Diseño de Horarios Escolares Utilizando Particiones y Programación Inteligente

[1]Lucero de Montserrat Ortiz-Aguilar*, [1]Juan Martín Carpio Valadez, [1]Héctor José Puga Soberanes, [2]Jorge Alberto Soria-Alcaraz y [1]Juan Adolfo Montesino Guerra
[1] *Instituto Tecnológico de León, León, Guanajuato, México*
[2] *2Universidad de Guanajuato, Guanajuato, Guanajuato, México*
*M09240932@itleon.edu.mx

Recibido: 28 de octubre, 2017
Aceptado: 10 de diciembre, 2017

RESUMEN

Diferentes tareas presentes en las organizaciones como escuelas, hospitales, centros de transporte, etc. se encuentran sujetas a un conjunto de restricciones, en la mayoría de los casos. A este tipo de problemas se les conoce como Satisfacción de Restricciones *CSP* por sus siglas en inglés (*Constraint Satisfaction Problems*). El problema de generar un diseño de horarios para las diferentes escuelas, como lo son las universidades es un problema sujeto a diferentes restricciones, como: estudiantes, profesores e inmueble de la institución. En este trabajo se muestra la aplicación de diseño de particiones apoyado con técnicas de inteligencia artificial y la metodología API-Carpio para mejorar el diseño de horarios del Instituto Tecnológico de León. Se hizo una comparación con análisis estadístico entre cinco Metaheurísticas dos trayectoriales y tres poblacionales. Finalmente se obtuvo que el algoritmo con mejor desempeño en general fue el de Búsqueda Local Iterada (*ILS*) y este consiguió soluciones aplicables en menor tiempo que las obtenidas por un experto humano.

Palabras claves: Partición, University Timetabling, Faculty Timetabling, Course Timetabling, Metaheurísticas

ABSTRACT

Different tasks present in organizations such as schools, hospitals, transport centers, etc. are subject to a set of restrictions, in most cases. These types of problems are known as CSP Constraint Satisfaction Problems. The problem of generating a timetabling design for different schools, such as universities, is a problem subject to different restrictions, such as students, teachers and buildings of the institution. This paper shows the application of partition design supported by an artificial intelligence algorithms and API-Carpio Methodology to improve the design of the timetable for Instituto Tecnológico de León. Finally, the algorithm with better performance was the Iterated Local Search (ILS) and this achieved solutions in less time than the gains for a human.

Palabras claves: Partition, University Timetabling, Faculty Timetabling, Course Timetabling, Metaheurísticas

1 INTRODUCCIÓN

El problema de calendarización de eventos es un proceso de toma de decisión, utilizado de manera frecuente en diversas organizaciones como lo son escuelas, hospitales, centros de transporte, etc. En él se trata de asignar un conjunto de tareas o eventos, durante un determinado período de tiempo y su fin es optimizar al menos uno o más objetivos. El objetivo de los CSP, es encontrar un arreglo de valores para un conjunto de variables sujetas a restricciones, donde los valores pueden ser asignados simultáneamente a un cierto subconjunto específico de variables (Jeavons et.al., 1998). Un ejemplo de CSP es la calendarización de horarios universitarios (*University Timetabling*).

La calendarización de actividades en una universidad tiene como propósito el garantizar que todos los estudiantes tomen sus asignaturas requeridas apegándose a los recursos que están disponibles. El conjunto de restricciones que debe contemplarse en el diseño de horarios involucra a los estudiantes, profesores e infraestructura. Dado que el conjunto de materias se agrupan en diferentes horarios, pero dos materias no pueden ir en el mismo horario, se aplicará un concepto de matemática que es el de partición. En una partición, la intersección de dos subconjuntos nos da como resultado el conjunto vacío y por lo tanto no existen dos objetos en común en esos subconjuntos, aplicando este concepto se busca generar un pre-diseño de horarios que ayude a mejorar la propuesta de soluciones actuales.

Tomando en cuenta que el problema de diseño de horarios depende del tipo de universidad, sistema de educación, plan educativo, alumnos, profesores, maestros y además se encuentra sujeto a un conjunto de restricciones duras y blandas, no existe un diseño de horarios que pueda ser aplicado de forma generalizada en todos los casos. Por lo anterior el diseño de horarios de forma automática mediante el uso de técnicas de inteligencia artificial, puede brindar la posibilidad de generar soluciones que permitan optimizar los recursos, que sean comparables y competitivas con las de un experto humano.

Esto puede permitir que en una institución se formalice y estandarice la metodología que se emplea en la elaboración de horarios, evitando así que los recursos como aulas y profesores sean desperdiciados; además cabe mencionar que el alumno puede tener la posibilidad de terminar su plan de estudios en tiempo, aprovechando sus horas de estancia dentro de la institución.

El teorema de *no-free luch* (K. H. Wolpert et.al., 1996) establece que no existe una Metaheurística capaz de generar una buena solución para cada problema posible. Debido a que hay una gran cantidad diferentes problemas, al menos de clase *NP*, así como diversos algoritmos aplicables en la búsqueda de una solución aceptable, en este trabajo se hace una comparativa del desempeño entre tres Metaheurísticas poblaciones: Algoritmo Genético, Memético y Sistema Inmune; y dos trayectoriales: *Iterated Local Search y Simulated Annealing*, para resolver un caso particular del problema *University Timetabling*. Las instancias utilizadas en este trabajo, pertenecen a datos reales del Instituto Tecnológico de León (ITL), donde la calendarización de horarios la elabora un experto humano. El proceso de elaboración de horarios en el ITL, por parte de un experto humano, puede tardar de cuatro a cinco semanas debido a la cantidad de restricciones a que está sujeto. Por lo anterior, se busca mediante los algoritmos Metaheurísticos generar soluciones de forma automática, que liberen al experto humano del diseño de horarios, en menor tiempo; que igualen e inclusive mejoren la calidad del de tal diseño de horarios, considerando cierto conjunto de restricciones. Para sustentar si existen diferencias significativas en el desempeño de dichas Metaheurísticas, se aplican pruebas estadísticas no paramétricas.

2 MARCO TEÓRICO

2.1 Partición

Una partición puede ser definida como (Rosen, 2004): Una ***partición*** de un **conjunto *S*** es una colección de subconjuntos **disjuntos dos a dos y no vacíos** de ***S*** tales que su unión es un todo S.

Es decir, la colección de subconjuntos A_i, $i \in I$(donde I es un conjunto de índices) forma un partición de S si, y sólo si, $A_i \neq \emptyset$ para $i \in I$, $A_i \cap A_j$ si $i \neq j$ y $U_{i \in I} A_i = S$ (F. Comellas, 2001).

2.2 El problema de Course Timetabling

El *University Timetabling* y el *High School Timetabling* son dos de los problemas más estudiados del *Educational Timetabling*. En este trabajo nos enfocaremos en resolver un caso particular del *University Timetabling*. Adriaen et.al., en 2006, agrupa en cinco grupos diferentes tipos de *University Timetabling*:

1. **Faculty Timetabling (FTT).** Es la asignación de profesores a eventos (materias) (grupos de estudiantes).
2. **Class-Teacher Timetabling (CTTT).** Es asignación de eventos (materias) con el menor conflicto temporal posible entre grupos de estudiantes.
3. **Course Timetabling (CTT).** Es la asignación de eventos (materias) con el menor conflicto temporal posible entre estudiantes individuales.
4. **Examination Timetabling (ETT).** Es la asignación de exámenes a los estudiantes, de tal forma que el alumno no aplique dos pruebas al mismo tiempo.

5. **Classroom assignment (CATT).** Después de asignar los eventos (materias) a los profesores, se asignan *class-teacher* a los salones.

El *Course Timetabling* puede ser definido como el proceso de asignar clases a recursos como lo son de tiempo (*timeslots*), espacio (salones) y profesores (personal), mientras se cumpla un conjunto restricciones (LAI et. al., 2008).

Existen dos tipos de restricción (McCollum et.al., 2011):

- **Duras**. Es la restricción que absolutamente no puede ser violada.
- **Blandas.** El conjunto de restricciones que se prefieren cumplir, pero no genera un costo mayor el que no se satisfagan.

En este trabajo se enfoca a generar soluciones aceptables al problema de *Course Timetabling*. Las instancias utilizadas pertenecen a datos reales del Instituto Tecnológico de León (ITL), donde la calendarización de horarios la elabora un experto y se busca mediante técnicas de inteligencia artificial generar soluciones factibles en mejor tiempo.

2.3 Algoritmos Metaheurísticos

2.3.1 Algoritmo Genético

Los algoritmos Genéticos fueron desarrollados por J. Hollan en los 70s, con el fin de entender el proceso adaptativo del sistema natural, para luego aplicarlo en la optimización y Machine Learning en los 1980s (Goldberg, 1989). Los algoritmos Genéticos son métodos adaptativos, generalmente usados en problemas de búsqueda y optimización de parámetros, basados en la reproducción sexual y en el principio de supervivencia del más apto. En (Yang, 2010) definen el algoritmo que corresponde a un Genético Simple.

En (Talbi, 2009) menciona que el Genético es una de las estrategias más usadas y en este trabajo se empleó la selección por Ruleta pesada (*Roulette Wheel Selection*.). Esta consiste en asignar a cada individuo una parte proporcional de probabilidad en relación al *fitness*. Teniendo f_i el fitness del individuo p_i en la población P y La fórmula de probabilidad se puede consultar en (ABDOUN et.al., 2011). La cruza es el proceso probabilístico que cambia la información entre dos cromosomas (padres) para generar dos cromosomas hijos (Mahiba et.al., 2012) y la empleada en este trabajo es la cruza a un punto descrita en (Talbi, 2009), donde el sitio de cruza k es seleccionado aleatoriamente y los dos hijos son creados intercambiando los segmentos de los padres. Y para el operador de muta tenemos que será a un punto donde seleccionaremos aleatoriamente una posición k y cambiaremos el valor de esa posición por otro que se encuentre en la LPH (Talbi, 2009).

La representación de las soluciones candidatas se muestra en la figura 1, donde en cada posición representan un evento o materia y la columna representa el *timeslot* asignado de la LPH.

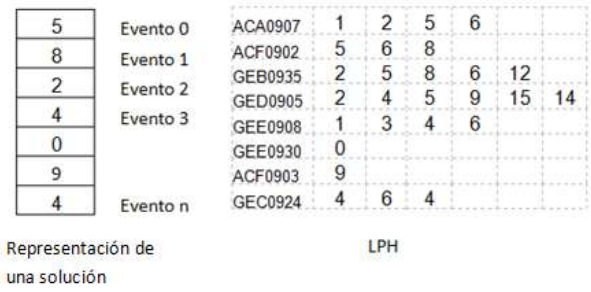

Figura 1: Representación de las soluciones candidatas

2.3.2 Algoritmos Meméticos

En 1976 Dawkins diseñó el concepto de meme, el cual a diferencia del gen puede ser modificado por su portador. Supuso que existe un progreso como un gen del algoritmo genético que es transferido a la próxima generación, es decir, las características obtenidas se transfieren de una generación anterior a una siguiente, junto con los cambios de población. Moscato en (Lü et.al., 2010) donde describe lo que es el algoritmo Memético.

Un meme es una unidad de datos que puede recrearse a sí misma. Estas unidades se transmiten entre las personas y cualquier otra, que se pueden ajustar con ella, y es capaz de salvar la unidad de datos; mientras que un gen se

mantiene sin cambios durante la transmisión (Araujo et.al., 2010). Los componentes de un Algoritmo Memético son (Talbi, 2009): Algoritmo Genético y Búsqueda Local.

La búsqueda local es una modificación que se puede llegar a hacer toda la población de individuos con la que trabaja el algoritmo. En este se realiza una copia de cada individuo, donde esta copia es alterada de alguna forma; si la copia de un individuo en específico es mejor que la original, la copia remplaza al individuo original.

2.3.3 Sistema Inmune

Estos se basan en imitar el comportamiento del sistema inmunológico humano, el cual se encarga de proteger al cuerpo de los patógenos externos e internos y su tarea principal es reconocer las células en el cuerpo clasificarlas como propias y no propias. Los algoritmos de Sistema Inmune artificial han sido aplicados con éxito en diversos problemas de optimización (Azuaje et.al., 2003).

Básicamente el proceso del algoritmo de Sistema Inmune Artificial consiste en generar aleatoriamente una población de soluciones candidatas; después seleccionamos un porcentaje de los mejores individuos, los cuales son clonados, luego a estos individuos se les aplica una hipermuta y finalmente continuamos hasta llegar a nuestra solución objetivo, pero para evitar que nuestra población crezca sin medida se pone una poda la cual nos permitirá regresar al tamaño inicial de la población (Villalobos et.al., 2004). En el algoritmo del sistema Inmune artificial se puede consultar en (Herrera et.al., 2004).

2.3.4 Simulated Annealing

El algoritmo de Recocido Simulado (*Simulated Annealing Algorithm* - SAA) pertenece a una clase de Algoritmos de búsqueda local (*Local Search Algorithms* - LSA) comúnmente llamada Algoritmos de Umbral (*Threshold Algorithm* - TA) (Talbi, 2009).

El recocido simulado es llamado así debido a su analogía con el proceso de la física de recocido de sólidos, en el cual un sólido cristalino se calienta y después se permite enfriarlo muy lentamente hasta que alcanza su configuración de enrejado cristalina en su mayoría, permitiendo estar libre de formaciones anisotrópicas, y por ende mayor pureza cristalina (Fowlesi, 1987). Si el tiempo de enfriamiento es suficientemente lente, la configuración final resulta un sólido con integridad estructural superior (Michel et.al, 2010). En el algoritmo de esta Metaheurística se puede consultar en (Michel et.al, 2010).

2.3.5 Iterated Local Search

El término *Iterated Local Search* (ILS) fue propuesta en (Helena et.al, 2010). La esencia de la Metaheurística de búsqueda local iterada está en que dada una solución inicial: va construyendo una secuencia de soluciones generadas por una heurística embebida, dando lugar a soluciones mejores que si sólo se repitieran corridas aleatorias aisladas de la heurística (Talbi, 2009). En (Talbi, 2009) definen los algoritmos de búsqueda local iterada. En este algoritmo, la fase de perturbación aplica un simple operador movimiento aplicado en la solución. Este operador (*Simple Random Perturbation*) simplemente selecciona uniformemente una variable y la sustituye por otra variable en el rango, seleccionándola uniformemente.

3 METODOLOGÍA.

En este trabajo de se busca hacer un diseño de horarios para estudiantes y profesores utilizando la metodología API-Carpio (Carpio, 2006) y la propuesta por Soria (Soria-Alcaraz et al, 2013).

3.1 Metodología API-Carpio.

La metodología consiste fundamentalmente en considerar los siguientes actores de la problemática: estudiantes, profesores, institución y datos históricos (Carpio, 2006). Así como las interrelaciones que existen entre ellos. De cada uno de estos actores, tomamos en cuenta los siguientes aspectos:

- **Estudiantes**: las materias con las que cuenta tienen una seriación de materias, pre-requisitos y un seguimiento curricular.
- **Profesores**: se les pide que propongan un mínimo de *k* materias diferentes que deseen o puedan impartir de acuerdo con su perfil académico, indicando el orden de prioridad en que desean impartir.

Además su disponibilidad de horario para impartir clases, horas de nombramiento, nombramientos administrativos y otras actividades relacionadas con la docencia.

La metodología API-Carpio (Carpio, 2006) describe el proceso de calendarización de horarios educativos como:

$$f(x) = FA(x) + FP(x) + FI(x) \quad (1)$$

Dónde:

$FA(x) =$ Número de estudiantes en conflicto dentro del horario x, (CTT).

$FP(x) =$ Número de profesores en conflicto dentro del horario x, (FTT).

$FI(x) =$ Número de aulas y laboratorios en conflicto dentro del horario x,(CATT).

En este trabajo se restringe a tomar solamente hasta $FA(x)$ la cual está definida como:

$$FA = \sum_{j=1}^{k} FA_{V_j} \ (2)$$

Dónde:

$$FA_{V_j} = \sum_{s=1}^{\left(M_{V_j}\right)-1} \sum_{l=1}^{\left(M_{V_i}\right)-s} (A_{j,s} \wedge A_{j,s+l}) \quad (3)$$

Teniendo:

$FA_{V_j} =$ Número de estudiantes en conflicto dentro del vector V_j.

$V_j =$ Es un vector de tiempo que contiene diferentes materias.

$A_{j,s} \wedge A_{j,s+l} =$ Número de estudiantes que demandan la inscripción simultánea de las materias $M_{j,s} \wedge M_{j,s+1}$.

3.2 La metodología del diseño

La metodología del diseño propuesta por (Soria-Alcaraz et.al., 2013) permite que diferentes políticas de la calendarización de tareas y listas de restricciones sean modelados mediante la conversión de todas las restricciones de tiempo y espacio en un simple tipo de restricción: conflictos de estudiantes. En esta se proponen estructuras como lo son la matriz MMA, lista LPH, lista LPA y lista LPS. Las primeras tres representan las restricciones duras y la última representa las restricciones blandas. En este trabajo se utilizaran dos de las cuatro estructuras las cuales son: matriz MMA y lista LPH. En (Soria-Alcaraz et.al., 2013) nos definen que sus estructuras son las siguientes:

- **Matriz MMA**: Contiene el número de conflictos (entre estudiantes) posibles si dos lecciones son asignadas en el mismo espacio de tiempo. La figura 1a muestra un ejemplo de matriz MMA.
- **Lista LPH:** Esta lista nos da información acerca de cada lección (clase, evento o materia) en que posible espacio de tiempo puede ser asignada. La figura 1b muestra un ejemplo de LPH.

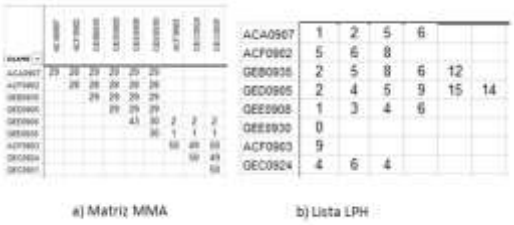

Fig. 2**.** Ejemplo de MMA y LPH

Con la información del alumno se construye la matriz MMA, la cual contiene la posible demanda de los estudiantes por materia. La matriz se elabora por cada carrera en un semestre determinado y la cantidad mínima de grupos por materia que cubren las necesidades de todos los estudiantes involucrados está dada por el número que contiene en su diagonal.

A partir de la matriz MMA, se construyen particiones de acuerdo con el número de horarios disponibles que oferta la institución, pero tomando en cuenta la restricción dura: "dos materias que son atendidas por un mismo

maestro, no deben estar en la misma partición". Una partición se construye por una colección de materias que en su intersección en la matriz MMA, tienen una cantidad de estudiantes demandantes simultáneamente, menor o igual a un umbral (Carpio, 2006).

4 RESULTADOS Y DISCUSION

En esta experimentación se utilizaron instancias reales de las diferentes carreras como lo son: IXX, IGE, LIA, etc. Además de que en este experimento no se considera la restricción dura de que dos materias del mismo semestre, no pueden ir en la misma *partición*, es decir, todas las materias tienen una lista posible de horarios y en dicha lista puede tener inmersa alguna restricción dura, pero no se cuida que dos materias del mismo semestre queden en el mismo número de *partición*.
Las instancias reales provienen de datos reales del ITL, de las carreras de Licenciatura en administración, Ingeniería en Gestión Empresarial e Ingeniería en Industrial. Cada instancia está compuesta de cuatro archivos:

- **Matriz MMA**. Este archivo contiene la matriz de conflictos.
- **Lista nombre.LPH**. Este archivo contiene la lista posible de horarios de cada materia.
- **Lista LMS**. Este archivo contiene el semestre de cada una de las materias.
- Solución propuesta por el experto.

La configuración utilizada en los algoritmos Genético, Memético, SI, ILS y SA se muestran en la tabla 1, donde tenemos que las llamadas a función fueron 300,000, la población inicial para cada uno fue la misma. El criterio de paro de los algoritmos fue el de llamadas a función. La tabla 2 muestra los resultados obtenidos (conflictos) de acuerdo a la función objetivo mostrada en 1, donde se puede observar la media, la mediana y la deviación estándar de las ocho instancias evaluadas con el AG, AM, SI, ILS y SA.

Se puede observar que las desviaciones estándar del Algoritmo Genético tiene en algunas instancias variaciones significativamente mayor al de los demás algoritmos, ejemplo en las instancias: 12, 15, 16 y 19. Para los demás algoritmos poblacionales sus desviaciones estándar tienden a tener valores similares.

Los algoritmos de tipo trayectorial que son el ILS y SA tienen desviaciones estándar con valores para las diferentes instancias de un rango desde cero hasta 12.6, en algunas instancias no hay gran diferencia entre algoritmos, como son el caso de la instancia 9, 10 y 21. Debidos a que se busca encontrar algoritmos que tengan soluciones aceptables con una baja desviación estándar, lo algoritmos Memético, Sistema Inmune, *Iterated Local Search* y *SA* fueron seleccionados, lo cual es indicativo de buena reproducibilidad de resultados por parte de los algoritmos Metaheurísticos utilizados.

Se aplicó el test de Friedman (Derrac et.al.,2011), a los algoritmos Genético, Memético, SI, *Iterated Local Search* y *SA*, donde el objetivo es determinar si existe diferencia entre las distribuciones de las medianas. Para esto se plantearon las siguientes hipótesis:

H_0: Existen diferencias entre las distribuciones de las medianas de los algoritmos

H_a: No existen diferencias entre las distribuciones medianas de los algoritmos.

En la tabla 3 se muestran los rangos y resultados del *test ómnibus de Friedman, Friedman Alienado* y *Quade* de donde el P-valor es menor en los tres casos, que el valor de $\alpha = 0.05$, por lo que tenemos suficiente evidencia para rechazar H_0.

Entonces tomando como algoritmo de control al *Iterated Local Search*, debido a que es el que tiene el menor rango en el caso de *Friedman Alineado*, y para el caso de *Friedman* y *Quade* tomamos como algoritmo de control al Algoritmo Memético hacemos las pruebas *post-hoc*, con un valor de $\alpha = 0.05$.

En las pruebas *Post-Hoc* mostradas en la tabla 4, *Friedman* alineado nos dice que no existe suficiente evidencia para rechazar H_0 en el caso de su comparación del ILS con el AM. Para los demás casos si existe suficiente evidencia para rechazar H_0 y afirmar que provienen de distribuciones diferentes.

Tabla 1. Datos para la configuración Inicial del AG, AM, SI, ILS y SA

Parámetro	Genético	Memético	Sistema Inmune	ILS	SA
Población	20			NA	NA

Llamadas a función	300,000				
Elitismo	0.1		NA		
Cruza	0.9		NA		
Muta	0.15		NA		
Poda	NA		100		NA
Iteraciones Búsqueda Local	NA	10	NA	10	NA

Tabla 2**. Resultados Estadísticos las diferentes Metaheurísticas aplicados a las instancias**

	Algoritmo	AG	AM	SI	ILS	SA	AG	AM	SI	ILS	SA	AG	AM	SI	ILS	SA
No	Instancia	Media					Mediana					Desviación Estándar				
1	LIA2009M041	264	250	269	**237**	269	288	273	283	**251**	283	14.6	13.5	13.7	**9.4**	13.7
2	LIA2009V041	134	132	132	**105**	132	172	155	164	**125**	164	14.7	11.9	15.2	**9.6**	15.2
3	LIA2009M042	209	220	237	**194**	237	260	249	271	**209**	271	17.8	16.2	21.2	**12.6**	21.2
4	LIA2009V042	134	121	127	**107**	127	154	143	160	**120**	160	13.0	10.7	12.2	**6.2**	12.2
5	LIA2010M041	200	195	203	**185**	203	227	218	227	**197**	227	14.7	10.0	13.4	**7.9**	13.4
6	LIA2010V041	125	118	123	**112**	123	141	135	144	**122**	144	7.2	8.1	8.6	**4.6**	8.6
7	LIA2010M042	100	94	100	**89**	100	112	109	113	**95**	113	7.4	7.2	8.6	**4.1**	8.6
8	LIA2010V042	25	25	25	**24**	25	32	31	34	**30**	34	6.4	4.3	6.5	**4.3**	6.5
9	**IGE2010M101**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0.4**	**0.0**	**0.8**	**0.0**	**0.8**
10	**IGE2010V101**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
11	IGE2010M102	11	8	8	**8**	8	18	15	18	**9**	18	3.7	3.2	4.0	**1.6**	4.0
12	IGE2010V102	47	2	2	**2**	2	131	3	3	**2**	3	64.9	0.6	0.9	**0.2**	0.9
13	LIA2011M041	50	49	51	**49**	51	58	57	58	**51**	58	4.9	5.0	4.7	**2.5**	4.7
14	LIA2011V041	44	41	39	**39**	39	49	49	51	**41**	51	3.9	4.5	4.3	**1.9**	4.3
15	LIA2011M042	58	5	5	**5**	5	153	8	9	**6**	9	104.5	1.7	2.2	**1.6**	2.2
16	LIA2011V042	235	24	25	**24**	25	393	29	31	**26**	31	108.1	3.1	4.1	**1.1**	4.1
17	IGE2011M101	50	50	51	**49**	51	59	55	57	**51**	57	6.1	4.1	5.0	**1.5**	5.0
18	IGE2011V101	40	39	43	**39**	43	50	47	53	**42**	53	4.4	4.2	4.6	**2.3**	4.6
19	IGE2011M102	494	22	26	**17**	26	993	30	33	**20**	33	312.3	4.1	5.5	**1.6**	5.5
20	IGE2011V102	30	27	24	**19**	24	38	33	40	**22**	40	3.9	4.6	6.0	**2.2**	6.0
21	**LIA2012M041**	**55**	**55**	**55**	**55**	**55**	**55**	**55**	**55**	**55**	**55**	**0.2**	**0.0**	**0.4**	**0.0**	0.4
22	LIA2012V041	42	42	42	**42**	42	44	43	44	**42**	44	1.1	0.8	1.0	**0.7**	1.0
23	IGE2012M101	129	131	120	**93**	120	157	147	162	**106**	162	16.2	11.5	18.3	**7.0**	18.3
24	IGE2012V101	77	66	72	**52**	72	93	86	94	**63**	94	9.4	6.9	10.4	**5.9**	10.4
25	IGE2013M101	526	492	551	**493**	551	571	570	607	**510**	607	27.2	25.6	25.5	**9.7**	25.5
26	IGE2013V101	303	298	330	**307**	330	328	354	371	**319**	371	22.9	10.7	23.4	**6.6**	23.4
27	IGE2014M101	117	115	116	**90**	116	141	129	138	**103**	138	11.5	9.0	11.0	**6.2**	11.0
28	IGE2014V101	63	65	64	**49**	64	78	72	78	**55**	78	6.8	5.3	6.0	**2.9**	6.0
29	IGE2014M102	63	57	62	**43**	62	83	71	81	**52**	81	9.5	8.6	10.2	**3.9**	10.2
30	IGE2014V102	62	65	66	**47**	66	94	83	97	**54**	97	13.9	12.0	14.3	**4.1**	14.3
31	IGE2015M101	168	162	162	**132**	162	198	189	200	**142**	200	14.9	13.3	12.7	**5.9**	12.7
32	IGE2015V101	91	77	91	**66**	91	110	100	107	**76**	107	11.4	11.2	11.4	**4.5**	11.4

Tabla 3. Resultados de las Pruebas de *Friedman, Friedman Alienado y Quade*

Algoritmos	**Friedman**	**Friedman Alineado**	**Quade**
Memético	1.671875	1334.5	1.747632576
ILS	1.703125	1267.5	1.894412879

SA	2.96875	2935	2.90719697
Genético	4.234375	3894.5	4.115056818
SI	4.421875	3448.5	4.335700758
Estadístico	89.5	78.4568916	24.72871556
***P*-Valor**	1.68156E-18	3.6969E-16	2.89503E-09

Tabla 4. Pruebas a pares para las pruebas de *Friedman, Friedman Alienado y Quade*

Algoritmos	**Friedman Alineado**			**Algoritmos**	**Friedman**			**Quade**		
	z	**P-Valor**	**Bonferroni**		**z**	**P-Valor**	**Bonferroni**	**z**	**P-Valor**	**Bonferroni**
ILS vs AG	36.71	**0.0000**	**0.0000**	AM vs AG	6.48	**9.01E-11**	**3.60E-10**	3.69	**0.0002**	**0.0008**
ILS vs AM	0.94	0.3490	1.0000	AM vs SI	6.96	**3.48E-12**	**1.39E-11**	4.04	**5E-05**	**0.0002**
ILS vs SI	30.48	**0.0000**	**0.0000**	AM vs ILS	0.08	0.936987	1.00	0.22	0.8187	1.0000
ILS vs SA	23.30	**0.0000**	**0.0000**	AM vs SA	3.28	**0.0010**	**0.004140**	1.81	0.0702	0.2810

A diferencia del *Friedman* alineado en el *Friedman y Quade* nos dice que el Algoritmo de control es el Memético. Continuamos con las pruebas a pares y nos dice que no existen diferencias en los representantes estadísticos en el caso del AM y ILS, similar a lo que nos dice el *Friedman Alineado* en sus pruebas a pares. Por lo tanto, tenemos que efectivamente los datos provienen de poblaciones con medianas diferentes, es decir, con base en los valores de la tabla 2 y las pruebas estadísticas podemos deducir que el algoritmo *Iterated local Search* es el que tiene el mejor rendimiento de los demás algoritmos.

5 CONCLUSIONES Y/O PROYECTOS FUTUROS.

La importancia de este trabajo estribó en el hecho de generar un diseño de horarios que minimiza de forma automática, el número de conflictos para los alumnos, logrando igualar e incluso mejorar el mínimo obtenido por el experto humano, considerando cierto conjunto de restricciones. Otro aspecto importante de esta investigación es que se empleó parte de dos metodologías para generar el diseño de horarios. La metodología API-Carpio que nos permite medir los conflictos de los alumnos y la propuesta por Soria, la cual permite considerar restricciones duras y blandas del problema, mediante el uso de diferentes estructuras (modelado de insumos). Esta hibridación de las metodologías API-Carpio y Soria, también nos permitió disminuir drásticamente el tiempo que le toma al experto humano realizar un diseño de horarios (de varios días o hasta semanas) a horas.

La búsqueda de estas soluciones de diseño de horarios, con el fin de propiciar resultados buenos en términos de rendimiento, debe ser guiada por algún tipo de estrategia. En otras palabras, con el uso de técnicas como la de diseño de particiones se presenta una forma diferente para la búsqueda de soluciones. Esta técnica propone una manera de dividir los elementos o variables de interés, con la importante particularidad de que busca respetar y cumplir las restricciones del problema, como la que se presenta aquí de "*dos materias diferentes impartidas por un mismo profesor deben estar en secciones diferentes de la partición*", es esa una de las estrategias de guía, es decir, dividir permite efectuar la labor de búsqueda de forma más sencilla. A diferencia de algunos enfoques del estado del arte donde parten de una solución que no considera todas o la mayoría de las restricciones, nuestra propuesta parte de trabajar en un espacio de soluciones factibles, para no dejar la labor de mejorar y satisfacer las restricciones a los algoritmos de inteligencia artificial.

Aun cuando la estrategia fue la misma para todos los algoritmos presentados en esta comparación, no todos obtuvieron el mismo resultado, para este caso con estas configuraciones en la experimentación se logró evidenciar que hay al menos una técnica Metaheurística, que difiere de manera favorable en cuanto a los resultados de las demás técnicas presentadas. Como trabajo futuro para el diseño de horarios, se adicionará la restricción dura en la que "*dos materias del mismo semestre no deben pertenecer a la misma sección de la partición*" y probar nuevamente el rendimiento de las técnicas Metaheurísticas con el fin de encontrar el enfoque que mejor gestione las variables del problema con el nuevo conjunto de restricciones.

6 AGRADECIMIENTO.

Agradecimientos al Tecnológico Nacional de México (TecNM), por el apoyo brindado a través del proyecto de investigación "*Enfoque evolutivo para la optimización del diseño de particiones en la solución de problemas de Timetabling*" Proyecto: xlfh6p(1559), al Consejo Nacional de Ciencia y Tecnología (CONACYT) por el apoyo económico brindado en esta investigación, a través de las becas nacionales para estudios de doctorado a becarios 308646 y 309062. Finalmente a la División de Estudios de Posgrado e Investigación del Instituto Tecnológico de León.

7 REFERENCIAS BIBLIOGRAFICAS

ABDOUN Otman, ABOUCHABAKA Jaafar. (2011). "A Comparative Study of Adaptive Crosso-ver Operators for Genetic Algorithms to Resolve the Traveling Salesman Problem", Inter-national Journal of Computer Applications.

ADRIAEN, M., Causmaecker, P., Demeester, P. (2006). Tackling the university course timetabling problem with an aggregation approach. In: Burke, K., Rudova, H. (eds.) Proceedings PATAT 2006, pp. 330–335.

AZUAJE, F. (2003). Artificial immune systems: A new computational intelligence approach Elsevier

CARPIO, M. (2006) Modelo integral de asignación optima de carga académica usando un algoritmo heurístico Encuentro de investigación en Ingeniería eléctrica,

F. COMELLAS, J. Fabrega, and A. S. O. Serra.(2001). Matemática discreta. Alfaomega,.

FOWLES, G.F.(1987) Introduction to Modern Optics Longman,

GOLDBERG, D. E. (1989). Genetic algorithms in search, optimization, and machine learning. Reading, Mass: Addison-Wesley Pub. Co

HELENA, L.; Olivier, M. & Thomas, S. Glover, F.; Kochenberger, G. & Hillier, F. S. (Eds.). (2003) Iterated Local Search Handbook of Metaheuristics, Springer New York, 57, 320-353

HERRERA LOZADA, J. C., Calvo, H., & Taud, H. (2011). A micro artificial immune system. A Micro Artificial Immune System.

JEAVONS, P.; Cohen, D. & Cooper, M. C. (1998) Constraints, consistency and closure Artificial Intelligence, 101, 251-265

JOAQUÍN DERRAC, SALVADOR GARCÍA. (2011) . "A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm inte", Swarm and Evolu-tionary Computation

K. H. ROSEN, Matemtica discreta y sus aplicaciones. McGraw-Hill Interamericana de Espaa S.L., 2004.

LAI, L. F., WU, C., HSUEH, N., HUANG, L., & HWANG, S. (2008). an artificial intelli-gence approach to course timetabling. International Journal on Artificial Intelligence Tools, 17(1), 223-240. doi:10.1007/s10479-011-0997-x.

LOURDES ARAUJO, CARLOS CERVIGÓN. (2009). Algoritmos Evolutivos, Un enfoque práctico (Alfaomega).

LÜ, Z., & hao, J. (2010). Adaptive tabu search for course timetabling. European Journal of Operational Research, 200(1), 235-244. doi:10.1016/j.ejor.2008.12.007

MAHIBA, A. A., Y DURAI, C. A. D. (2012). Genetic algorithm with search bank strategies for university course timetabling problem. Procedia Engineering, 38, 253-263. doi:10.1016/j.proeng.2012.06.033

MCCOLLUM B., McMullan, P., Parkes, A. J., Burke, E. K., & Qu, R. (2012; 2011). A new model for automated examination timetabling. Annals of Operations Research, 194(1), 291-315.

MICHEL, G. & Jean-Yves, P. (2010) Handbook of Metaheuristics Springer Publishing Company, Incorporated

SORIA-ALCARAZ, J. A., Martin, C., Héctor, P., Hugo, T., Laura, C. R., & Sotelo-Figueroa, M. A. (2013). Methodology of design: A novel generic approach applied to the course timetabling problem. Paper presented at the , 294 287-319. doi:10.1007/978-3-642-35323-9-12

TALBI, E. (2009). Metaheuristics: From design to implementation. US: Wiley.

VILLALOBOS ARIAS, M., Coello Coello, C. A., & Hernández Lerma, O. (2004). Convergence analysis of a multiobjective artificial immune system algorithm. Convergence Analysis of a Multiobjective Artificial Immune System Algorithm

WOLPERT, H., Macready, G. (1996). No free lunch Theorems for Search. Technical report The Santa Fe Institute, 1.

YANG, X.-S. (2010). Nature-inspired metaheuristic algorithms. Luniver press.