encuentran cada periodo académico con la necesidad de proporcionar una solución lo más adecuada posible de acuerdo a una serie de restricciones, que para un programa educativo que pretenda observar medidas de calidad, implican un gran esfuerzo.

El problema de programación de horarios en el área educativa consiste en programar las asignaturas de un periodo académico (trimestre, cuatrimestre, semestre), considerando:

- Los profesores necesarios en cada asignatura
- Los grupos de alumnos que toman un conjunto de asignaturas,
- Los días o periodos disponibles,
- Los salones requeridos (tratando de que satisfagan un conjunto de restricciones relacionadas con el programa de estudio),
- El personal con que se cuenta y
- Las instalaciones.

Las soluciones a un problema de programación de horarios que son construidas de manera manual, además de tomar días o semanas, no garantizan una solución ideal pues puede darse el caso de que los alumnos se encuentren en la situación de no poder tomar dos materias que deberían porque los horarios se traslapan, puede ocurrir que se pasen por alto algunas restricciones, entre otros casos más.

Esta situación da lugar a un problema cuya resolución se complica cuanto mayor sea el número de restricciones se deban tomar en cuenta. Dada la complejidad del problema de calendarización de horarios académicos su solución se centra en la búsqueda de métodos heurísticos que encuentren buenas soluciones. Los Métodos heurísticos son procedimientos eficientes para encontrar buenas soluciones. En estos métodos, la

rapidez del proceso es tan importante como la calidad de la solución obtenida. Un método heurístico es un procedimiento para resolver un problema de optimización bien definido mediante una aproximación intuitiva, en la que la estructura del problema se utiliza de forma inteligente para obtener una buena solución [4].

Una serie de métodos con el nombre de Metaheurísticas han surgido con el propósito de obtener mejores resultados que los alcanzados por los métodos heurísticos tradicionales. Las metaheurísticas son estrategias para diseñar o mejorar los procedimientos heurísticos con miras a obtener un alto rendimiento. El término metaheurística fue introducido por Fred Glover en 1986 y a partir de entonces han aparecido muchas propuestas de pautas o guías para diseñar mejores procedimientos de solución de problemas combinatorios.

Las metaheurísticas se sitúan conceptualmente ”por encima” de las heurísticas en el sentido de que guían el diseño de éstas, pueden estar compuestas por una combinación de algunas heurísticas, por ejemplo una metaheurística puede usar una heurística constructiva para generar una solución inicial y luego usar otra heurística de búsqueda para encontrar una mejor solución.

Dentro de las metaheurísticas se encuentran Búsqueda Tabú (Tabu search), Recocido Simulado (Simulated annealing), Algoritmos Genéticos, Búsqueda Dispersa (Scatter search), entre otros [5]. Los algoritmos genéticos son métodos adaptativos utilizados para resolver problemas de optimización. Estos algoritmos están basados en el proceso genético de los organismos biológicos y simulan el principio de selección natural, siendo posible con ellos evolucionar la forma de solucionar problemas del mundo real. Por ejemplo pueden ser usados para diseñar estructuras de puentes o para determinar la forma en que se disminuye el desperdicio en el corte de patrones de ropa [6].

2. Un Algoritmo Genético para Resolver el Problema de Programación de Cursos.

En [7] se presenta la aplicación de un algoritmo genético adaptado a un caso real del problema de calendarización de cursos de una escuela italiana. El problema considerado está dado por:

- Una lista de m profesores (20-24 en el caso de estudio);
- Una lista de p clases involucradas (10);
- Una lista de n horas para cada clase (30);
- Algunas condiciones externas (por ejemplo las horas durante las cuales algunos profesores están involucrados en otras secciones de actividades).

La representación formal encontrada en Colorni es la siguiente:

Dada una 5-tupla < **T, A, H, R, f** > donde

T es un conjunto finito **{T1, T2,.., Ti,.., Tm}** de *m* recursos (profesores);

A es un conjuntos de trabajos (enseñar en las p clases o en otras actividades)

a ser cumplidos por los profesores;

H es un conjunto finito **{H1, H2,.., Hj,.., Hn}** de n intervalos de tiempo (horas);

R es un matriz *m.n* de *rij* ∈ *A* (un horario de clases);
f es la función a ser minimizada.

Cada solución (horario) factible generada por el algoritmo implementado satisface las siguientes restricciones:

- Cada profesor y cada clase debe estar presente en la programación en un número de horas predefinido;
- No debe haber más de un profesor en la misma clase a la misma hora;
- Ningún profesor debe estar en dos clases a la misma hora;
- No debe haber "horas sin cubrir" (esto es, horas de clase a las que no se haya asignado un profesor).

Para la representación del horario de clases en Colorni se utiliza un alfabeto que es el conjunto A de trabajos que los profesores deben llevar a cabo. Sus elementos son las clases a cubrir y otras actividades. Ellos indican:

- Con los caracteres 1, 2, 3,.. ,0 las 10 clases donde las lecciones deben ser enseñadas;
- Con el caracter D las horas disponibles para asesorías temporales
- Con el caracter A las horas para el desarrollo profesional;
- Con el character S las horas durante las cuales las lecciones son enseñadas en clases de secciones diferentes de las 2 consideradas; estas horas son arregladas en la fase de inicialización y son llamadas "*horas arregladas*"
- Con el carácter ♦ las horas en las cuales el profesor no tiene trabajo;
- Con los caracteres ----- el día libre del profesor.

Ese alfabeto permite representar el problema como una matriz **R** (una matriz *m.n* de *rij* ∈ **A**) donde cada fila corresponde a un profesor y cada columna a una hora. Cada elemento *rij* de la matriz **R** es un gen; su valor puede variar en el subconjunto de **A** específico al profesor correspondiente a la fila que contiene el gen. La Figura 1 muestra un ejemplo de representación de horario.

Teacher-Subject	Mon	Tue	Wed	Thu	Fri	Sat
Literature - 1	♦11♦1	112♦♦	♦♦♦11	2212♦	11111	------
Literature - 2	♦6♦♦6	7777♦	♦♦♦77	66♦♦7	-----	7777♦
Literature - 3	♦♦♦2♦	666♦6	2♦♦22	-----	6266♦	6622♦
Literature - 4	♦8♦♦♦	44♦♦♦	-----	♦♦4♦8	84888	88444
Literature - 5	-----	♦5555	♦♦355	♦♦353	3♦33♦	33♦♦♦
Literature - 6	000♦0	-----	0♦999	0099♦	9♦♦♦♦	♦9♦♦♦
English	152♦5	32411	53♦♦♦	♦♦♦♦5	43422	-----
German	77997	98800	607♦6	-----	♦6♦♦♦	♦♦08♦
History and Philosophy - 1	5♦33♦	♦3343	-----	55♦44	♦♦♦4♦	4555♦
History and Philosophy - 2	9♦♦8♦	-----	♦88♦0	990♦♦	0♦009	908♦8
Math and Physics - 1	-----	5♦♦♦♦	45434	4453♦	5♦55♦	♦4333
Math and Physics - 2	♦9♦09	09998	♦♦♦08	88800	♦9♦♦0	-----
Math - 1	SSSS2	2S1AA	112♦♦	♦♦♦1♦	-----	22♦1♦
Math - 2	6S66S	SS♦AA	♦7S6♦	-----	7777♦	♦♦♦6♦
Natural sciences	33444	80022	-----	7378♦	♦8995	5♦9♦♦
Art	84518	-----	96643	37279	2♦♦♦♦	01♦05
Experimental Physics	2277S	S♦♦67	-----	1166♦	♦♦2SS	SS1♦♦
Gymnastic - 1	SSS♦♦	♦♦♦34	345SS	♦♦♦SS	S5♦♦♦	-----
Gymnastic - 2	SSS♦♦	♦♦♦89	890SS	♦♦♦SS	S0♦♦♦	-----
Religion	4S853	♦♦♦♦♦	721S♦	SS♦♦♦	-----	SS690

Fig. 1. *Representación de horario [7] Colorni (1992:Fig. 1)*

Las restricciones son manejadas por:

- Los operadores genéticos: de modo que el conjunto de horas asignadas a cada profesor no puede ser cambiada por la aplicación de operadores genéticos.
- Algoritmo de filtro: logra que la infactibilidad provocada por la aplicación de operadores genéticos sea total o parcialmente eliminada.
- Función Objetivo: la infactibilidad de individuos es limitada por la función objetivo ya que solo se seleccionan las soluciones que están mejor adaptadas.

Los autores distinguen entre dos tipos de restricciones: filas y columnas. Las restricciones de filas son incorporadas a los operadores genéticos y siempre satisfechas; las restricciones de columnas son manejadas por medio de un valor de adaptación y reparación genética. En la valoración de la función objetivo, los autores consideran una estructura de tres niveles:

Nivel 1. Condiciones de factibilidad

Nivel 2. Manejo de requerimientos didácticos, organizacionales y requerimientos de los profesores

Nivel 3. Preferencias expresadas por un profesor para su horario específico y su día de descanso.

En la etapa de reproducción del algoritmo implementado se utilizó un operador de reproducción que promueve a los individuos con un valor promedio por arriba del valor de adaptación.

El cruzamiento fue implementado considerando el valor local de adaptación de los posibles padres. La población se ordenaba de en forma decreciente respecto al valor de adaptación.

El operador de filtro toma una solución infactible y devuelve como salida una solución factible. El filtro utilizado se basó en la observación de que en cada columna (hora) de la matriz cada clase debe estar presente solo una vez.

El algoritmo fue implementado en Lenguaje C y se probó con la programación de horarios de una escuela en Milán. Al aplicar el algoritmo a programaciones de horarios de años anteriores observaron que obtenían una mejora en el acomodo de varias clases de modo que se satisfacían de mejor manera tanto requerimientos didácticos y como varias preferencias de los profesores. Los autores compararon su algoritmo genético con Recocido Simulado y Búsqueda Tabú y obtuvieron mejores programaciones de horarios que Recocido Simulado pero obtuvieron resultados ligeramente peores que Búsqueda Tabú.

3. Algoritmo Genético para el Problema de Programación de Cursos Semanales Universitarios

En [8] proponen un algoritmo genético basado en sectores para resolver un problema de calendarización de cursos semanales de una universidad No compara resultados con otros algoritmos Señala que fue aplicado a un caso real pero no indica variables utilizadas.

Definiendo el algoritmo

Los elementos necesarios para la programación de horarios son intervalos de tiempo, clases, profesores, lugares y materias. Para tomar en cuenta todos los elementos que son requeridos en la programación de horarios y hacer efectivo el proceso evolutivo, los autores diseñaron el esquema de los cromosomas como se muestra en la Figura 2.

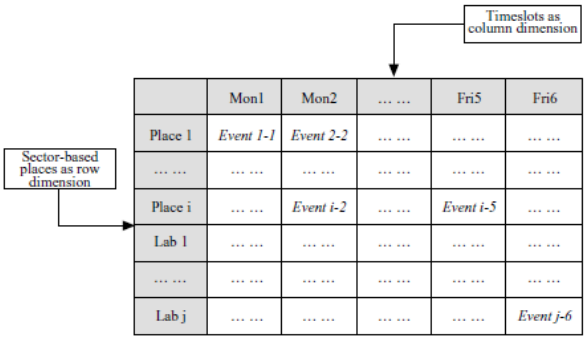

Fig. 2. *Representación de cromosomas para el problema de programación de cursos [8] Enzhe (2002:Fig. 2)*

Periodos de Tiempo: Los eventos toman lugar durante días laborables. En un horario de clases estándar, la dimensión de las columnas representa los días de la semana y la dimensión de las filas muestra los períodos de tiempo en que se dividen los días. En el diseño de cromosomas, los periodos de tiempo son puestos en la dimensión de las columnas Así, si los días a trabajar son 5 y los periodos de tiempo de cada día son 6, la tabla de programación de horarios tendría 30 columnas (5 días *6 periodos). Tal como se muestra en la Figura 1.

Lugares: Todos los lugares (salones de clase, laboratorios, estudios, etc.) se encuentran en la dimensión de las filas. Aquí se introduce el concepto de “sector” y su significado es el de un conjunto de lugares que tienen características comunes con el tipo de lugar, tamaño, etc. Los lugares se clasifican primero en dos grandes sectores: sector de laboratorios y sector de salones de clases. Después se ordenan por el tamaño.

Cada lugar tiene la siguiente forma:

Lugar -> nombre X tamaño X host X tipo

Eventos: Cada evento consiste de seis elementos:

Evento -> materia X clase X profesor X tamaño X host X tipo

Donde clase es el nombre del grupo que va a asistir determinado evento y host es el departamento anfitrión al que pertenece el evento. La relación entre lugares y eventos muestra que se ven involucradas, al menos, dos restricciones duras y una restricción suave, tal como se muestra en la Figura 3:

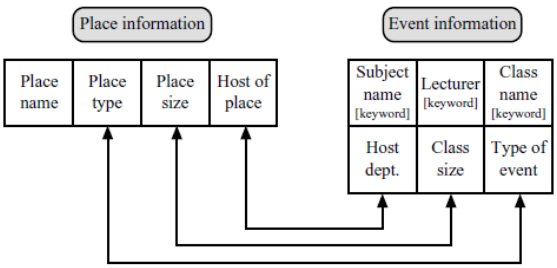

Fig. 3. *Relación entre lugares y eventos [8] Enzhe (2002:Fig. 3)*

Inicialización: Durante la inicialización se crea aleatoriamente una población de soluciones factibles sin observar su valor de adaptación. Las etapas que le siguen son:

Etapa 1: Pre-programación. Los requerimientos “debe ser” y las prioridades o restricciones duras son programados.

Etapa 2: Programación Subsecuente. Se programa el resto de los eventos denominados de segunda prioridad asignándolos de forma aleatoria de acuerdo al tipo de lugar que requieren.

El concepto de “sector” introducido por los autores le ayuda a reducir la complejidad del

proceso de inicialización puesto que en la asignación aleatoria es infactible la asignación de un evento a un sector que no le corresponde dada la naturaleza del evento.

Evaluación: El proceso de evaluación tiene que ver con restricciones suaves y cuenta con un mecanismo para penalizar la ocurrencia de éstas. La función de evaluación es la siguiente:

$$Eval(f) = 1/(1 + costo(f))$$

El objetivo es minimizar el costo y lograr que la función de evaluación sea lo más cercana posible a uno.

Operadores Genéticos.

Cruzamiento: Implementan un cruzamiento basado en sector al que denominan SBPMX y que tiene sus bases en el cruzamiento parcialmente mapeado (PMX). El espacio de cruzamiento es divido en sectores de acuerdo a los tipos de lugares y eventos, o del tamaño de los lugares y eventos.

El subconjunto de eventos derivados de un horario M y otro horario P se denominan S1 y S2 y podría tener una de las siguientes condiciones:

1.- Todos los eventos son clases
2.- Contener tanto clases como experimentos.
3.- Todos los eventos son experimentos.

El cruzamiento tiene lugar en los subconjuntos S1 y S2 y son generados los hijos C1 y C2 que, en caso de ser soluciones inaceptables, son sometidos a un proceso de reparación que permite verificar las restricciones duras.

Mutación: Las mutaciones se ejecutan con cada subrango de lugares. Los genes mutados son del mismo sector, lo cual reduce la posibilidad de crear cromosomas infactibles. Si existiera violación de restricciones duras se volvería a aplicar el proceso de reparación.

Experimentos y Resultados: Los autores hicieron pruebas de 3000 iteraciones con diferentes tamaño de población (25, 50 7 100), rangos de cruzamiento y mutación de 0.6 y 0.7 respectivamente. A través de la experimentación, una población de 100 obtuvo mejor comportamiento. Hacen notar los autores que su investigación requiere más experimentación con otros casos de estudio con más restricciones y diversidad de requerimientos.

4. Algoritmo Evolutivo para Resolver el Problema de Programación de Horarios.

En [9] describen un algoritmo evolutivo para el problema de calendarización de horarios aplicado al sistema de escuela italiano. La representación de la instancia es identificada por los siguientes elementos y relaciones:

• ***Salones:*** definidos por su tipo, capacidad y ubicación
• ***Materias:*** identificadas por el tipo de salón que requieren y por prioridades respecto a otras materias.
• ***Profesores:*** que cuentan con una cierta disponibilidad de horario para enseñar materias.

• ***Clases:*** grupos de estudiantes (incluyendo el caso de un solo estudiante) asignados a una determinada ubicación (edificio)

• ***Lecciones:*** denota la relación (t, s, c, l) donde t es profesor, s es materia, c es la clase que atiende la lección y l es la duración, en periodos.

Una solución candidata es considerada satisfactoria si cumple los requerimientos de los estudiantes y profesores, y además respeta la disponibilidad de los recursos. Esas propiedades se formalizan por medio de restricciones duras y suaves. Una solución será factible si satisface todas las restricciones duras y esa misma solución tendrá un grado menor o mayor de aceptabilidad dependiendo del nivel en que son satisfechas las restricciones suaves.

Se muestra un resumen de las restricciones tomadas en cuenta en [9] en la Figura 4:

hard constraints		soft constraints	
Constraint	Entity	Constraint	Entity
bounded rooms	rooms	room availability	rooms
physical availability	teachers	priority	subjects
contractual availability	teachers	subjective availability	teachers
co-location	classes	displacement	teachers
presence	classes	concentration	teachers
pre-assignment	lessons	distribution	lessons
sequentiality	lessons	decreasing burden	lessons
no gaps	classes	locality	lessons
non overlap	lessons	uniformity	lessons

Fig. 4. *Clasificación de restricciones para el sistema escolar italiano [9] Colagero (2001:Table. 1)*

El algoritmo que implementaron los autores es un algoritmo evolutivo elitista estándar excepto por el operador de perturbación, que alterna entre un operador de mutación inteligente y el operador de mejora. Un individuo en la población está representado por un vector de enteros que tienen el mismo tamaño que el número de clases a ser programadas. Cada entero contiene la codificación del periodo de inicio de la lección asociada.

El valor que toma cada gen depende de la información tal como el tamaño (en periodos) de lecciones relevantes, el número de días en la semana de escuela, el número de periodos de cada día, etc.

La población es inicializada colocando las lecciones una por una de forma aleatoria en periodos de inicio que no violen las restricciones, el proceso continúa hasta que todas las lecciones son asignadas. Los pesos asignados a cada restricción dura y suave permiten al algoritmo estar mejor sintonizado para una mejor búsqueda directa. El peso es elegido de acuerdo a la importancia relativa de las restricciones en las instancias tratadas.

Los individuos son elegidos para la reproducción utilizando selección por torneo. Este mecanismo de selección refuerza el elitismo al introducir al menos una copia del mejor individuo en la próxima generación, sin perturbación alguna. Los movimientos de perturbación que aplican los autores intentan no decrementar la adaptación del individuo (mutación inteligente) así como incrementar la adaptación y preservar la factibilidad de la solución (mejora).

Los experimentos realizados contemplaron tres casos del sistema escolar italiano nombrados como: “G. Marconi”, “G. Garducci” y “Leonardo da Vinci”. El resumen de las instancias se presenta en las Figuras 5 y 6.

Resources	Main Building	Branch Building	Remarks
classes	12	18	6 sections of 5 classes
teachers	35	27	3 Physical Education teachers work in both buildings
subjects	10	13	6 subjects are taught in both buildings
lessons	558	384	no lesson requires class grouping
labs	8	4	
gyms	2	1	the figures refer to the number classes which can use the gym at one time

Fig. 5. *Instancia G. Marconi [9] Colagero (2001:Table. 2)*

Resources	G. Carducci	Leonardo da Vinci
classes	33	35
teachers	60	71
subjects	13	23
lessons	924	1003
labs	1	3
gyms	2	2

Fig. 6. *Instancia G. Garducci y Leonardo da Vinci [9] Colagero (2001:Table. 3)*

Una ejecución típica con las instancias mencionadas tarda 40 minutos en promedio para obtener una solución factible y 5 horas y media en lograr una solución aceptable.

En el caso de la instancia “Leonardo da Vinci” la primera solución factible se encontró en la generación 2001, y la ejecución se detuvo en la generación 10,200 cuando el grado de satisfacción de restricciones suaves no mostró mayor mejora.

Los autores concluyen que la eficiencia y eficacia del algoritmo se basa en la hibridación con dos heurísticas, basadas en búsqueda local, implementadas por medio de los operadores de mutación inteligente y de mejora. Indican que la calidad de los resultados obtenidos fue notable e indican que un ejemplo de ello fue el caso de la instancia “Leonardo da Vinci” puesto que la programación de horarios obtenida fue sometida al comité de la escuela italiana y su implementación no requirió ningún ajuste manual.

5. Un Algoritmo Genético Híbrido para el Problema de Programación de Horarios Altamente Restringido.

En [10] presentan un algoritmo genético híbrido basado en un marco de trabajo de heurísticas para el problema de programación de horarios. El algoritmo combina el uso de representación directa del problema con operadores heurísticos de cruzamiento para asegurar que las restricciones fundamentales nunca son violadas.

El caso de estudio para los autores fue la asignación de horarios para exámenes de la Universidad de Nottingham que consiste en la programación de alrededor de ocho cientos exámenes en un periodo de dos semanas, proceso que se repite dos veces al año. Los conflictos en el problema que tratan los autores se refieren al número de exámenes que no pueden ser programados en el mismo periodo de tiempo porque tienen al menos un estudiante en común.

Los autores hacen mención de que el mayor problema que surge en todos los problemas complejos de asignación de horarios es la gran variedad y número de restricciones que algunas veces no son compatibles.

El marco de trabajo de la heurística aplicada en el problema de asignación de horarios se muestra en la Figura 7. Esta metodología combina dos algoritmos heurísticos. El Algoritmo 1 encuentra un conjunto de exámenes que no tienen conflictos y el Algoritmo 2 asigna esos exámenes

seleccionados a los salones. El proceso se repite para cada periodo hasta que todos los exámenes han sido programados en un periodo y en un salón sin conflictos o traslape.

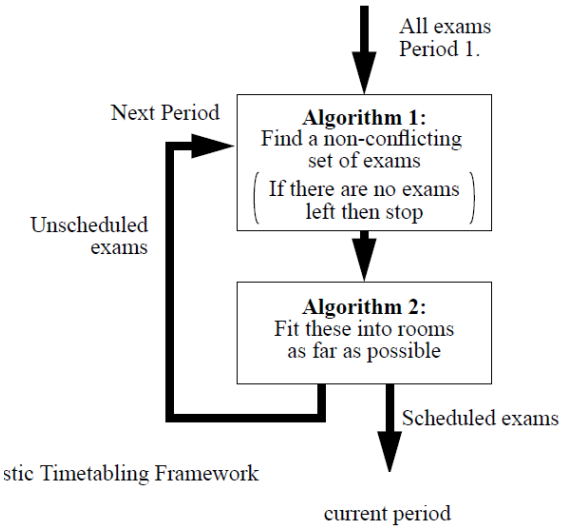

Fig. 7. *Un Marco de trabajo para el problema de asignación de horarios [10] Weare (1995:Fig. 1)*

El Marco de trabajo puede ser usado también como base para un operador híbrido de cruzamiento, con lo cual mantiene una población de horarios factibles y precisamente en ese dominio de horarios factibles es en el que el algoritmo evolutivo realiza sus búsquedas.

Creación de la población inicial: El algoritmo aleatorio utilizado para formar la población inicial se muestra en la figura 8.

For each population member:
Generate a random ordering of exams
Taking each exam in turn according to that ordering:
Find the first period in which the exam may be placed without conflict and so that the number of students does not go above a set maximum.
Place the exam in that period.

Fig. 8. *Algoritmo para generar la población inicial [10] Weare (1995)*

Los autores indican que con ese algoritmo obtienen grandes poblaciones de horarios factibles para la programación de exámenes y que no produce horarios de mínimos tamaños ni demasiado grandes que contengan periodos sin asignar o pocos periodos asignados.

Operadores de Cruzamiento: Los operadores inician buscando en el primer periodo. El operador toma los exámenes programados en ese periodo (en ambos padres) y utiliza un algoritmo para seleccionar otros exámenes de modo que no tengan conflictos con aquellos ya programados y que el límite de número de espacios no se haya excedido. Una vez completado, el cruzamiento pasa al segundo periodo y a los demás periodos hasta que todos los exámenes son ubicados.

Cálculo de adaptación y selección: La función de evaluación se construye tomando en cuenta factores relacionados. En el experimento que llevaron a cabo los autores se concentraron en dos requerimientos comunes:

• El tamaño de los horarios programados
• El número de conflictos entre exámenes en periodos adyacentes.

El Algoritmo Genético que trabajaron en [10] consideró una población de tamaño 200 exámenes, con la indicación de que aunque era una instancia típica de un problema de una facultad, era considerable el número de restricciones. La probabilidad de que dos exámenes tuvieran conflicto es ½ (equivalente a 10,000 restricciones). Una programación mínima de este conjunto sería de catorce

periodos. Los autores indican que en el problema observado con tantas restricciones, ni el operador aleatorio ni el operador de alto grado se comportó bien.

Los autores concluyen que los algoritmos evolutivos muestran ser prometedores en la programación de horarios en el área de educación, particularmente por su habilidad para considerar, y optimizar, la gran variedad de restricciones en las diferentes universidades. Consideran también que el haber utilizado operadores de cruzamiento híbridos les permitió trabajar con problemas con demasiadas restricciones.

6. Una Aplicación de Algoritmos Genéticos para el Problema de Programación de Horarios Escolares.

El estudio presentado en [11] evalúa el comportamiento de un algoritmo genético para el propósito de generar horarios escolares. Comparan el rendimiento del Algoritmo Genético aplicado a cinco problemas base (benchmarks) con el rendimiento de redes neuronales, Recocido Simulado, Búsqueda Tabú y Algoritmo Voraz aplicados a los mismos problemas base. Los autores indican que los resultados del Algoritmo Genético son comparables y mejoran los resultados obtenidos por los otros métodos.

El Algoritmo Genético comienza con la generación de una población inicial de horarios. En la representación utilizada cada elemento de la población consiste de dos genes, uno representa el horario de la clase y el otro, el horario de un profesor.

Cada gen es una matriz de dos dimensiones donde las filas corresponden a los periodos de tiempo y las columnas a los lugares. La intersección de una fila (periodo de tiempo) y una columna (lugar) contiene el profesor o la clase que son requeridos en el lugar en un periodo particular. Esta representación se muestra en la Figura 9.

Period	Venue1	Venue2	Venue3	Venue4
0	(C4,T2)	(C1,T1)	(C3,T3)	(C2,T4)
1	(C3,T4)	(C4,T3)	(C1,T1)	(C2,T2)
2	(C3,T4)	(C2,T3)	(C4,T2)	(C1,T1)
3	(C4,T3)	(C2,T2)	(C1,T4)	(C3,T1)
4	(C3,T1)	(C1,T2)	(C4,T3)	(C2,T4)
5	(C1,T4)	(C2,T3)	(C3,T2)	(C4,T1)
6	(C2,T4)	(C1,T3)	(C4,T1)	(C3,T2)
7	(C4,T2)	(C3,T1)	(C2,T4)	(C1,T3)
		...		

Fig. 9. *Representación de Horario en [11] Raghavjee (2008:Fig. 1)*

Así por ejemplo: La clase 4 (C4) y el profesor 2(T2) son requeridos en el lugar 1 durante el primer periodo de clases (0). Los autores indican que la representación utilizada asegura que no habrá conflictos de lugares y que también permite tratar por separados los requerimiento de las clases y de los profesores.

Para la construcción de horarios, el programa construye primero una lista de requerimientos del problema. Cada requerimiento es una tupla Clase-Profesor. Si el profesor P debe encontrarse con una clase C “n” veces a la semana, la tupla claseprofesor (C, P) ocurre “n” veces en la lista de requerimientos.

La población inicial no es generada de forma aleatoria, en lugar de ello, implementan un método de construcción secuencial (SCM) n veces para producir n horarios diferentes. El SCM ubica cada tupla de la lista de

requerimientos de acuerdo al grado de saturación, por ejemplo: el número de veces que la tupla puede ser ubicada en los periodos del horario sin que haya conflictos.

El algoritmo se detiene ya sea por haber encontrado la solución factible o porque se haya alcanzado el número máximo de generaciones. Si el problema tiene restricciones suaves al igual que restricciones duras solo se utiliza el criterio de terminación. La adaptación de un individuo es el número de restricciones violadas en el horario programado. En [11] las restricciones son que no debe haber conflictos de estudiantes, profesores o lugares. Además la adaptación del horario es la suma de los conflictos de estudiantes, profesores y lugares que tienen lugar en el horario generado. Para el proceso de selección, los autores incluyen una variante del método de selección por torneo. Esa variante inicia como el método estándar de torneo, eligiendo al azar *t* elementos de la población pero después se van seleccionando y comparando los individuos por pares para encontrar al individuo que mejor se adapta. Los autores reportan que esta variante en el método les dio mejores resultados que el método estándar de torneo.

El operador de mutación escoge al azar el hacer un cambio en el horario de una clase o en el del profesor. El cambio se realiza ubicando primero una fila en el horario que tenga un conflicto debido a la falta de la programación de una clase o de un profesor, luego se ubica otra fila que contenga la clase o el profesor faltantes en la misma columna en que se ubicó el primer conflicto. Así, el intercambio ocurre entre las filas.

Los experimentos que llevaron a cabo se realizaron con un conjunto de problemas base (benckmarks) que se muestran en la Figura 10 que de acuerdo a [12] aunque estos problemas pueden no ser tan grandes como una programación de horarios real, los problemas considerados tienen más restricciones y por ello son más difíciles. Las restricciones suaves no son consideradas en el estudio realizado. Los valores de los parámetros genéticos se muestran en la Figura 11.

Problem	No. of Classes	No. of Teachers	No. of Venues	No. of Periods	No. of Reqs
hdtt4	4	4	4	30	120
hdtt5	5	5	5	30	150
hdtt6	6	6	6	30	180
hdtt7	7	7	7	30	210
hdtt8	8	8	8	30	240

Fig. 10. *Problemas base (benchmarks) utilizados en el estudio [11] Raghavjee(2008:Table 1)*

Parameter	Value
No. of timetables created using SCM for each element of the initial population (*scm_size*)	1
GA Population Size	1001
Tournament Size	10
No. of Mutation Swaps	20
Maximum No. of Generations	50

Fig. 11. *Parámetros utilizados [11] Raghavjee (2008:Table. 2)*

Los autores publican que el Algoritmo Genético aplicado obtuvo horarios factibles para los cinco problemas base. El tiempo promedio que tomó al Algoritmo Genético para encontrar una solución es menos de un minuto para cuatro de los cinco problemas y u minuto para la instancia más difícil.

En la comparación con otros métodos encontraron que las variaciones de Redes Neuronales implementadas por Smith et. al.,

Recocido Simulado empleado por [13] y el Algoritmo Genético fueron los únicos en encontrar soluciones factibles para todos los problemas base. Mientras que el Algoritmo Voraz no encontró soluciones para ninguno de los problemas base.

7. Una aproximación al Problema de Asignación de Horarios con un Algoritmo Híbrido Evolutivo

En [14] describen un algoritmo para tratar el problema de asignación de horarios en el que para obtener una solución factible es necesario satisfacer un cierto número de restricciones. La calidad de la solución está medida en términos de un valor de penalización que representa el grado en que las restricciones suaves han sido satisfechas.

El estudio fue probado con instancias del problema planteadas en [15] que considera las siguientes restricciones duras:

- Ningún estudiante puede ser asignado a más de un curso al mismo tiempo.
- El salón debe satisfacer las necesidades requeridas por el curso
- El número de estudiantes que acuden al curso debe ser menor o igual a la capacidad del salón.
- Solo se programará un curso en un periodo de tiempo en cada salón.

También se consideran las restricciones suaves consideradas en [15]:

- Un estudiante tiene un curso programado en el último periodo del día
- Un estudiante tiene más de 2 cursos consecutivos
- Un estudiante tiene una sola clase en un día

El problema tiene como características:

- Un conjunto de N cursos e = {e1 ,…, eN}
- 45 periodos de tiempo
- Un conjunto R de salones
- U conjunto F de facilidades en los salones
- Un conjunto M de estudiantes

El objetivo del problema de programación de horarios es satisfacer las restricciones duras y minimizar la violación de restricciones suaves.

El método de hibridación empleado por los autores es un algoritmo evolutivo con búsqueda local, que en la literatura también se conoce como algoritmo memético, algoritmo genético híbrido o algoritmo con búsqueda local genética.

La técnica principal usada en el algoritmo evolutivo es el operador de mutación ligero seguido de un algoritmo aleatorio iterativo para mejora (búsqueda local). La mutación se llevó a cabo en el 20% de los cursos del 20% de los individuos seleccionados. El operador de cruzamiento no fue empleado en este estudio.

Representación de la Solución: Utilizan una representación directa para representar la solución ilustrada en la Figura 12. Cada gen contiene información acerca del periodo de tiempo y el salón de un curso en particular. Por ejemplo: El curso c1 está programado en el periodo 5 del salón 2.

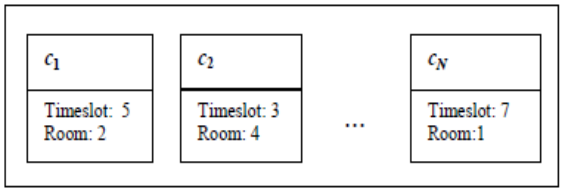

Fig. 12. *Representación de la solución en [14] Abdullah (2007:Fig. 1)*

La combinación de las Figuras 13 y 14 representan la vista del algoritmo que los autores emplearon en su experimentación. El algoritmo inicia creando una población inicial de 100 individuos. El proceso crea generaciones subsecuentes seleccionando primero el 20% de la población previa y aplicando la mutación al 20% de los cursos de cada individuo seleccionado. La búsqueda local se emplea enseguida de la mutación. Se conserva el mejor resultado obtenido luego de la búsqueda local. Las soluciones obtenidas luego de la búsqueda local son guardadas en una cola de población y pueden ser seleccionados para usarse en la próxima generación donde se aplica selección por ruleta.

El proceso termina después de un determinado número de generaciones (que para propósitos de experimentación se fijó en 30) o cuando se encuentra una programación horarios con costo cero.

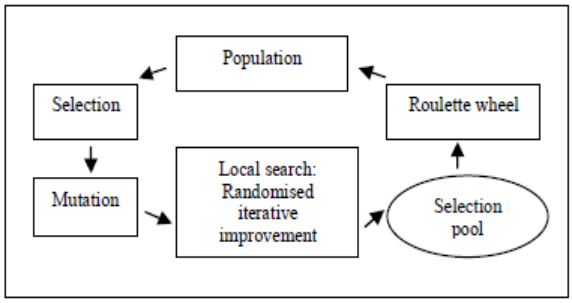

Fig. 13. *Esquema del algoritmo evolutivo híbrido [14] Abdullah (2007:Fig. 4)*

```
begin
Create population;
     do while (maximum number of generations
     or zero penalty timetable is not met)
          Select individuals to be mutated;
          Apply mutation;
          Apply local search;
          Select individual members of new
          populations;
     end do;
end
```

Fig. 14. *Pseudo código del algoritmo evolutivo híbrido [14] Abdullah (2007:Fig. 2)*

El algoritmo fue probado con un rango de instancias de problemas base (benchmarks) que se muestran en la figura 15.

Los autores presentan los mejores resultados obtenidos luego de 5 ejecuciones. Indican que en los experimentos realizaron 200,000 iteraciones que tardaron aproximadamente diez horas por cada conjunto de datos.

Category	*small*	*medium*	*large*
Number of courses, N	100	400	400
Number of rooms, R	5	10	10
Number of features, F	5	5	10
Number of students, M	80	200	400

Fig. 15. *Problemas base para la experimentación [14] Abdullah (2007:Table 1)*

La comparación del algoritmo evolutivo híbrido propuesto por los autores con otros estudios documentados en la literatura se muestran en la Figura 16. El término “x% Inf.” indica el porcentaje de ejecuciones en que el algoritmo falló en obtener soluciones factibles.

Datasets	Our approach, M1		M2		M3
	Best	Average	Best	Average	Best
small1	0	**0**	0	**0**	**0**
small2	0	**0**	0	**0**	**0**
small3	0	**0**	0	**0**	**0**
small4	0	**0**	0	**0**	**0**
small5	0	**0**	0	**0**	**0**
medium1	221	224.8	242	245	317
medium2	147	**150.6**	161	162.6	313
medium3	246	252	265	267.8	357
medium4	165	167.8	181	183.6	247
medium5	130	**135.4**	151	152.6	292
large	529	**552.4**	100% Inf.	100% Inf.	100% Inf.

Datasets	M4	M5	M6	M7	M8
	Average	Average	Best	Best	Best
small1	8	1	1	6	10
small2	11	3	2	7	9
small3	8	1	**0**	3	7
small4	7	1	1	3	17
small5	5	**0**	**0**	4	7
medium1	199	195	146	372	243
medium2	202.5	184	173	419	325
medium3	77.5%Inf.	**248**	267	359	249
medium4	177.5	**164.5**	169	348	285
medium5	100% Inf.	219.5	303	171	132
large	100% Inf.	851.5	80% Inf.	1068	1138

Fig. 16. *Comparación de Resultados [14] Abdullah (2007:Table 2)*

Donde se tiene:

M1: El algoritmo evolutivo híbrido de los autores [14]
M2: El algoritmo aleatorio iterativo de mejora con estructuras de vecindad [16]
M3: La búsqueda variable en vecindades [17]
M4: Búsqueda Local [15]
M5: Algoritmo Colonia de Hormigas [15]
M6: Híper heurística Búsqueda Tabú [18]
M7: Híper heurística basada en Grafos [19]
M8: Heurística Múltiple [20]

En la comparación se muestra que el algoritmo evolutivo híbrido propuesto por los autores obtuvo mejores resultados para la instancia grande y para todas las instancias medianas. Por lo que concluyen que la hibridación de operadores de mutación y búsqueda local genera un algoritmo más robusto que cuando se aplica solo búsqueda local.

8. Conclusiones

Los trabajos comparados en la presente revisión muestran que el uso de algoritmos evolutivos, aplicados al problema de programación de horarios, permite generar buenas soluciones.

Es de relevancia notar que los autores coinciden en que los buenos resultados, obtenidos en la implementación de los algoritmos evolutivos, estriban en la hibridación realizada en los operadores genéticos y/o la combinación del algoritmo evolutivo con búsqueda local. Tal hibridación de los algoritmos los hace más robustos y ello impacta en la obtención de soluciones de mejor calidad.

Algunos autores implementaron sus algoritmos considerando casos reales de programación de horarios y, en general, reportaron mejores resultados comparados con otros algoritmos, o bien, con la obtención manual de la programación de horarios. Si bien algunos autores aclaran que no se consideran todas las restricciones de los casos de estudio, la implementación de los algoritmos evolutivos muestran resultados prometedores para continuar usándolos en aproximaciones que incluyan más restricciones o, incluso, para su adaptación en casos de estudio diferentes de los analizados, con las adaptaciones necesarias a los requerimientos y restricciones propias de la(s) universidad(es) que se deseen considerar.

9. Referencias

[1] K. Banczyk; T. Boinski; H. Krawczyk Parallelisation of Genetic Algorithms for Solving University Timetabling Problems PAR ELEC. Pgs.:325 – 330 2006

[2] A. Saldaña Crovo; C. Oliva San Martín,; L. Pradenas Rojas. Modelos de programación entera para un Problema de Programación de Horarios para Universidades Ingeniare. Revista chilena de ingeniería, Vol. 15 No.3, pp.245-259 2007

[3] S. Even, A. Itai and A. Shamir On the Complexity of Timetable and Multicommodity Flow Problems SIAM Journal on Computing, Vol. 5, No.4 pags 691-703 1976.

[4] A. Díaz, F. Glover, H.M. Ghaziri. Optimización Heurística y Redes Neuronales. Madrid, Paraninfo. 1996

[5] A. Chaudhuri and K. De Fuzzy Genetic Heuristic for University Course Timetable Problem 2010

[6] D. Beasly, D.R. Bull, R. Martin An Overview of Genetic Algorithms: Part 1 Fundamentals. 1993

[7] A. Colorni, M. Dorigo and V. Maniezzo. A genetic algorithm to solve the timetable problem,Technical Report No. 90-060 Politecnico di Milano, Italy 1992

[8] Y. Enzhe and S. Ki-Seok. A genetic algorithm for a university weekly courses timetabling problem International Federation of Operational Research Societies, 2002.

[9] D. S. Calogero and A. G. B. Tettamanzi An Evolutionary Algorithm for Solving the School Time-Tabling Problem EvoWorkshop LNCS 2037. pgs. 452-462 2001

[10] E. Burke, R. Weare and D. Elliman A hybrid Genetic Algorithm for Highly Constrained Timetabling Problems Computer Science Technical Report No. NOTTCS-TR-1995-8 1995

[11] R. Raghavjee and N. Pillay An Application of Genetic Algorithms to the School Timetabling Problem SAICSIT 2008

[12] K. Smith, D. Abramson, and D. Hopfield Duke. Neural Networks for Timetabling: Formulations, Methods and Comparative Timetabling. Computers and Industrial Engineering, 4(2). 2003, 283-305. 2003

[13] M. Randall, D. Abramson, and C. Wild. A Meta-Heuristic Based Solver for Combinatorial Optimization Problems. Technical, Report, TR99-01, School of Information Technology, Bold University, Australia 1999

[14] S. Abdullah, E. K. Burke and B. McCollum. A Hybrid Evolutionary Approach to the University Course Timetabling Problem. IEEE Congress on Evolutionary Computation, ISBN: 1-4244-1340-0. Singapore, 25-28 September, pp 1764-1768 2007.

[15] K. Socha, J. Knowles, M. Samples, A Max-Min Ant System for the University Course Timetabling Problem. In: The Proceedings of the 3rd International Workshop on Ant Algorithms, ANTS 2002, Springer Lecture Notes in Computer Science Vol.2463, Springer-Verlag, 1-13. 2002

[16] S. Abdullah, E.K. Burke, B. McCollum, Using a Randomized Iterative Improvement Algorithm with Composite

Neighborhoods Structures for the University Course Timetabling Problem. Accepted for publication in the Metaheuristics International Conference (MIC'2005), Vienna, Austria, August 22nd-26th, post conference volume (eds. Karl Doerner, Michel Gendreau, Walter Gutjahr, Peter Greistorfer, Richard Hartl, Marc Reimann), to appear 2007.

[17] S. Abdullah, E.K. Burke, B. McCollum, An Investigation of a Variable Neighborhoods Search Approach for Course Timetabling. In: The Proceedings of the 2nd Multidisciplinary International Conference on Scheduling: Theory and Applications (MISTA 2005), New York, USA, July 18th-21st, 413-427, 2005.

[18] E.K. Burke, G. Kendall, E. Soubeiga, A Tabu-Search Hyperheuristic for Timetabling and Rostering. Journal of Heuristics, Volume 9, No.6, 451- 470 2003

[19] E.K. Burke, B. McCollum, A. Meisels, S. Petrovic, R. Qu, A Graph-Based Hyper Heuristic for Educational Timetabling Problems. European Journal of Operational Research 176(1), 177-192. 2007

[20] H. Asmuni, E.K. Burke, J.M. Garibaldi, Fuzzy Multiple Heuristic Ordering for Course Timetabling. In: The Proceedings of the 5th United Kingdom Workshop on Computational Intelligence (UKCI05), London, UK, September 5th-7th, 302-309, 2005.

Mireya Flores Pichardo.

Licenciada en informática egresada de la Facultad de contaduría, administración e informática (UAEM). Maestría en Ciencias en Ciencias de la Computación con especialidad en Ingeniería de Software. "Estudio empírico para la transformación gramatical de código en lenguaje COBOL hacia lenguaje Java". Centro Nacional de Investigación y Desarrollo Tecnológico (CENIDET).

Actualmente estudiante de Doctorado en Ingeniería y Ciencias Aplicadas con especialidad optimización combinatoria

Área: asignación de recursos - programación de cursos universitarios