MÉTODOS COMPUTACIONALES BIO- INSPIRADOS

PSO y GA aplicado a timetabling problem.

John E. Tapias Zarrazola
Dirigido por Daniel A. Soto Forero.
Ingeniería de sistemas – proyecto integrador
Universidad de Antioquia, 2015-II
Medellín, Colombia
jedisson.tapias@udea.edu.co

Carlos A. Peña Montenegro
Dirigido por Daniel A. Soto Forero.
Ingeniería de sistemas – Proyecto integrador
Universidad de Antioquia, 2015-II
Medellín, Colombia
alberto.pena@udea.edu.co

***Abstract*— this paper deals with mathematical optimization, Particle Swarm Optimization. In this paper is exposed the basics of the method and some information about heuristics and the difficulty of the algorithms. The method is applied to the timetabling problem, the PSO is simplified by the Standard Particle Swarm Optimization, and finally this article also implements the Genetic Algorithms to get better solutions.**

Keywords—SPSO, GA, timetabling, heuristics.

***Resumen*— En este artículo se presenta las bases del método de enjambre de partículas, por sus siglas en inglés PSO. Se implementa el algoritmo aplicado a la asignación de horarios, en la implementación se agrega algunas mejoras del método como la llamada SPSO y una aplicación particular que mezcla el método de algoritmos genéticos con las soluciones obtenidas en SPSO.**

Palabras clave—SPSO, GA, asignación de horarios, heurística.

I. INTRODUCCIÓN

En la búsqueda de soluciones a problemas cotidianos que lidien con máximos y mínimos, existen métodos de optimización para resolver los problemas que lidian con formas precisas para abordarlos como el método simplex y sus variaciones, Newton-Raphson y sus variantes, método de multiplicadores de Lagrange, entre otros. Además existen otras técnicas llamadas heurísticas que abordan la solución desde el ensayo y error, estás tienen más que ver con el azar y el análisis de comportamientos de fenómenos que con la iteración simple de los otros métodos. Es necesario utilizar este tipo de métodos cuando el universo de combinaciones para llegar a la solución es muy amplio, bien sea por la cantidad de variables que posea el problema o por la complejidad del mismo.

En la familia de métodos heurísticos, encontramos:

- Aquellos que son estrictamente heurísticos, que se encargan de resolver un problema en específico, son métodos hechos a la medida del problema.
- Los meta-heurísticos, son una abstracción o generalización de varios problemas, de modo que la solución puede utilizarse en diversas aplicaciones.
- Los hiper-heurísticos, pueden ser también una abstracción o generalización pero se caracterizan por utilizar varios métodos heurísticos para resolver el problema. [1]

Cualquier problema de aplicación que deseemos abordar, tiene además un nivel de complejidad que está clasificado en términos matemáticos, estos son los problemas tipo P o NP (nondeterministic polynomial time). “No determinísticos en tiempo polinomial” hace alusión a que la solución del problema en cuestión pueda resolverse en cierta cantidad de tiempo, esta cantidad de tiempo es determinada por una formula. A grosso modo los problemas tipo P son aquellos que conllevan procesos de multiplicación y combinatoria pero no son extensos, mientras que los NP son problemas de alta complejidad y/o muchas variables.[2].

II. EL MÉTODO PSO

A. *Generalidades*

Método desarrollado por James Kennedy y Russell Eberhart entre 1995 y 1998 inspirado en las ciencias del comportamiento, las neurociencias y etiología social. En este método se tiene en cuenta una población de partículas, que representan posiciones dentro de un espacio de búsqueda y poseen una velocidad que sirve para actualizar sus posiciones.

Las posiciones que pueden tomar las partículas en el espacio de búsqueda toman un valor en base a una función objetivo, la cual es el objeto a optimizar, valga la redundancia.

Las partículas guardan la información de la mejor posición que tuvieron cómo la posición hallada por toda la población.

Finalmente el algoritmo hace que haya una tendencia a la mejor solución. No es necesario que la función sea diferenciable. [3]–[5].

B. *Elementos del PSO*

- Función objetivo multivariable

$$f(x_1, x_2, x_3, x_4, x_5, x_6, \dots x_n)$$

Se seleccionan partículas en el espacio R^n.

- Cada partícula tiene velocidades en las distintas dimensiones

$$(v_1, v_2, v_3, v_4, v_5, v_6, \dots v_n)$$

La función objetivo está determinada por el problema que se desea abordar.

Estas son las ecuaciones utilizadas:

x_i	Posición inicial de la partícula.
x_{i+1}	Nueva posición.
v_i	Velocidad de la partículas.
v_{i+1}	Nuevo valor para la velocidad de la partícula.
$c_{1,2}$	Factores de aprendizaje.
r	Valores aleatorios.
$p_{i,d}$	Es el mejor valor de aptitud encontrado en la iteración actual.
$p_{g,d}$	Es el mejor valor de aptitud encontrado global hasta el momento.

$$v_{i+1} = v_i + c_1 r(p_{id} - x_{id}) + c_2 r(p_{gd} - x_{id})$$

$$X_{i+1} = x_i + x_{i+1}$$

Cabe resaltar que el concepto de velocidad tiene que ver con la distancia que tiene una partícula seleccionada respecto a la solución más cercana registrada.

C. *Seudo código*

[5] **Asignar** la función objetivo multivariable

Para cada partícula

{Inicializar partícula}

Haga hasta que se cumpla un máximo de iteraciones **o** criterio error mínimo

{ **Para cada** partícula

{ "valor calculado"**:=**Calcula el valor de la función objetivo

Si "valor calculado" es mejor que "el mejor valor de la partícula"

{"El mejor valor de la partícula":= valor calculado}

Si el "El mejor valor de la partícula" es "El mejor global"

{"El mejor global":="El mejor valor de la partícula"}

}

Para cada partícula

{ Calcular el valor de la velocidad de la partícula

Actualizar los datos con "mejor global" y su "velocidad" }}

III. MODIFICACIÓN DE PSO A SPSO

Standard Particle Swarm Optimization, por sus siglas en inglés [6]. Básicamente la modificación consiste en sustituir la función de velocidad y sus constantes, por unas constantes que hacen que el método sea más preciso, estas modificaciones son:

C1=2.05

C2=2.05

K=0.72984

$$V_{id}^{t+1} = K[V_{id}^t + c_1 \times Rand1 \times (P_{id} - X_{id}) + c_2 \times Rand2 \times (P_{gd} - X_{id})]$$

$$K = \frac{2}{2 - \varphi - \sqrt{\varphi^2 - 4\kappa}}, \ \varphi = c_1 + c_2, \ \varphi > 4$$

IV. ALGORITMOS GÉNETICOS

A. *Generalidades*

GA por sus siglas en inglés, Genetic Algorithms. Consiste en tomar una población de individuos y cruzarlos, eliminando los menos aptos y cruzando los mejores, con algunas mutaciones cada vez en cuando para darle un comportamiento genético al método. Finalmente las nuevas generaciones de individuos serán siempre las más aptas.[7]

V. EL PROBLEMA DE ASIGNACIÓN DE HORARIOS

El problema de asignación de horarios es conocido en inglés como Timetabling problem. Este consiste en asignar una cantidad de recursos en unos bloques de tiempo determinados.

Para el caso de un colegio o una universidad, los recursos a asignarse suelen ser estudiantes, profesores y materias. Este articulo solo trata con la asignación de profesores y materias a esos bloques de tiempo [8].

A continuación se presenta todos los elementos que se van a tener en cuenta para la solución de este problema, para ello es necesario llevar el problema a un nivel de generalización que aborde los situaciones más comunes.

En principio, pensemos en un horario sencillo de representar.

HORA - DÍA	Lunes	Martes	Miércoles	Jueves	Viernes	Sábado	Domingo
06:00 - 8:00							
08:00 - 10:00							
10:00 - 12:00							
12:00 - 14:00							
14:00 - 16:00							
16:00 - 18:00							
18:00 - 20:00							

Ilustración 1. Horario sin agrupar.

Para efectos de visualización de la información, esta forma clásica de representar un horario es bastante ilustrativa respecto a las necesidades que tienen las instituciones educativas o aplicaciones similares en la asignación de horas.

Las casillas en blanco representan LA ASIGNACIÓN dada en una hora y día determinado.

Una generalización como el horario propuesto en la *ilustración 1*, asume que todos los días se va a necesitar el mismo arreglo de horas, lo cual NO aplica para todos los casos, dado que los horarios se ajustan a distintas necesidades.

Para el caso particular de este ejemplo, deberíamos ajustar nuestra forma de trabajo a bloques de dos horas para todos los días. Por lo anterior se propone como **Paso # 1** definir los días que tengan el mismo horario y las horas que se vayan a necesitar en dicho día. Se utiliza la definición de TANDA dada por la Real Academia Española (RAE).

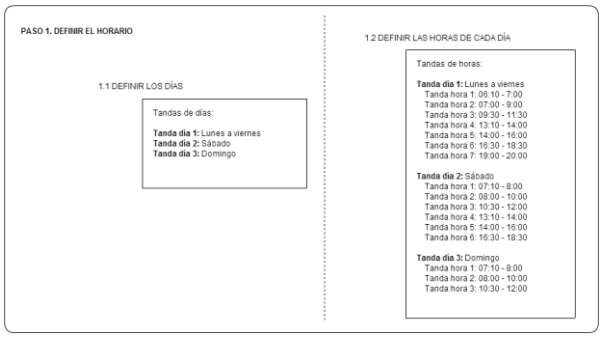

Ilustración 2. Paso 1, asignación de tandas.

De acuerdo a las necesidades de quién requiere el horario, la especificidad de éste pude variar de acuerdo a cada problema, es decir, se pueden definir tandas más generales como semanas, meses o años. Este trabajo le deja al usuario la libre elección y asignación de sus tandas.

Tanda de día 1: Lunes a viernes	Asignación Lunes a viernes	Tanda de día 2: Sábado	Asignación Sábado	Tanda de día 3: Domingo	Asignación Domingo
Tanda hora 1: 06:10 - 7:00		Tanda hora 1: 07:10 - 8:00		Tanda hora 1: 07:10 - 8:00	
Tanda hora 2: 07:00 - 9:00		Tanda hora 2: 08:00 - 10:00		Tanda hora 2: 08:00 - 10:00	
Tanda hora 3: 09:30 - 11:30		Tanda hora 3: 10:30 - 12:00		Tanda hora 3: 10:30 - 12:00	
Tanda hora 4: 13:10 - 14:00		Tanda hora 4: 13:10 - 14:00			
Tanda hora 5: 14:00 - 16:00		Tanda hora 5: 14:00 - 16:00			
Tanda hora 6: 16:30 - 18:30		Tanda hora 6: 16:30 - 18:30			
Tanda hora 7: 19:00 - 20:00					

Ilustración 3. Horario agrupado en tandas.

De esta forma hemos reducido una matriz de horas y días a un vector de hora-día, donde la casilla vacía sigue representando la ASIGNACIÓN dada por la aplicación que se le encuentre al problema.

		t	ASIGNACIÓN
Tanda de día 1: Lunes a viernes	Tanda hora 1: 06:10 - 7:00	1	
	Tanda hora 2: 07:00 - 9:00	2	
	Tanda hora 3: 09:30 - 11:30	3	
	Tanda hora 4: 13:10 - 14:00	4	
	Tanda hora 5: 14:00 - 16:00	5	
	Tanda hora 6: 16:30 - 18:30	6	
	Tanda hora 7: 19:00 - 20:00	7	
Tanda de día 2: Sábado	Tanda hora 1: 07:10 - 8:00	8	
	Tanda hora 2: 08:00 - 10:00	9	
	Tanda hora 3: 10:30 - 12:00	10	
	Tanda hora 4: 13:10 - 14:00	11	
	Tanda hora 5: 14:00 - 16:00	12	
	Tanda hora 6: 16:30 - 18:30	13	
Tanda de día 3: Domingo	Tanda hora 1: 07:10 - 8:00	14	
	Tanda hora 2: 08:00 - 10:00	15	
	Tanda hora 3: 10:30 - 12:00	16	

Ilustración 4. Horario agrupado en tandas representado como un vector.

De este modo podríamos definir dos vectores de tiempo, uno para la **tanda de días** y otro para la **tanda de horas**. Esta información puede almacenarse en arreglos, o vectores de vectores debido a que cada entrada hace referencia a un nuevo conjunto de tandas más específico, para el caso de este ejemplo, cada entrada del arreglo de días tiene que definir un arreglo nuevo de horas.

En este tipo de problema se debe tener en cuenta que existen tres tipos de restricciones, la mayoría de ellas depende del problema de aplicación, por expresarlas de manera general, estas son las restricciones:

- Fijas
- Duras
- Blandas

En este problema asumimos que los estudiantes ya están asignados a las aulas y las aulas son fijas, por lo tanto las aulas coinciden con los grupos de estudiantes. Las asignaciones que se hacen en este artículo son de materias con su respectivo profesor a los bloques de tiempo.

Después de definir los bloques de tiempo, los pasos siguientes para la asignación de horas:

- Obtener lista de aulas y su disponibilidad.
- Obtener lista de profesores con las materias que dicta cada uno de ellos y su disponibilidad.
- Definir las restricciones del problema.
- Verificar que la cantidad de profesores en cada bloque de tiempo (tanda hora + tanda día, en inglés timeslot) siempre haya mayor o igual cantidad de profesores que aulas en cada bloque de tiempo.
- Una vez verificada la condición anterior se procede a hacer la asignación de horas verificando las restricciones duras, blandas y fijas del problema.

VI. SPSO Y GA EN LA ASIGNACIÓN DE HORARIOS

A. Selección de la función objetivo.

La función objetivo para este problema en particular se tiene en cuenta en base a unas penalizaciones por cumplimiento o incumplimiento de restricciones. Por lo tanto buscamos minimizar la función.

Fitness=Constraint1(x)+Constraint2(x)+Constraint3(x)

Constraint1(x): entre más asignaciones tenga un profesor a un grupo es mayor la penalización.

Constraint2(x): a menos profesores asignados mayor la penalización.

Constrainte3(x): se cuenta la intensidad horaria asociada a cada profesor, se saca la moda de la intensidad horaria y por cada profesor se calcula el valor absoluto de la distancia a la moda, en cuanto a cantidad de horas y por último se hace la sumatoria de esas distancias y ese es el resultado de la penalización.

B. Tratamiento del problema.

[9] Para esta solución particular se trató el problema en el siguiente orden:

1. El algoritmo genera los bloques de tiempo (timeslots) que el usuario requiere.
2. Se genera una población de profesores y materias al azar, esto con el objeto de obtener situaciones aleatorias.
3. Se genera una matriz de disponibilidad aleatoria.
4. Se le solicita al usuario el número de enjambres que desea modelar, el número de partículas por enjambre, el número de iteraciones que se requiere y el número de rondas de apareamiento.
5. El algoritmo corre inicialmente con PSO haciendo las verificaciones de las restricciones en todo momento.
6. Posteriormente de cada enjambre elige los mejores individuos para cruzarlos y los peores para eliminarlos del enjambre y sustituirlos por los hijos obtenidos en los cruces genéticos. Las restricciones son verificadas en todo momento.
7. Cada cierta cantidad de iteraciones se aparean, estas se denominan en este artículo como **rondas de apareamiento**. Se escoge enjambres al azar, el número de enjambres debe ser par para que se puedan aparear.
8. A cada enjambre se extrae la mejor y peor partícula con el rendimiento histórico de las iteraciones, se cruzan las mejores partículas, la información genética está en la posición y ésta se divide a la mitad. De aquí sacamos dos hijos, el primer hijo tiene primera mitad del mejor individuo del enjambre padre y la segunda mitad, es la segunda mitad de la partícula del enjambre madre. El hijo dos tiene la primera mitad del enjambre madre y la segunda del individuo del enjambre padre. El hijo uno reemplaza la peor partícula del enjambre padre y el hijo dos reemplaza la peor partícula del enjambre madre.

El espacio se trata como una hiper-esfera, de modo que no se pierden partículas cuando se llega a los bordes.

El espacio está discretizado, una vez calculada la velocidad se actualiza la ecuación de posición pero esta posición se aproxima al entero más próximo.

VII. EXPERIMENTOS Y RESULTADOS

Experimento 1.

Este experimento se lleva a cabo solo utilizando SPSO

4	Bloques de tiempo.
4	Aulas o grupos.
8	Profesores.
5	Materias.
3	Materias máximo por profesor.
18	Códigos (materia & profesor)
8	Partículas por enjambre.
10	Enjambres.
10	Iteraciones.

Experimento 2

A SPSO se le incluye GA.

15	Bloques de tiempo.
12	Aulas o grupos.
24	Profesores.
10	Materias.
3	Materias máximo por profesor.
50	Códigos (materia & profesor)
8	Partículas por enjambre.
10	Enjambres.
100	Iteraciones.
5	Rondas de apareamiento

Experimento 3

Corrida 2 del experimento 2. Se repiten los mismos parámetros del experimento 2, excepto la cantidad de códigos.

46	Códigos (materia & profesor).

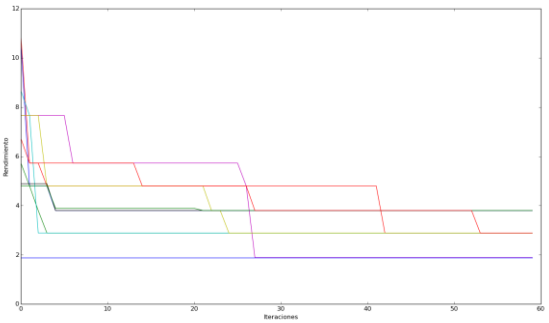

Ilustración 5. Resultados experimento 1.

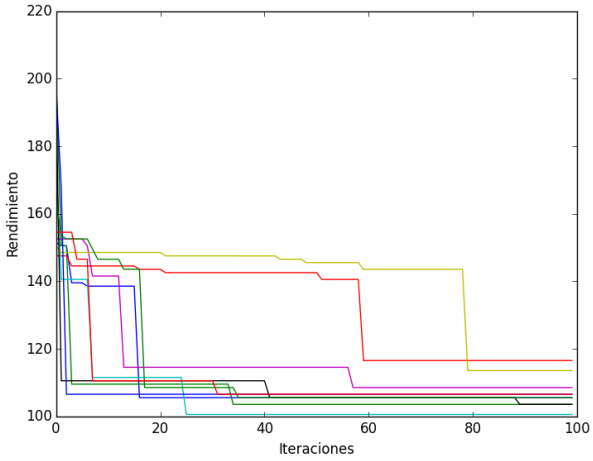

Ilustración 6. Resultados experimento 2.

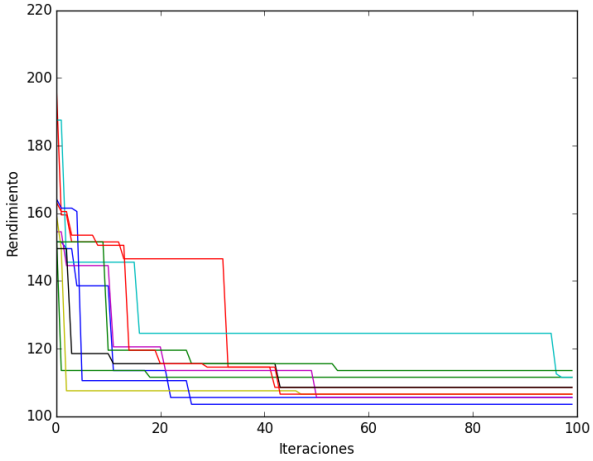

Ilustración 7. Resultados experimento 3.

VIII. ANÁLISIS DE RESULTADOS

Se buscaba minimizar la función objetivo. Entre menor era la penalidad, se cumplían más restricciones. Por lo tanto, es evidente que es mejor el resultado en los experimento dos y tres que en el uno, dado a que nuestro objetivo era minimizar la función. Las gráficas convergen a datos menores. Cabe resaltar que entre el experimento hecho sin GA y el que tiene GA, la cantidad de parámetros es distinta.

Espacio de búsqueda=Cantidad de códigos $^{\text{bloques*aula}}$

Debido a que el espacio de búsqueda de la solución es tan grande, es normal que cuando se hacen varias corridas con los mismos parámetros, se tienda a encontrar soluciones distintas, pues las partículas iniciales se generan al azar y pueden encontrarse en un lugar distinto del espacio, en todo caso son soluciones válidas, por ese motivo las corridas de los experimentos aunque tengan los mismos parámetros pueden resultar distintas, como se puede apreciar en las ilustraciones 6 y 7.

IX. CONCLUSIONES

- Por experimentación en el algoritmo, concluimos que alrededor de 60 iteraciones o más el método ya converge, después de esta cantidad el resultado tiende a soluciones similares.
- La experiencia muestra que utilizar las constantes que se usan en SPSO, hace tender el algoritmo a converger.
- Haber agregado GA a SPSO aumenta la precisión del método. Por tanto las soluciones son mejores y convergen más rápido.
- El SPSO tiende a converger, pero cada enjambre se optimiza como si fuera un problema aislado. Por otro lado el SPSO con GA permite tratar los conjuntos de enjambres como un solo problema que comparte información entre sus miembros.
- Los enjambres tienden a mostrar rendimientos asintóticos y se nota que cuando un enjambre mejora tiende a generar un efecto domino sobre el resto de enjambres, llevando a una mejora global de los enjambres.

REFERENCIAS BIBLIOGRÁFICAS

[1] I. Ribeiro Sucupira, “Um estudo empírico de métodos Hiper-Heurísticos,” 2005. [Online]. Available: http://www.ime.usp.br/~igorrs/seminarios/qualific.pdf.

[2] Hackerdashery, “P vs. NP and the Computational Complexity Zoo,” 2014. [Online]. Available: https://www.youtube.com/watch?v=YX40hbAHx3s.

[3] D. Karaboga and B. Basturk, “On the performance of artificial bee colony (ABC) algorithm,” *Appl. Soft Comput.*, vol. 8, no. 1, pp. 687–697, 2008.

[4] Swarmintelligence.org, “PSO tutorial,” 2006. [Online]. Available: http://www.swarmintelligence.org/tutorials.php.

[5] J. McCullock, “Particle Swarm Optimization,” 2012. [Online]. Available: http://www.mnemstudio.org/particle-swarm-introduction.htm.

[6] R. M. Chen and H. F. Shih, “Solving university course timetabling problems using constriction particle swarm optimization with local search.,” 2013.

[7] "Genetic algorithms," 2003. [Online]. Available: http://www.mnemstudio.org/genetic-algorithms-algorithm.htm.
[8] K. Socha, "Metaheruristics for the timetabling problem," Université Libre de Bruxelles, 2003.
[9] M. Y. Lin, K. S. Chin, K. L. Tsui, and T. C. Wong, "Genetic based discrete particle swarm optimization for Elderly Day Care Center timetabling," *Comput. Oper. Res.*, vol. 65, pp. 125–138, Jan. 2016.