Revista de Matemática: Teoría y Aplicaciones 2011 **18**(2) : 215–229
cimpa – ucr issn: 1409-2433

UN ALGORITMO GENÉTICO PARA UN PROBLEMA DE HORARIOS CON RESTRICCIONES ESPECIALES

A GENETIC ALGORITHM IN A SCHEDULE PROBLEM WITH SPECIAL CONSTRAINTS

Carlos Pérez de la Cruz*
Javier Ramírez Rodríguez†

Received: 23 Feb 2010; Revised: 23 Nov 2010;
Accepted: 1 Feb 2011

---

*Posgrado en Ingeniería en Sistemas, Universidad Nacional Autónoma de México, Circuito Exterior, Ciudad Universitaria, México D. F., México. E-Mail: larsoinde@hotmail.com

†Departamento de Sistemas, Universidad Autónoma Metropolitana – Azcapotzalco. Avenida San Pablo 180, Colonia Reynosa Tamaulipas, 02200 México D. F., México. E-Mail: jararo@correo.azc.uam.mx

Resumen

En Ramírez (2001) se introdujo el problema de coloración robusta generalizado (PCRG), el cual resuelve problemas de horarios que consideran restricciones del tipo: dos eventos no pueden realizarse a la misma hora y debe haber al menos d días entre dos eventos. El PCRG es una coloración robusta, en que dada una gráfica y un número fijo de colores, no necesariamente el número cromático, considera la distancia entre colores como penalización de las aristas complementarias. Se demostró que el problema es NP-Completo, por lo que es necesario utilizar métodos aproximados para encontrar buenas soluciones en un tiempo razonable. En este trabajo se presenta un híbrido de un algoritmo genético con uno de búsqueda local para casos de 30 a 120 horas por semana, se demuestra que para algunos la solución es óptima y en otros se encuentran soluciones muy prometedoras.

Palabras clave: horarios; restricciones especiales; heuristícas.

Abstract

Ramírez (2001) introduced the generalized robust coloring problem (GRCP), this problem lets solve timetabling problems which considers constraints such as: two events can not be assigned at the same time and there must be at least d days between two events. The GRCP deals with a robust coloring for a given graph with a fixed number of colors, not necessarily the chromatic number and considers the distance between colors as the penalization of complementary edges. It was shown that the problem is NP-complete, so it is necessary to use approximate methods to find good solutions in a reasonable time. This paper presents a hybrid of a genetic algorithm with a local search for cases of 30-120 hours per week; it is shown that for some cases the found solution is optimal and in other cases the solutions are very promising.

Keywords: timetabling; special constrains; heuristics.

Mathematics Subject Classification: 05C15, 90C59, 68T20.

1 Introducción

La asignación de horarios es un problema muy común en la planeación de actividades en todas las empresas, poder modelar y resolver estos problemas de una manera más precisa permite lograr mejoras significativas en los procesos. Existen modelos en los cuales se ven reflejadas restricciones tales como que dos o más procesos o actividades no puedan compartir un recurso, muchos de estos modelos no contemplan restricciones en cuanto a qué tan lejos en el tiempo deben estar estas actividades.

En Ramírez (2001) se introdujo el problema de coloración robusta generalizado (PCRG), el cual resuelve problemas de horarios que consideran restricciones del tipo: dos eventos no pueden realizarse a la misma hora y debe haber al menos d días entre dos eventos. Se demostró que el problema es NP- Completo, por lo que es necesario utilizar métodos aproximados para encontrar buenas soluciones, en Lara et al. (2010) se presenta un procedimiento glotón aleatorizado, GRASP por sus siglas en inglés.

En este trabajo se presenta un algoritmo genético para el PCRG que encuentra mejores soluciones para algunos casos reportados en Lara et al. (2010) y como el algoritmo genético y caso reportados en Ramírez (2001) encuentra la solución óptima. En la sección 2 se recuerdan algunas definiciones dadas en Ramírez (2001); en la Sección 3 se presenta un problema simplificado y se describe el algoritmo genético para el PCRG; en la Sección 4 se presentan resultados computacionales y se demuestra que la solución encontrada para algunos casos es óptima; finalmente en la sección 5 se presentan algunas conclusiones.

2 Definiciones

i. Problema de Coloración Robusta (PCR).
Dada una gráfica $G = (V, A)$ con $|V| = n$, $k > 0$, entero y una matriz de penalizaciones $P = \{p_{ij}, (i,j) \in \bar{A}\}$, el PCR busca construir una coloración válida C: $V \to \{1, 2, ..., k\}$, donde k no es necesariamente el mínimo, que cuente con el menor grado de rigidez, $R(C)$, entendiéndose por rigidez de la coloración C a la suma de las penalizaciones de las aristas complementarias cuyos extremos están igualmente coloreados, el objetivo es:

$$\min R(C) = \sum_{\{i,j\} \in \bar{A}, C(i)=C(j)} p_{ij}$$

$$s.a. \quad C(i) \neq C(j) \quad \forall \{i,j\} \in A.$$

Distintas penalizaciones de las aristas complementarias permiten obtener coloraciones válidas con diferentes propiedades.

ii. Dada una k-coloración C sobre una gráfica $G = (V, A)$, se define una k-distancia d como una distancia en el conjunto de colores $\{1, 2, ..., k\}$

iii. Problema de Coloración Robusta Generalizada (PCRG).
El PCRG busca construir una coloración válida con un número fijo de colores (no necesariamente el mínimo) que cuente con el menor número violaciones a restricciones del tipo "debe haber al menos d

días entre dos eventos de un mismo tipo". Para esto se define el $\bar{d}$-grado de rigidez generalizado de la k-coloración (C), siendo $\bar{d} \geq 0$, como la suma de las penalizaciones de las aristas complementarias cuyos extremos están pintados con colores que tienen una distancia menor o igual $\bar{d}$.

$$R^{\bar{d}}(C) = \sum_{\{i,j\} \in \bar{A}, d(C(i),C(j)) \leq \bar{d}} p_{ij}$$

$$s.a \quad C(i) \neq C(j) \quad \forall \{i,j\} \in A.$$

Al igual que el PCR, lo que se busca es encontrar la coloración con menor grado de rigidez generalizada.

3 Resolviendo el PCRG con un híbrido de algoritmo genético y búsqueda local

Un algoritmo genético básico consta de las siguientes etapas:

1. Generación de la población inicial.

   De manera aleatoria se crean posibles soluciones (cromosomas) que en conjunto formarán la población inicial.

2. Evaluación de los elementos de la población.

   Los elementos de la población son evaluados por medio de una función de aptitud.

3. Producir una nueva generación.

   Se aplican los operadores genéticos de cruce y mutación y se actualiza la población.

El algoritmo desarrollado en este trabajo considera una cuarta etapa en la cual se realiza una búsqueda local en las vecindades de los individuos de la población en cada generación.

El siguiente ejemplo es presentado en Ramírez (2001) y en Lara et al. (2010). Se trata de planificar los cursos de un diplomado que consta de 15 materias distribuidas en tres módulos. Los estudiantes se inscribirán al módulo que sea de su interés y de manera obligatoria llevarán todas las clases de las materias de dicho módulo:

Módulo	Materia	Número de clases a la semana
I	A,B,C	3
	K	1
II	D,E	3
	F,G	2
III	H,I,J	2
	L,M,N,O	1

A la administración le toca planificar las clases en los horarios y salones disponibles de modo que; clases de la misma materia o del mismo módulo no se impartan a la misma hora y además deberá de haber al menos 2 días entre clases de la misma materia. Para este ejemplo, en cada día se cuenta con tres horarios diferentes y se debe procurar no utilizar más de dos salones.

Los días hábiles en la semana son: lunes, martes, miércoles, jueves y viernes.

Representación de la solución

De acuerdo al ejemplo citado, y ya que cada materia tiene un número de clases que deben ser impartidas, a cada una de ellas se les puede etiquetar de la siguiente manera:

Materia	Clase	Etiqueta
A	1	1
A	2	2
A	3	3
B	1	4
B	2	5
B	3	6
⋮	⋮	⋮
O	1	30

A las horas disponibles también se les puede etiquetar de manera análoga:

Día	Hora	Etiqueta
lunes	1	1
lunes	2	2
lunes	3	3
martes	1	4
martes	2	5
martes	3	6
⋮	⋮	⋮
viernes	3	15

La solución se representa asignándole a cada uno de los vértices (clases) un color que representa la hora en que serán impartidas.

Un ejemplo de una solución es:

Clase con etiqueta #	Hora con etiqueta #
1	3
2	9
3	15
⋮	⋮
30	1

En esta solución la clase de la materia A etiquetada con 1 se impartirá en la hora etiquetada con 3, esto es en la tercera hora del lunes, la clase de la materia A etiquetada con 2 se impartirá en la hora etiquetada con 9, esto es en la tercera hora del miércoles, la clase de la materia A etiquetada con 3 se impartirá en la hora etiquetada con 15, que corresponde a la tercera hora del viernes etc.

Si a dos o más clases se les asigna impartirse a la misma hora, aun sin saber cual es ésta, diremos que pertenecen al mismo conjunto. Por tamaño de conjunto se entenderá en este caso por el número de clases que lo conforman.

Creación de la población inicial

Se crea una población inicial de tamaño 20, utilizando el siguiente procedimiento:

Se crea una solución que cumpla con la familia de restricciones del tipo "la clase x no puede darse a la misma hora que la clase y", de manera aleatoria. Esto dará como resultado conjuntos que contienen clases que se impartirán a la misma hora. La familia de restricciones del tipo: "debe haber d días entre clases de la misma materia" se corrige con el algoritmo.

Tomando el ejemplo de las 30 clases:

		$a1$ 1	$a2$ 2	$a3$ 3	$b1$ 4	$b2$ 5	$b3$ 6	$c1$ 7	$c2$ 8	$c3$ 9	$k1$ 10	$d1$ 11	$d2$ 12	$d3$ 13	$e1$ 14	$e2$ 15	$e3$ 16	$f1$ 17	$f2$ 18	$g1$ 19	$g2$ 20	$h1$ 21	$h2$ 22	$i1$ 23	$i2$ 24	$j1$ 25	$j2$ 26	$l1$ 27	$m1$ 28	$n1$ 29	$o1$ 30
$a1$	1																														
$a2$	2	1																													
$a3$	3	1	1																												
$b1$	4	1	1	1																											
$b2$	5	1	1	1	1																										
$b3$	6	1	1	1	1	1																									
$c1$	7	1	1	1	1	1	1																								
$c2$	8	1	1	1	1	1	1	1																							
$c3$	9	1	1	1	1	1	1	1	1																						
$k1$	10	1	1	1	1	1	1	1	1	1																					
$d1$	11	0	0	0	0	0	0	0	0	0	0																				
$d2$	12	0	0	0	0	0	0	0	0	0	0	1																			
$d3$	13	0	0	0	0	0	0	0	0	0	0	1	1																		
$e1$	14	0	0	0	0	0	0	0	0	0	0	1	1	1																	
$e2$	15	0	0	0	0	0	0	0	0	0	0	1	1	1	1																
$e3$	16	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1															
$f1$	17	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1														
$f2$	18	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1													
$g1$	19	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1												
$g2$	20	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1											
$h1$	21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0										
$h2$	22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1									
$i1$	23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1								
$i2$	24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1							
$j1$	25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1						
$j2$	26	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1					
$l1$	27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1				
$m1$	28	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1			
$n1$	29	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1		
$o1$	30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1

Los unos representan las restricciones de que dos conjuntos no se puedan unir, los ceros que si se puedan unir. Se elige al azar un cero, y se unen los conjuntos asociados en caso de que el conjunto resultante sea de tamaño menor o igual al máximo fijado, por ejemplo:

Supongamos que se elige al cero que relaciona la clase de la materia A etiquetada con 1 y la clase de la materia D etiquetada con 12. Se unirán dichos conjuntos y se agregaran las restricciones derivadas de esa unión.

		$a1,d2$ 1,12	$a2$ 2	$a3$ 3	$b1$ 4	$b2$ 5	$b3$ 6	$c1$ 7	$c2$ 8	$c3$ 9	$k1$ 10	$d1$ 11	$d3$ 13	$e1$ 14	$e2$ 15	$e3$ 16	$f1$ 17	$f2$ 18	$g1$ 19	$g2$ 20	$h1$ 21	$h2$ 22	$i1$ 23	$i2$ 24	$j1$ 25	$j2$ 26	$l1$ 27	$m1$ 28	$n1$ 29	$o1$ 30
$a1,d2$	1,12																														
$a2$	2	1																													
$a3$	3	1	1																												
$b1$	4	1	1	1																											
$b2$	5	1	1	1	1																										
$b3$	6	1	1	1	1	1																									
$c1$	7	1	1	1	1	1	1																								
$c2$	8	1	1	1	1	1	1	1																							
$c3$	9	1	1	1	1	1	1	1	1																						
$k1$	10	1	1	1	1	1	1	1	1	1																					
$d1$	11	1	0	0	0	0	0	0	0	0	0																				
$d3$	13	1	0	0	0	0	0	0	0	0	0	0																			
$e1$	14	1	0	0	0	0	0	0	0	0	0	0	1																		
$e2$	15	1	0	0	0	0	0	0	0	0	0	0	1	1																	
$e3$	16	1	0	0	0	0	0	0	0	0	0	0	1	1	1																
$f1$	17	1	0	0	0	0	0	0	0	0	0	0	1	1	1	1															
$f2$	18	1	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1														
$g1$	19	1	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1													
$g2$	20	1	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1												
$h1$	21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0											
$h2$	22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1										
$i1$	23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1									
$i2$	24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1								
$j1$	25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1							
$j2$	26	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1						
$l1$	27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1					
$m1$	28	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1				
$n1$	29	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1			
$o1$	30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1

Si ese conjunto ya tiene el número máximo de elementos que se permiten (para conjuntos que forman las soluciones de la población inicial este número máximo es el doble del número de clases por hora permitidas = 4), se añaden nuevas restricciones (unos) con el fin de evitar que se unan nuevos elementos a ese conjunto. En caso de que la unión de dos conjuntos de cómo resultado un conjunto de tamaño mayor al máximo fijado, se reparten al azar los elementos de este ultimo conjunto de tal manera que se creara un conjunto que cuente con el número máximo de elementos fijado y otro conjunto que contendrá los elementos sobrantes. Para el conjunto con el número máximo de elementos se añaden unos con el fin de que no se puedan añadir más elementos.

Sólo en caso de que este número máximo se hubiera fijado como 2, los unos que se tendrían que agregar serían los marcados con rojo en la figura de abajo:

		$a1$	$a2$	$a3$	$b1$	$b2$	$b3$	$c1$	$c2$	$c3$	$k1$	$d1$	$d3$	$e1$	$e2$	$e3$	$f1$	$f2$	$g1$	$g2$	$h1$	$h2$	$i1$	$i2$	$j1$	$j2$	$l1$	$m1$	$n1$	$o1$
		$d2$ 1 12	2	3	4	5	6	7	8	9	10	11	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
$a1,d2$	1,12																													
$a2$	2	1																												
$a3$	3	1	1																											
$b1$	4	1	1	1																										
$b2$	5	1	1	1	1																									
$b3$	6	1	1	1	1	1																								
$c1$	7	1	1	1	1	1	1																							
$c2$	8	1	1	1	1	1	1	1																						
$c3$	9	1	1	1	1	1	1	1	1																					
$k1$	10	1	1	1	1	1	1	1	1	1																				
$d1$	11	1	0	0	0	0	0	0	0	0	0																			
$d3$	13	1	0	0	0	0	0	0	0	0	0	1																		
$e1$	14	1	0	0	0	0	0	0	0	0	0	1	1																	
$e2$	15	1	0	0	0	0	0	0	0	0	0	1	1	1																
$e3$	16	1	0	0	0	0	0	0	0	0	0	1	1	1	1															
$f1$	17	1	0	0	0	0	0	0	0	0	0	1	1	1	1	1														
$f2$	18	1	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1													
$g1$	19	1	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1												
$g2$	20	1	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1											
$h1$	21	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0										
$h2$	22	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1									
$i1$	23	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1								
$i2$	24	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1							
$j1$	25	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1						
$j2$	26	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1					
$l1$	27	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1				
$m1$	28	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1			
$n1$	29	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1		
$o1$	30	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	

La elección de ceros, unión de conjuntos y actualización de unos se lleva a cabo hasta que ya no queda cero alguno por elegir.

Evaluación de la soluciones de la población inicial

A los conjuntos, se les asigna al azar un número (que representa una hora) de tal manera que cada hora disponible quede asignada a un conjunto. Se evalúa la solución de acuerdo al número de restricciones que son violadas (estas restricciones son del tipo x clase debe darse a m días de distancia o más de y clase).

Por ejemplo a un conjunto formado por alguna de las clases de la materia A (supongamos la etiquetada con 1) y alguna de las clases de la materia D (supongamos la etiquetada con 12), se les asigna al azar que se impartan a la quinta hora, esto es, las dos clases se impartirán a la segunda hora del martes: (1,5), (12,5)

A otro conjunto formado por alguna de las clases de la materia A (supongamos la etiquetada con 2) y la clase de la materia O etiquetada con 30 se les asigna al azar la novena hora, esto es, las dos clases de este conjunto se impartirán la tercera hora del miércoles: (2,9), (30,9).

Ya que clases de la misma materia deben de ser impartidas a dos días de distancia, el que se asigne impartir dos clases de la materia A en días consecutivos genera un violación al tipo de restricciones "x clase debe de darse a m d días de distancia o más de y clase". Por lo que una solución que presente esta asignación será penalizada en su evaluación, y así se irá penalizando a las soluciones por el número de restricciones que violen de este tipo. De esta forma la función de aptitud que evaluará qué tan buena es una solución será el grado de rigidez generalizado.

$$R^{\bar{d}}(C) = \sum_{\{i,j\}\in\bar{A}, d(C(i),C(j))\leq\bar{d}} p_{ij}.$$

Búsqueda local en la población inicial

Se evalúa la mejora de intercambiar la hora asignada de dos conjuntos cualquiera. Si hay mejora, se realiza el intercambio que resulta mejor evaluado y se actualiza la evaluación de la solución.

Se repite el procedimiento anterior hasta que no haya mejora posible con este k2 intercambio.

Operadores genéticos

Cruce

Para generar nuevos elementos en la población se ordenan las soluciones de manera aleatoria (para tamaño de población 20 hay 20! formas de hacerlo), y ya que están ordenadas las soluciones se toman en forma creciente, de dos en dos, y para cada 1 de las 10 parejas se realiza lo siguiente:

Fase 1 Clases que tuvieron el mismo color en cada una de las soluciones (individuos), aunque no sea precisamente el mismo color en las dos soluciones, con una probabilidad grande se creara un conjunto en que el estas clases pertenezcan al mismo.

$$P = MID * PG$$

Si en el conjunto al que pertenecen las clases en el padre mejor evaluado ninguno de sus elementos genero violación, $MID = 1$, en caso contrario $MID = 0.3$.

Si no ocurre que algunas clases tuvieran el mismo color en cada una de las soluciones (individuos), pero en el padre mejor evaluado estas clases aparecen en algún conjunto.

$$P = MID * (PG - .15)$$

En este caso, si en el conjunto al que pertenecen las clases en el padre mejor evaluado ninguno de sus elementos genero violación, $MID = 1$, en caso contrario $MID = 0.5$.

Fase 2 Supongamos que producto de la fase 1 se crearon 3 conjuntos, el primer conjunto formado por la clase de la materia A etiquetada con 1 y la clase de la materia D etiquetada con 12, el segundo conjunto formado por la clase de la materia B etiquetada con 6 y la clase de la materia J etiquetada con 25 y un tercer conjunto formado por la clase de la materia E etiquetada con 14 y por la clase de la materia J etiquetada con 26, podríamos representar esto como:

		$a1$ $d2$	$a2$	$a3$	$b1$	$b2$	$b3$ $j1$	$c1$	$c2$	$c3$	$k1$	$d1$	$d3$	$e1$ $j2$	$e2$	$e3$	$f1$	$f2$	$g1$	$g2$	$h1$	$h2$	$i1$	$i2$	$l1$	$m1$	$n1$	$o1$
		1	2	3	4	5	6	7	8	9	10	11	13	14	15	16	17	18	19	20	21	22	23	24	27	28	29	30
		12					25							26														
$a1$,$d2$	1,12																											
$a2$	2	1																										
$a3$	3	1	1																									
$b1$	4	1	1	1																								
$b2$	5	1	1	1	1																							
$b3$,$j1$	6,25	1	1	1	1	1																						
$c1$	7	1	1	1	1	1	1																					
$c2$	8	1	1	1	1	1	1	1																				
$c3$	9	1	1	1	1	1	1	1	1																			
$k1$	10	1	1	1	1	1	1	1	1	1																		
$d1$	11	1	0	0	0	0	0	0	0	0	0																	
$d3$	13	1	0	0	0	0	0	0	0	0	0	1																
$e1$,$j2$	14,26	1	0	0	0	0	0	0	0	0	0	1	1															
$e2$	15	1	0	0	0	0	0	0	0	0	0	1	1	1														
$e3$	16	1	0	0	0	0	0	0	0	0	0	1	1	1	1													
$f1$	17	1	0	0	0	0	0	0	0	0	0	1	1	1	1	1												
$f2$	18	1	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1											
$g1$	19	1	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1										
$g2$	20	1	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1									
$h1$	21	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0								
$h2$	22	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	1							
$i1$	23	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	1	1						
$i2$	24	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	1	1	1					
$l1$	27	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	1	1	1	1				
$m1$	28	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	1	1	1	1	1			
$n1$	29	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	1	1	1	1	1	1		
$o1$	30	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	1	1	1	1	1	1	1	

A partir de este punto se elige al azar un cero, se unen conjuntos y se actualizan unos con el mismo procedimiento como se crearon los elementos de la población inicial, con la diferencia que el máximo número de elementos que puede contener un conjunto varia cada vez que se elige un cero de la siguiente manera

$$n.m.e = (No.declasesporhorapermitidas) + INT(RND() * HOLG + .5).$$

HOLG es la holgura en el número clases por hora permitidas que se puede fijar para permitir que algunos conjuntos tengan más elementos de los permitidos. En la mayoría de los problemas sirvió $HOLG = 0$, pero para aquellos que no se encontró una solución con cero violaciones se puso $HOLG = 1$.

Después de que la solución está formada, a los conjuntos se les asigna al azar un número (que representa una hora) de tal manera que cada hora disponible quede asignada a un conjunto y se evalúa la solución tal y como se hizo anteriormente con los miembros de la población inicial.

Búsqueda local

Se evalúa la mejora de que dos conjuntos cualesquiera intercambien el número que les fue asignado (hora). Si hay mejora, el intercambio mejor evaluado se realiza y se actualiza la evaluación de la solución.

Se repite el procedimiento anterior hasta que no haya mejora posible con este k_2 intercambio.

Se evalúa la mejora de que dos clases cualquiera intercambien su conjunto contenedor, siempre y cuando no se violen restricciones del tipo "x clase no puede darse a la misma hora que y clase". Si hay mejora, el intercambio mejor evaluado se realiza y se actualiza la evaluación de la solución.

Se repite el procedimiento anterior hasta que no se encuentre mejora alguna con este intercambio.

Se evalúa la mejora de cambiar una clase cualquiera cuyo conjunto contenedor tuviese un tamaño mayor al "número de clases por hora permitidas", a un conjunto cuyo tamaño fuese menor al "número de clases por hora permitidas", siempre y cuando no se violen restricciones del tipo "x clase no puede darse a la misma hora que y clase". Si hay mejora o no se modifica la evaluación de individuo, el cambio mejor evaluado se realiza y se actualiza la evaluación de la solución.

Se repite el procedimiento anterior hasta que no se encuentre un cambio que al menos no modifique la evaluación del individuo.

Actualización de la siguiente generación

Si la solución generada es parecida al mejor de los padres, es igualmente evaluada y cada hora fue asignada de manera igual o más homogéneamente, se cambia en la población el padre mejor evaluado por el hijo.

Si no pasa lo anterior, la solución generada es igualmente evaluada que el peor de los padres y cada hora fue asignada de manera igual o más

homogéneamente se cambia en la población el padre peor evaluado por el hijo.

Si la solución generada es mejor evaluada que cualquiera de los padres, se cambia en la población el padre peor evaluado por el hijo.

Si no se logra nada de los tres párrafos anteriores permanecen en la población ambos padres y el hijo se desecha.

4 Cota del ejemplo citado y resultados computacionales

Cota del ejemplo citado

Considérese el problema con el que se ha venido trabajando (ver página 219).

Proposición 1 *No existe una solución en la que se asignen las horas a las clases de manera homogénea (para este ejemplo cada hora sería asignada dos veces) sin violar alguna de las restricciones.*

Demostración por contraejemplo:
Supongamos que existe una solución donde se violen cero restricciones y en la que las horas son asignadas máximo 2 veces. Existen 5 materias de tres clases que deben ser repartidas en lunes, miércoles y viernes para que no se viole ninguna restricción. Ya que cada hora sólo puede ser asignada a cuando más dos materias, sólo quedaría libre por asignar una hora del lunes, una hora del miércoles y una hora del viernes.

En el mejor de los casos podríamos elegir qué hora queda libre* en cada uno de esos tres días, por ejemplo:

Hora/día	lunes		martes	miércoles		jueves	viernes	
1	-	*		*	-		-	*
2	-	-		-	-		-	-
3	-	-		-	-		-	-

Por otro lado, al sólo haber 3 horas libres por asignar en lunes, miércoles y viernes, forzosamente tres(/) de las 5 materias de dos clases deberán de ser programadas para ser impartidas en martes y jueves. Utilizando el principio del palomar podemos inferir que al menos una de esas tres materias corresponde al módulo III.

Hora/día	lunes		martes	miércoles		jueves	viernes	
1	-	*	/	*	-	/	-	*
2	-	-	/	-	-	/	-	-
3	-	-	/	-	-	/	-	-

De esta forma lo podemos ver como que existen sólo tres casos, las tres materias anteriores pertenecen al módulo 3 o dos de las tres materias pertenecen al módulo 3 o sólo una de las tres materias pertenece al módulo 3.

Analizando el caso en que las tres materias pertenecen al módulo 3 quedarían por asignar del módulo 3 las cuatro materias de una sola clase. Ya que clases del mismo módulo no deben de impartirse a la misma hora sólo quedarían libres por asignar a estas cuatro clases la hora libre del lunes, la hora libre del miércoles y la hora libre del viernes. Por lo que en este caso no hay forma de lograr una asignación con cero restricciones violadas.

Hora/día	lunes		martes		miércoles		jueves		viernes	
1	-	*	H	no (L o M o N o O)	*	-	H	no (L o M o N o O)	-	*
2	-	-	I	no (L o M o N o O)	-	-	I	no (L o M o N o O)	-	-
3	-	-	J	no (L o M o N o O)	-	-	J	no (L o M o N o O)	-	-

Analizando el caso en que dos de las materias pertenecen al módulo 3, por ejemplo:

Hora/día	lunes		martes		miércoles		jueves		viernes	
1	-	*	F	*	*	-	F	*	-	*
2	-	-	H	no (L o M o N o O)	-	-	H	no (L o M o N o O)	-	-
3	-	-	I	no (L o M o N o O)	-	-	I	no (L o M o N o O)	-	-

Quedarían por asignar del módulo 3 las cuatro materias de una sola clase y una de las materias de dos clases; en total seis clases. Ya que clases del mismo módulo no deben de impartirse a la misma hora sólo quedarían libres por asignar a estas 6 clases la hora libre del lunes, la hora libre del miércoles, la hora libre del viernes, una de las horas del martes y una de las horas del jueves, en total 5 horas. Por lo que en este caso no hay forma de lograr una asignación con cero restricciones violadas.

Analizando el caso en que sólo una materia pertenece al módulo 3, por ejemplo:

Hora/día	lunes		martes		miércoles		jueves		viernes	
1	-	*	F	*	*	-	F	*	-	*
2	-	-	G	*	-	-	G	*	-	-
3	-	-	H	no (L o M o N o O)	-	-	H	no (L o M o N o O)	-	-

Quedarían por asignar del módulo 3 las cuatro materias de una sola clase y dos de las materias de dos clases; en total ocho clases. Ya que clases del mismo módulo no deben de impartirse a la misma hora sólo quedarían libres por asignar a estas 8 clases la hora libre del lunes, la hora libre del miércoles, la hora libre del viernes, dos de las horas del martes y dos de las horas del jueves, en total 7 horas. Por lo que en este caso no hay forma de lograr una asignación con cero restricciones violadas.

En los tres casos no hay forma de lograr una asignación con cero restricciones violadas, por lo que podemos concluir que no existe tal solución bajo las consideraciones antes planteadas.

Del mismo modo podemos concluir que una solución donde no se violen ninguno de los dos tipos de restricciones mencionadas en ente trabajo, debe permitir al menos que en una hora sean impartidas 3 materias. Y en caso de que se logre esto permitiendo que sólo en una hora se den 3 clases, esta solución será la óptima.

Resultados computacionales

La tabla 1 muestra los resultados computacionales para distintas instancias. Para una de las dos instancias (EDDC96441) en las que el GRASP no pudo encontrar una solución con cero violaciones en la que cada hora fuese asignada un número igual de veces, el Algoritmo Genético logro encontrarla; para la otra instancia (ED4) se demostró que aunque hay una clase fuera de lugar (fue asignada una hora a más clases de las permitidas) en las soluciones encontradas con GRASP y el Algoritmo Genético (AG), no hay mejor solución que pueda ser encontrada.

# vértices	Instancia	# colores	¬GRASP	¬AG	óptimo
30	ED4	15	1	1	sí
30	A42	15	0	0	sí
60	ECA864	20	0	0	sí
60	976532	20	0	0	sí
90	EDDC96441	30	1	0	sí
90	DCB875322	30	0	0	sí
120	EEDCCBA87644	30	0	0	sí

Tabla 1: Resultados computacionales para distintas instancias. La columna ¬GRASP indica el No. de clases "fuera de lugar" con GRASP y la columna ¬AG el No. de clases "fuera de lugar" con el algoritmo genético.

5 Conclusiones

Basados en los resultados anteriores se puede concluir que utilizar el Algoritmo Genético desarrollado en este trabajo es una buena opción para problemas de asignación de horarios con restricciones adicionales del tipo "debe haber al menos d días entre dos eventos".

En el desarrollo del algoritmo se manejó holgura en el número de colores (horas) y holgura en el número de clases permitidas por hora; con tal de obtener soluciones en las que no se violaran ninguno de los siguientes dos tipos de restricciones: dos eventos no pueden realizarse el mismo día y debe haber al menos d días entre dos eventos. Al parecer dejar que algunos miembros de la población tuviesen un número de colores mayor al establecido no aporto nada de información pero dejar en algunas soluciones holgura en el número de clases permitidas por hora aporto valiosa información que permitió llegar al óptimo más rápidamente para las instancias citadas.

Referencias

[1] Lara, P.; López, R.; Ramírez, J.; Yáñez, J. (2010) "SA model for timetabling problems with period spread constraints", *Journal of Operational Research Society* **173**: 1–6.

[2] Ramírez-Rodríguez, J. (2001) *Extensiones del Problema de Coloración de Grafos*. Tesis de Doctorado, Universidad Complutense de Madrid, Madrid.

[3] Grimaldi, R. (2009) *Matemáticas Discreta y Combinatoria, Una introducción con Aplicaciones*. Prentice Hall, México.

[4] Yáñez, J.; Ramírez, J. (2003) "The Robust Coloring Problem", *European Journal of Operational Research* **148**(3): 546–558.