Training OFF-LineHyperheuristics For Course Timetabling Using K-Folds Cross Validation

Entrenamiento de una Hiperheurística con aprendizaje fuera de línea para el problema de Calendarización de horarios usando Validación Cruzada

Lucero de M. Ortiz-Aguilar1, Martín Carpio1, Jorge Alberto Soria-Alcaraz2, Héctor Puga1, Claudia Díaz1,Carlos Lino1, Jesús Eduardo Aldape1, Ofelia Alatorre1, Antonio Aguila1 y Verónica Tapia1

1Tecnológico Nacional de México-Instituto Tecnológico de León, León, Guanajuato, México
{ldm_oa, jmcarpio61, aldape_1310, vianey128, aguila_reyes_antonio}@hotmail.com, {pugahector, vtapia}@yahoo.com, {posgrado, carloslino}@itleon.edu.mx
2Universidad de Guanajuato, Guanajuato, Guanajuato, México
jorge.soria@ugto.mx

PALABRAS CLAVE:

: Hiperheurística, Validación Cruzada, Calendarización de Horarios, Búsqueda local Iterada, Metaheurística, Heurística

RESUMEN

En las Universidades, se busca un esquema de diseño de horarios que cumpla con las restricciones del alumnado, docentes, plan de estudios de la oferta educativa e inmuebles de la institución. Las Hiperheurísticas nos permiten generar metodologías que solucionen un conjunto de instancias de un problema. En este trabajo se muestra el uso de k-Folds Cross Validation en su versión de Leave-One-Out para el entrenamiento de Hiperheurísticas con aprendizaje fuera de línea aplicadas al problema de Course Timetabling, siendo esto el aporte del artículo. Como órgano rector para la Hiperheurística se utilizó el Iterated Local Search, empleando la metodología API-CARPIO donde las instancias de prueba provienen de datos reales del Instituto Tecnológico de León.

KEYWORDS:

HyperHeuristics, K-Folds Cross Validation, Course Timetabling, Iterated Local Search, Metaheuristics, Heuristics

ABSTRACT

In the universities seeks a timetabling scheme that covers a set of restrictions from the students, teachers, curriculum and school spaces. The hyper heuristics allow us generate methodologies which solve a set of instances of a specific problem. In this paper we use k-Folds Cross Validation Leave-One-Out version has been applied to Course Timetabling problem, so this is the most important contribution of this work, because we need a support for training a hyper heuristic and we used the k-Folds cross validation which is commonly using in pattern recognition. We use iterated Local Search as High level heuristic Chooser, using the API-CARPIO methodology and were used set of instances from Institute Technology of Leon for test this hyper heuristic.

Recibido: 1de julio de 2015 • Aceptado: 28 de octubre de 2015 • Publicado en línea: 7 de octubre de 2016

1 INTRODUCCIÓN

Dentro de una institución es importante optimizar recursos, tanto de personal como materiales. En el caso de las instituciones educativas, se busca que los alumnos aprovechen al máximo su estancia dentro de ésta, es decir, que su horario contemple cierto número de materias, para que un estudiante regular pueda terminar en tiempo y forma con el plan de estudios.

La calendarización de eventos dentro de las organizaciones es uno de los problemas más comunes y difíciles de tratar, debido a que se busca asignar diversas actividades y recursos en un timeslot [1]. En general el problema de calendarización de horarios se define a partir de un conjunto de eventos (clases, cursos, exámenes), los cuales deben ser asignados en un conjunto de timeslot y que están sujetos a un conjunto de restricciones [2].

El término Hiperheurística ha tenido una gran aceptación desde el 2000 donde Edmund K. Burke lo define como: "heurísticas que escogen heurísticas" en el contexto optimización combinatoria [3]. El problema de diseño de horarios modelado mediante la metodología API-Carpio puede ser resuelto mediante el enfoque Hiperheurístico [4]. Con el uso de Hiperheurísticas fuera de línea en este caso, se trata de encontrar un método o metodología o secuencia de heurísticas adecuado dada una situación, en lugar de tratar de resolver el problema directamente [3].

Para este trabajo se muestra el uso de k-Folds Cross Validation para el entrenamiento de una Hiperheurística de selección con aprendizaje fuera de línea. La herramienta de validación cruzada es comúnmente utilizada en el ámbito del Reconocimiento de patrones, para los clasificadores supervisados y Redes Neuronales [5].

Las instancias utilizadas para el entrenamiento de la Hiperheurística pertenecen a datos reales del Instituto Tecnológico de León (ITL), corresponden a diferentes planes educativos y distintas carreras como lo son Licenciatura en Administración (LIA), ingeniería en gestión Empresarial (IGE) e Ingeniería en Industrial (IIX).

2 COURSE TIMETABLING PROBLEM

El problema de calendarización de tareas puede ser definido como el proceso de asignar clases a recursos como lo son de tiempo, espacio (salones) y maestros (personal), mientras se satisfaga un conjunto restricciones [6].

Existen dos tipos de restricciones [7]:

•Duras. Es la restricción que absolutamente no puede ser quebrantada. Algunos ejemplos de restricciones duras son [8]: disponibilidad del salón, conflictos entre estudiantes, disponibilidad de recursos (maestros, salones).

•Blandas. El conjunto de restricciones que se prefieren satisfacer pero no se supone satisfacerlas todas. Algunos ejemplos de restricciones blandas son [8]: capacidad del Salón, mínimo de días ocupados, Etc.

Para esta experimentación se considera que se tiene la restricción dura de que no se pueden asignar dos materias del mismo semestre en el mismo número de timeslot.

2. 1 METODOLOGÍA API-CARPIO

La metodología API-Carpio [9] 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 salones 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:

$$FA=\sum(j=1)^k[\![FA_(V_j)]\!] \quad (2)$$

Dónde:

$$FA(Vj)=\Sigma(s=1)^((M_(V_j))-1)\sum(l=1)^((M(Vi))-s)▒[\![(A_(j,s)\wedge A_(j,s+l))]\!] \quad (3)$$

Teniendo:

FA(Vj)= Número de estudiantes en conflicto dentro del vector Vj.

Vj= Es un vector de tiempo que contiene diferentes materias.

A(j,s)∧A(j,s+l)= Número de estudiantes que demandan la inscripción simultánea de las materias M(j,s)∧M(j,s+1).

2.2 METODOLOGÍA DEL DISEÑO

La metodología del diseño propuesta por Soria et al. [10] 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. En este trabajo se utilizaran dos de las cuatro estructuras las cuales son: matriz MMA, lista LPH y adicionalmente una lista LMS. En [10] 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 timeslot.

Lista LPH: Esta lista nos da información acerca de cada lección (clase, evento o materia) en que posible timeslot puede ser asignada.

Lista LMS: Esta lista contiene información de cada materia y su correspondiente semestre. Esta información es importante debido a que las instancias cuentan con la restricción dura de que dos materias del mismo semestre no pueden ser asignadas en el mismo timeslot.

3 HIPERHEURÍSTICAS

La definición de Hiperheurística ha sido recientemente extendido para referirse a "método de búsqueda o mecanismo de aprendizaje para seleccionar o generar heurísticos, para solucionar problemas de búsqueda computacional" [3]. La clasificación de los enfoques hiperheurísticos puede darse de acuerdo a dos dimensiones definidas en [3]:

a)Naturaleza del espacio de búsqueda heurística. Estas comprenden las heurísticas de selección y generación. Estas a su vez pueden ser de construcción o perturbación.

b)La forma de retroalimentación durante el aprendizaje. Estos pueden ser aprendizaje en línea, fuera de línea y sin aprendizaje.

Las hiperheurísticas buscan en el espacio de las heurísticas; en lugar de buscar en el espacio de las soluciones, utiliza información específica para un problema limitado para controlar el proceso de búsqueda [11]. La información específica esta encapsulada en el modelo del problema y el pool de heurísticas de bajo nivel [11]. En este trabajo se usa una Hiperheurística con aprendizaje fuera de línea de selección, teniendo como órgano rector al Iterated Local Search. Uno de los componentes de una Hiperheurística es:

•Órgano rector de alto nivel: es una técnica de selección que interactúa no sobre el problema si no sobre la elección de heurísticas, debe de conocer en todo momento el estado actual del problema dado [1].

3.1 CONJUNTO DE HEURÍSTICAS.

Una parte fundamental del enfoque hiperheurístico es el conjunto de heurísticas de bajo nivel utilizadas a fin de generar un algoritmo de solución. En este trabajo se utilizaron 5 heurísticas diferentes, la cuales se obtuvieron con base en la experiencia y proceso de elaboración de horarios que realiza el experto del ITL, el cual cuenta con más de 15 años de experiencia en la elaboración de horarios. Dichas heurísticas se describen a continuación:

•Heurística 1 (H1). Esta heurística selecciona dos materias de un mismo semestre e intercambia sus números de timeslot. Los pasos de esta heurística están escritos en el algoritmo 1.

•Heurística 2 (H2). Esta heurística selecciona la materia con mayor conflicto de un determinado timeslot y la mueve a otro timeslot posible. Los pasos de esta heurística están escritos en el algoritmo 2.

•Heurística 3 (H3). Esta heurística selecciona la materia con menor conflicto de un determinado timeslot y la mueve a otro timeslot posible. Los pasos de esta heurística están escritos en el algoritmo 3.

•Heurística 4 (H4). Esta heurística selecciona la materia con el mayor conflicto con respecto a las demás, en un determinado timeslot y la mueve a otro posible timeslot. Los pasos de esta heurística están escritos en el algoritmo 4.

•Heurística 5 (H5). Esta heurística selecciona dos materias una con el mayor y otra con el menor conflicto con respecto a las demás, de dos diferentes timeslot e intercambia sus números de timeslot entre ellas. Los pasos de esta heurística están escritos en el algoritmo 5.

Algoritmo 1 **Heurística 1**

Recibe como entrada una solución del problema denotada como I_n
1: $id = aleatorio$ //se selecciona un semestre aleatoriamente de I_n
2: $m_1 = seleccionar\ una\ materia\ con\ semestre\ id$
3: $m_2 = seleccionar\ una\ materia\ con\ semestre\ id\ , diferente\ de\ m_1$
4: $t_1 = timeslot\ de\ m_1\ ; t_2 = timeslot\ de\ m_2$
5: $t_t = timeslot\ de\ m_1$
6: $t_1 = t_2\ ;\quad t_2 = t_t$
7: Calcular *fitness* de I_n
8: Regresar I_n

Algoritmo 2 **Heurística 2**

Recibe como entrada una solución del problema denotada como I_n
1: $id = aleatorio$ //se selecciona un semestre aleatoriamente de I_n
2: $m_1 = seleccionar\ una\ materia\ de\ semestre\ id$ con mayor conflicto
3: $l_{m1} = obtener\ lista\ de\ timeslots\ posibles\ para\ m_1$
4: Si $longitud\ de\ l_{m1} <> vacío$
5: $t_t = aletorio\ de\ l_{m1}$
6: $t_1 = timeslot\ de\ m_1$
7: $t_1 = t_t$
8: Calcular *fitness* de I_n
9: De otro modo Regresar I_n
10: Regresar I_n

Algoritmo 3 **Heurística 3**

Recibe como entrada una solución del problema denotada como I_n
1: $id = aleatorio$ //se selecciona un semestre aleatoriamente de I_n
2: $m_1 = seleccionar\ una\ materia\ de\ semestre\ id$ con menor conflicto
3: $l_{m1} = obtener\ lista\ de\ timeslots\ posibles\ para\ m_1$
4: Si $longitud\ de\ l_{m1} <> vacío$
5: $t_t = aletorio\ de\ l_{m1}$
6: $t_1 = timeslot\ de\ m_1$
7: $t_1 = t_t$
8: Calcular *fitness* de I_n
9: De otro modo Regresar I_n
10: Regresar I_n

Algoritmo 4 **Heurística 4**

Recibe como entrada una solución del problema denotada como I_n
1: $id = aleatorio$ //se selecciona un semestre aleatoriamente de I_n
2: $m_1 = seleccionar\ una\ materia\ de\ semestre\ id$ con mayor conflicto respecto a las demás materias de su mismo *timeslot*.
3: $l_{m1} = obtener\ lista\ de\ timeslots\ posibles\ para\ m_1$
3: Si $longitud\ de\ l_{m1} <> vacío$
4: $t_t = aletorio\ de\ l_{m1}$
5: $t_1 = timeslot\ de\ m_1$
6: $t_1 = t_t$
7: Calcular *fitness* de I_n
8: De otro modo Regresar I_n
9: Regresar I_n

Algoritmo 5 **Heurística 5**

Recibe como entrada una solución del problema denotada como I_n
1: $id = aleatorio$ //se selecciona un semestre aleatoriamente de I_n
2: $m_1 = seleccionar\ una\ materia\ de\ semestre\ id$ con menor conflicto
3: $m_2 = seleccionar\ una\ materia\ de\ semestre\ id$ con mayor conflicto
4: $t_1 = timeslot\ de\ m_1$
5: $t_2 = timeslot\ de\ m_2$
6: $t_t = timeslot\ de\ m_1$
7: $t_1 = t_2$
8: $t_2 = t_t$
9: Calcular *fitness* de I_n
10: Regresar I_n

4 ITERATED LOCAL SEARCH

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, la repetición de corridas aleatorias aisladas de la heurística [12]. En [12] se define el algoritmo 6 del Iterated Local Search.

En este trabajo la función de perturbación recibirá como entrada una solución denotada s^*; a esta se le aplicará un swap-two, es decir, se intercambiarán dos heurísticas de lugar. Como la Hiperheurística nos regresa una metodología, el orden de dichas heurísticas es importante y no es lo mismo la secuencia 12345 que la 12354. El criterio de aceptación será, si el fitness de s^(*′) es mayor que el fitness de s^*, entonces s^*=s^(*′).

Algoritmo 6 Iterated Local Search

1: s_0 =Generar una solución Inicial
2: s^* =Local Search(s_0);
3: t=0; contador de iteraciones
4: **Mientras** $t < Max$ número de generaciones **hacer**
5: s' =Perturbacion(s^*)
6: $s^{*\prime}$ =Local Search(s')
7: s^* =Criterio de Aceptación($s^*, s^{*\prime}$)
8: **Fin del lazo**

La representación de las soluciones para el caso de la Hiperheurística se muestra en la tabla 1.

Tabla 1. Codificación de las heurísticas para la representación de la solución:

Heurística	H1	H2	H3	H4	H5
Representación	1	2	3	4	5

5 K-FOLDS CROSS VALIDATION

La Validación Cruzada (CV, por sus siglas en inglés) es un estándar de predicción de tazas de error en técnicas de aprendizaje, dado un conjunto muestra. Donde el conjunto muestra será dividido en varias partes. El k-Folds Cross Validation, se utiliza en el campo de máquinas de aprendizaje para determinar la precisión con la que un algoritmo será capaz de predecir datos que no fueron entrenados en él [13].

En el k-Folds Cross Validation algunas veces llamado estimación de rotación, el conjunto de datos D es dividido aleatoriamente en k subconjuntos (folds) mutuamente excluyentes D_1,D_2,…,D_k, de tamaños iguales aproximadamente [14]. Entonces nuestro conjunto de entrenamiento puede ser particionado en dos conjuntos disjuntos [5]:

Subconjunto de entrenamiento: es usado para entrenar el modelo.

Subconjunto de Validación: es usado para probar o validar el modelo.

Para generar cada par de conjuntos, en la versión de leave-one-out, se mantiene una de las k partes como un conjunto de validación y se combinan las k-1 restantes para formar un conjunto de entrenamiento [5].

5.1 APLICACIÓN DEL K-FOLDS CROSS VALIDATION PARA LA EVALUACIÓN DE HIPERHEURÍSTICAS FUERA DE LÍNEA

En este trabajo para hacer el entrenamiento de la Hiperheurística fuera de línea de selección, se utilizó el k-Folds Cross Validation en su versión de leave-one-out, El enfoque de la Hiperheurística fuera de línea requiere de un conjunto de instancias para su entrenamiento y un conjunto de instancias para la prueba, esto con el objetivo de lograr generalidad.

El primer lugar hay que dividir nuestro conjunto de instancias en los k folders deseados, por ejemplo: si se tienen 35 instancias y se tiene un k=5, cada folder contendrá 7 instancias. En segundo lugar se harán tantos entrenamientos como número de folders, por ejemplo: si se tienen k=5 folders se harán 5 experimentos (ver figura 1). Para cada experimento se tomará un folder para probar y el resto para hacer el entrenamiento en la Hiperheurística (HH), por ejemplo: en el primer experimento se toma del folder 1 al 4 y se deja el folder 5 para entrenar, en el caso de k=5.

Una vez entrenada la HH nos regresará una metodología la cual aplicaremos al folder de prueba y el fitness resultante de dicho folder es el que se almacenará para hacer el acumulado de todos los experimentos. Cuando se realicen todos los experimentos y se hallan obtenido todos los fitness de los folders de prueba, se hará una sumatoria de estos y el resultado es el Rendimiento de nuestra Hiperheurística. En la figura 1 se muestra un ejemplo de aplicación de la técnica de k-Folds.

No. Experimento	Folder de Entrenamiento				Folder de prueba	Fitness del Folder de Prueba	
1	1	2	3	4	5	10	+
2	1	2	3	5	4	89	+
3	1	2	4	5	3	30	+
4	1	3	4	5	2	50	+
5	2	3	4	5	1	-5	+
			Rendimiento de la HH			174	

Fig. 1. Ejemplo de experimento con k-Folds

6 EXPERIMENTACIÓN Y RESULTADOS

Anteriormente en [15] se hizo un comparativo de diferentes Metaheurísticas aplicadas al problema del Course Ttimetabling, donde se aplicaron pruebas estadísticas no paramétricas para contrastar los diferentes algoritmos, por lo tanto en este trabajo se presenta únicamente como aporte el uso de k-Folds Cross Validation para el entrenamiento de una Hiperheurística. Las 34 instancias usadas para las pruebas para la Hiperheurística pertenecen a ITL, estas corresponden a tres planes educativos distintos LIA, IGE e IIX, pertenecientes al año del 2009 y 2015, cuentan con aproximadamente de 46 a 64 clases (eventos) y una cantidad de 9 a 11 timeslots respectivamente.

Tabla 2. Datos relacionados a las instancias del ITL

No de instancia	Carrera	No. De Timeslot	No. De Materias
1-15	LIA	9	46-58
16-30	IGE	9-11	58
31-34	IIX	9	64

La configuración utilizada en el algoritmo Iterated Local Search se muestran en la tabla 3, donde tenemos que el criterio de paro fue el de cantidad máxima de iteraciones. La tabla 4, 5, 6, 7 y 8 muestran los resultados obtenidos (conflictos) al aplicar el k-Folds Cross Validation. La columna llamada Fitness E corresponde al fitness obtenido por la metodología que generó la Hiperheurística con su correspondiente folder de entrenamiento y la columna llamada Fitness P corresponde al fitness obtenido por la metodología al aplicarse al conjunto de prueba.

Tabla 3. Configuración inicial para el *Iterated Local Search*

Parámetro	Valor
Tamaño de solución	35
Máximo de iteraciones	1,000
Máximo de iteraciones del *local Search*	10

Tabla 4. Resultados de entrenamiento y prueba para la Hiperheurística con un k=2.

Entrenamiento	# Instancias	*Fitness* E	Prueba	# Instancias	*Fitness* P
Folder 1	17	153	Folder 2	17	-588
Folder 2	17	384	Folder 1	17	-590
			Total		-1178

Tabla 5. Resultados de entrenamiento y prueba para la Hiperheurística con un k=3.

Entrenamiento	# Instancias	*Fitness*E	Prueba	# Instancias	*Fitness*P
Folder 1	22	265	Folder 2,3	12	-333
Folder 2	23	258	Folder 1,3	11	-55
Folder 3	23	286	Folder 1,2	11	-136
			Total		-524

Tabla 6. Resultados de entrenamiento y prueba para la Hiperheurística con un k=4.

Entrenamiento	# Instancias	*FitnessE*	Prueba	# Instancias	*FitnessP*
Folder 1	25	330	Folder 2,3,4	9	-126
Folder 2	25	359	Folder 1,3,4	9	44
Folder 3	26	272	Folder 1,2,4	8	588
Folder 4	26	372	Folder 1,2,3	8	-350
			Total		-1020

Tabla 7. Resultados de entrenamiento y prueba para la Hiperheurística con un k=5.

Entrenamiento	# Instancias	*FitnessE*	Prueba	# Instancias	*FitnessP*
Folder 1	27	153	Folder 2,3,4,5	7	-215
Folder 2	27	384	Folder 1,3,4,5	7	-257
Folder 3	27	163	Folder 1,2,4,5	7	-113
Folder 4	27	314	Folder 1,2,3,5	7	-17
Folder 5	28	275	Folder 1,2,3,4	6	-133
			Total		-735

Tabla 8. Resultados de entrenamiento y prueba para la Hiperheurística con un k=6.

Entrenamiento	# Instancias	*FitnessE*	Prueba	# Instancias	*FitnessP*
Folder 1	28	220	Folder 2,3,4,5,6	6	-71
Folder 2	28	338	Folder 1,3,4,5,6	6	-370
Folder 3	28	135	Folder 1,2,4,5,6	6	-81
Folder 4	28	352	Folder 1,2,3,5,6	6	-96
Folder 5	29	282	Folder 1,2,3,4,6	5	-111
Folder 6	29	288	Folder 1,2,3,4,5	5	-7
			Total		-736

7 CONCLUSIONES

El enfoque hiperheurístico nos genera metodologías que pueden dar solución a un conjunto de instancias de un problema, y no a una sola instancia en específico. En las hiperheurísticas con aprendizaje fuera de línea, es muy importante el conjunto de instancias del problema con el que se realiza el entrenamiento. Por lo tanto, es primordial que el conjunto de instancias de entrenamiento y prueba no este sesgado, es decir, que en alguno de los conjuntos contenga un solo tipo de instancias.

En este trabajo se usó la técnica de k-Folds Cross Validation en su versión de leave-one-out para el entrenamiento de la Hiperheurística de selección con perturbación. Siendo esta una técnica empleada comúnmente en el área de reconocimiento de patrones, se logró demostrar que se puede aplicar en el enfoque hiperheurístico.

Como se puede apreciar en los resultados se utilizó desde un k=2 hasta un k=6, además de que conforme se va incrementando el valor de k se generan una mayor cantidad de folders con pocas instancias y en el caso específico cuando k toma el valor de dos, los folders contienen un mayor número de instancias. Por lo tanto conforme k se incrementa, el número de instancias de prueba es menor y aumenta el número de instancias con el que se entrena.

Como trabajo futuro se propone el realizar más experimentación con diferentes valores de k, además de hacer un análisis exhaustivo del conjunto de instancias o aplicar alguna regla de muestreo que nos permita seleccionar instancias que por compartir alguna característica se puedan agrupar en clases; por ejemplo: misma cantidad de eventos, timeslot, etc., y seleccionar al menos una de cada clase para que estén presentes en el conjunto de entrenamiento y prueba. En la parte Hiperheurística se propone realizar experimentación utilizando diferentes Metaheurísticas como órgano rector.

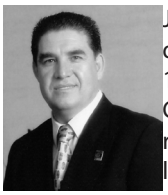
AGRADECIMIENTOS

Agradecimientos al Consejo Nacional de Ciencia y Tecnología (CONACYT) México por el apoyo brindado en esta investigación con el número de beca 308646 y a la División de Estudios de Posgrado del Instituto Tecnológico de León.

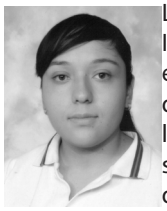
REFERENCIAS

1. Jorge A, S., Martin, C. J., & Hugo, T.: Academic timetabling design using hyper-heuristics. Berlin, Heidelberg: Springer Berlin Heidelberg, (2011; 2010), pp. 43-56 doi:10.1007/978-3-642-15534-5_3.
2. Soria-Alcaraz Jorge, A., Martín, C., Héctor, P., & Sotelo-Figueroa, M. A.: Comparison of metaheuristic algorithms with a methodology of design for the evaluation of hard constraints over the course timetabling problem. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 289-302 (2013), doi:10.1007/978-3-642-33021-6_23.
3. Burke, E. K., Gendreau, M., Hyde, M., Kendall, G., Ochoa, G., Özcan, E., et al. (2013). Hyper-heuristics: A survey of the state of the art. Journal of the Operational Research Society (JORS), 64(12), 1695–1724.
4. Soria-Alcaraz, J. A.; Carpio, J. M.; Puga, Hé.; Melin, P.; Terashima-Marn, H.; Reyes, L. C. & Sotelo-Figueroa, M. A. Castillo, O.; Melin, P.; Pedrycz, W. & Kacprzyk, J.: Generic Memetic Algorithm for Course Timetabling ITC2007 Recent Advances on Hybrid Approaches for Designing Intelligent Systems, Springer, vol. 547, pp. 481-492 (2014).
5. Simon Haykin. Neural Networks A Comprehensive Foundation, Prentice Hall, pp. 235-240 (1999).
6. LAI, L. F., WU, C., HSUEH, N., HUANG, L., & HWANG, S.: An artificial intelligence approach to course timetabling. International Journal on Artificial Intelligence Tools, pp. 223-240 (2008). doi:10.1007/s10479-011-0997-x.
7. McCollum, B., McMullan, P., Parkes, A. J., Burke, E. K., & Qu, R.; A new model for automated examination timetabling. Annals of Operations Research, pp. 291-315 (2012; 2011).
8. Aladag, C., & Hocaoglu, G.: A tabu search algorithm to solve a course timetabling problem. HACETTEPE JOURNAL OF MATHEMATICS AND STATISTICS, pp. 53-64 (2007).
9. Carpio-Valadez, J.M.: Integral Model for optimal assignation of academic tasks, Encuentro de investigación en ingeniería eléctrica. ENVIE, Zacatecas, pp. 78–83 (2006).
10. Soria-Alcaraz, J. A., Martin, C., Héctor, P., Hugo, T., Laura, C. R., & Sotelo-Figueroa, M. A.: Methodology of design: A novel generic approach applied to the course timetabling problem, pp. 287-319 (2013). doi: 10.1007/978-3-642-35323-9-12.
11. Soria-Alcaraz, J., Ochoa, G., Swan, J., Carpio, M., Puga, H., & Burke, E. Effective learning hyper-heuristics for the course timetabling problem. European Journal of Operational Research, pp. 77-86 (2014). doi:10.1016/j.ejor.2014.03.046.
12. Talbi, E. (2009). Metaheuristics: From design to implementation. US: Wiley.
13. Rajput Sumangala and Rajeshwari Horakeri. Recognition of Kannada Handwritten Numerals Using Fourier Descriptors. 2010. Computer Vision and Information Technology Advances and Aplicayions. Edited by Kale, Mehrotra, Manza. Page 540.
14. Ron Kohavi. A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection. Appears in the International Joint Conference on Artificial Intelligence (IJCAI), pp. 1-7 (1995).
15. Lucero de Montserrat Ortiz Aguilar, Juan Martín Carpio Valadez, Héctor José Puga Soberanes, Claudia Leticia Díaz González, Carlos Lino Ramírez y Jorge Alberto Soria-Alcaraz. Comparativa de algoritmos bioinspirados aplicados al problema de calendarización de horarios.", in Research in Computing Science Issue 94 (2015), pp. 33-43.

Acerca de los autores



Juan Martín Carpio Valadez, obtuvo el grado de Doctor en Ciencias (Óptica) del CIO en 1995. Su experiencia profesional incluye el CIO, ITESM campus León, Universidad Iberoamericana plantel León, y desde 1994 a la fecha el Instituto Tecnológico de León, en donde ocupó el cargo de Jefe del Depto. de Sistemas y Computación de 1999 a 2004, el cargo de Jefe de la División de Estudios de Posgrado e Investigación de 2004 a 2006. Ha participado como responsable y colaborador de varios proyectos de investigación, apoyados por CONCyTEG, COSNET y DGEST. Colaboró en la formación de recursos humanos a través de la dirección tesis de licenciatura (10), maestría (35) y doctorado (3). Actualmente es miembro del Sistema Nacional de Investigadores nivel I. Es miembro del Consejo de Posgrado y miembro del claustro del Doctorado Interinstitucional en Ciencias en Computación. Sus áreas de interés son: sistemas inteligentes, University Timetabling, Metaheurística, Hiperheurísticas.



Lucero de Montserrat Ortiz Aguilar es Ingeniero en Sistemas Computacionales, egresada en 2009 en el Instituto Tecnológico de León. Actualmente se encuentra cursando la Maestría en Ciencias de la Computación, en su segundo año en el Instituto Tecnológico de León. Actualmente se encuentra investigando y aplicando diferentes técnicas Metaheurísticas e Hiperheurísticas para la generación del diseño de horarios. Ha publicado artículos en el Congreso de la Mujer 2015, organizado por el CIO y en COMIA 2015 (Congreso Mexicano de Inteligencia Artificial), en relación a la aplicación de Metaheurísticas al diseño de horarios. Sus áreas de interés son: Técnicas de optimización combinatoria, Metaheurísticas, Hiperheurísticas y Visión por computadora.

Jorge Alberto Soria Alcaraz es egresado como Ingeniero en sistemas computacionales por el instituto tecnológico de león en 2008, continúo su formación como Maestro en ciencias en Ciencias de la Computación por la misma casa de estudios egresando en 2010. Obtuvo el grado de Doctor en ciencias de la Computación por el Instituto Tecnológico de Tijuana B.C parte del Tecnológico Nacional de México en el 2015. El Dr Soria-Alcaraz cuenta trabajos publicados en el área nacional e internacional sobre los temas de Hiper-heurísticas, Timetabling así como Autonomus Search. Ha asistido a congresos nacionales e Internacionales a presentar trabajos acordes a estas áreas. Actualmente se desempeña como profesor investigador de tiempo completo en la División de Ciencias Económico-Administrativas de la Universidad de Guanajuato campus Guanajuato apoyando la carrera de Licenciatura en Sistemas de Información Administrativa. También pertenece al Sistema Nacional de Investigadores (SNI) con la distinción de Candidato a investigador Nacional.

Héctor J. Puga Soberanes se graduó de Licenciatura en Físico Matemáticas, en el Instituto Politécnico Nacional en 1993. Obtuvo el grado de Maestría en Ciencias (Óptica) en 1995, egresado del Centro de Investigaciones en Óptica, A. C. (CIO), obteniendo el titulo por parte de la Universidad de Guanajuato. Obtuvo el grado de Doctor en Ciencias (Óptica), egresado del CIO, obteniendo el titulo por parte de la Universidad de Guanajuato en 2002. Cuenta con publicaciones internacionales, en congresos internacionales y nacionales. Ha participado como responsable y colaborador de varios proyectos de investigación, apoyados por CONCyTEG, COSNET y DGEST. A colaborado en la formación de recursos humanos a través de la dirección tesis de licenciatura (3) y Maestría (4) Actualmente es miembro del Sistema Nacional de Investigadores nivel I. Profesor con perfil deseable (Promep) de Agosto de 2005 a la fecha. Sus áreas de interés son: Metrología , Hiperheurísticas y sistemas inteligentes.

Carlos Lino Ramírez profesor Investigador y jefe de la División de Estudios de Posgrado e Investigación (DEPI) del Instituto Tecnológico de León, en el programa de Maestría en Ciencias en Ciencias de la Computación desde Noviembre del 2010. Doctorado en Arquitectura y Tecnología de los Sistemas Informáticos por la Universidad Politécnica de Valencia, España (2012). Maestro en Ciencias en Ciencias Computacionales por el Instituto Tecnológico de León (1999). Ingeniero en Sistemas Computacionales por el Instituto Tecnológico de León (1996). Ha sido subdirector del Instituto Tecnológico de León (2006-2007), jefe del departamento de Sistemas y Computación del Instituto Tecnológico de León (2004-2006), jefe del área de redes en el Sistema Avanzado de Bachillerato y Educación Superior (1999-2001). Ha presentado sus trabajos de investigación en diversos congresos internacionales en España, Italia, Alemania y México.Sus áreas de investigación son Inteligencia de Ambiente, Algoritmos de Encaminamiento y Redes de Sensores Inalámbricas.