SOFTWARE PARA LA GENERACIÓN AUTOMÁTICA DE HORARIOS ACADÉMICOS

SOFTWARE FOR AUTOMATIC GENERATION OF ACADEMIC TIMETABLES

Samuel E. Viñas[1], Noel E. Rodríguez[2], Edgar Corona[3], Abraham J. Jiménez[3]
(1) Instituto Tecnológico Superior de Ciudad Hidalgo, División de Ingeniería en Sistemas Computacionales, Av. Ing. Carlos Rojas Gutiérrez 2120, Fracc. Valle de la Herradura, 61100 Ciudad Hidalgo, Michoacán - México
(2) Instituto Tecnológico de Zitácuaro, Departamento de Sistemas y Computación, Av. Tecnológico No. 186 Manzanillos, 61534 H. Zitácuaro, Michoacán - México
(3) Tecnológico de Estudios Superiores de Ecatepec, División de Ingeniería en Sistemas Computacionales, Av. Tecnológico S/N, 55210 Ecatepec de Morelos - México
(e-mail: salvarez@itsch.edu.mx)

Recibido: 20/10/2018 - Evaluado: 26/11/2018 - Aceptado: 20/12/2018

RESUMEN

En este trabajo se presenta un prototipo de software para resolver el Problema de Asignación de Horarios Académicos (PAHA) en Instituciones de Educación Superior (IES) de manera automática. El software implementa Algoritmos Genéticos (AG) con un operador de mutación adaptativa para la generación de los horarios académicos. Las pruebas se realizaron utilizando información académica de la Unidad de Estudios Superiores Villa Victoria, Estado de México, México. Los horarios académicos generados cumplieron con las restricciones establecidas y se generaron en un lapso de tiempo corto. La configuración de AG utilizado resolvió el PAHA de manera eficiente, además, el software puede ser fácilmente adaptado en otras IES.

ABSTRACT

In this paper it is presented a software prototype to solve the Problem of Assignment of Academic Schedules (PAAS) in Higher Education Institutions (HEI) automatically. The software implements Genetic Algorithms (GA) with an adaptive mutation operator for the generation of academic schedules. The tests were conducted using academic information of the Villa Victoria Superior Studies Unit, State of Mexico, Mexico. The academic schedules generated fulfill the established constraints and were generated in a short period of time. The used GA configuration solved the PAAS in an efficient manner, also, the software can be easily implemented in other IES.

Palabras clave: horarios académicos, cómputo evolutivo, algoritmos genéticos, universidades
Keywords: academic schedules, evolutionary computation, genetic algorithms, universities

INTRODUCCIÓN

En la actualidad, la carga de trabajo en las Instituciones de Educación Superior (IES) ha incrementado; por un lado, la oferta educativa se diversifica de acuerdo a las necesidades económicas/sociales de la comunidad, y por otro lado, la necesidad de un mercado laboral más competitivo y complejo ha aumentado el número de estudiantes, y todo lo anterior trae como consecuencia que la administración de los recursos académicos se vuelva compleja. El Problema de Asignación de Horarios Académicos (PAHA) consiste en la correcta aplicación de recursos (profesores, espacios, horarios, materias, entre otros) de tal manera que se cubran las necesidades académicas. Por su naturaleza, este problema se vuelve complejo al tener un número finito de recursos, necesidades con ciertas características y un conjunto de restricciones que deben cumplirse.

El Cómputo Evolutivo (CE) es una rama de la Inteligencia Artificial (IA) que se especializa en la solución de problemas de optimización; de manera general trabaja con un conjunto de soluciones (poblaciones de individuos), las cuales son seleccionadas y mejoradas (evolucionadas) utilizando ciertas estrategias hasta encontrar una solución óptima o muy cerca del óptimo (Haupt *et al.*, 2004). Dentro del CE se encuentran algoritmos que usan modelos inspirados en la naturaleza (bioinspirados) para resolver problemas de optimización, tal es el caso de los Algoritmos Evolutivos (AE); en diversas investigaciones los AE han demostrado ser efectivos para dar solución a diferentes casos de estudio del PAHA (Abdullah *et al.*, 2007; Rodríguez *et al.*, 2014; Raghavjee & Pillay, 2013).

En la literatura se pueden encontrar múltiples estrategias basadas en CE para dar solución a diferentes problemas de optimización: Estrategias Evolutivas (EE), usada para problemas de ruteo y redes de comunicaciones, problemas de bioquímica, óptica, ingeniería, magnetismo e hidrodinámica (Coello, 2016); Programación Evolutiva (PE) aplicada en problemas de predicción, optimización, planeación de rutas, entrenamiento de redes neuronales, reconocimiento de patrones y aprendizaje automático (Coello, 2016); Algoritmos Genéticos (AG) usados para problemas de optimización, predicción, asignación de recursos, sistemas clasificadores, aprendizaje, entre otros (Coello, 2016). Beligiannis *et al.* (2008), desarrollaron una solución basada en AE para resolver un conjunto de instancias griegas del PAHA, compararon sus resultados en contra de otros algoritmos obteniendo resultados de manera más rápida. Dino Matijas *et al.* (2010), propusieron una solución basada en Optimización por Colonia de Hormigas (OCH) la cual fue puesta a prueba en 5 instancias PAHA, las soluciones fueron comparados contra soluciones basadas en enfoques GRASP, los resultados muestran un mejor desempeño de la solución basada en OCH. Chen & Shih (2013), proponen una solución basada en Optimización por Enjambre de Partículas (OEP). La propuesta está basada en una versión especial de OEP, el Estándar OEP (EOEP), el cual es aplicado a una búsqueda local, los resultados muestran mejor desempeño para EOEP. De manera particular se han empleado diversas técnicas basadas en CE/AG para resolver el PAHA: Búsqueda Local (Raghavjee & Pillay, 2013), AG con reparación de genes (Rodriguez *et al.*, 2014), AG combinados con Recocido Simulado y Escalado de Colina (Raghavjee & Pillay, 2013; Cubillos *et al.*, 2013), AG con representación binaria y real (Flores *et al.*, 2004), entre otras. También existen métodos heurísticos los cuales son eficientes para refinar la búsqueda global en un tiempo corto: recocido simulado, búsqueda tabú, algoritmos evolutivos, entre otros (Arntzen & Løkketangen, 2005; Chmait & Challita, 2013; Daskalaki & Birbas, 2005).

Los AG originalmente fueron desarrollados por John Henry Holland con el propósito de resolver problemas de aprendizaje de máquinas (Coello, 2016). El funcionamiento básicamente consiste en: 1) generar aleatoriamente un conjunto de soluciones iniciales (población inicial), 2) calcular el valor de aptitud de cada individuo de la población, 3) realizar una selección probabilística de las mejores soluciones (individuos) basados en su aptitud, 4) aplicar operadores genéticos (cruce y mutación) para generar el siguiente conjunto de soluciones, y, repetir hasta que cierta condición se satisfaga (por ejemplo encontrar la solución óptima). Existen diferentes variantes de AG, siendo los más populares: AG Binarios, AG Continuos, AG Adaptativos, AG Híbridos, AG no Generacionales, entre otros (Haupt *et al.*, 2004). Para poder aplicar un AG se requiere: una representación de las mejores soluciones potenciales del problema, un método para crear una población inicial de soluciones (normalmente es aleatorio), una función de evaluación que permita clasificar una solución en base a su aptitud,

la aplicación de operadores genéticos que permitan alterar la composición genética de los descendientes de las nuevas generaciones y los parámetros de configuración (tamaño de la población, método de selección, tipo y tasa de cruce y mutación). El proceso de búsqueda de los AG se puede refinar/mejorar a través de diferentes técnicas: Búsqueda Local (BL), Búsqueda Tabú (BT), Escalador de Colina (EC), Recocido Simulado (RS), entre otras (Raghavjee & Pillay, 2013).

Existen diferentes aplicaciones de software para generar horarios de forma automática o semiautomática. Algunos de estos son comerciales y libres; entre los prototipos de software comercial más populares se tiene: aschorarios® (AscHorarios, 2017), Peñalara® (Peñalara, 2017), UntisExpress® (UntisExpress, 2017) entre otros, sin embargo, no se conoce con precisión los algoritmos implementados. Los prototipos de software libre de código abierto se tienen a FET (Lalescu, 2017), uniTime (Unitime, 2018), KHE (KHE, 2018) y GOAL (Group of Optimization and Algorithms - GOAL, 2018), estos prototipos iniciaron con los primeros resultados con algoritmos genéticos, sin embargo, con el transcurso de tiempo y de diversas pruebas y aportaciones han realizado diversas mejoras. Aunque todos los prototipos mencionados presentan buenas soluciones al PAHA uno de los principales problemas que presentan es que son de alto costo y poca adaptación a diferentes necesidades de las universidades en México.

En esta investigación se presenta un prototipo de software para dar solución al PAHA. El prototipo integra un AG y un Operador de Mutación Adaptativo (OMA). El OMA refina la exploración conforme avance el proceso de búsqueda para generar soluciones que cumplan con los requerimientos y condiciones establecidas. Se tiene como caso de estudio la Universidad Mexiquense del Bicentenario (UMB) específicamente en la Unidad de Estudios Superiores Villa Victoria (UESVV) del Estado de México, México. Los experimentos se realizaron utilizando diferentes configuraciones iniciales del AG y aplicando variantes en las estrategias de búsqueda adaptativa (AG-OMA y AG-RS)

METODOLOGÍA

Problema de Asignación de Horarios Académicos (PAHA)

El PAHA consiste en asignar cursos a profesores con restricciones de tiempo y aulas, de tal manera que se satisfagan las necesidades académicas. Las principales variables del problema son: los profesores, los cursos, las aulas de clase y los periodos de tiempo. Todas las variables se pueden asignar de múltiples maneras (Coello, 2016). En general existen dos tipos de restricciones: fuertes y débiles (Abdullah *et al.*, 2007). Las restricciones fuertes son aquellos requisitos que deben ser cumplidos forzosamente, mientras que las restricciones débiles son requisitos deseables pero su cumplimiento no es obligatorio, en la medida que se cumplan ofrecerán una mayor calidad en la solución encontrada.

Este tipo de problemas se formula como un problema de optimización combinatoria, los cuales usualmente son NP-Completos (Ponce *et al.*, 2014; Gen & Cheng, 1999), en donde se busca encontrar la mejor solución posible minimizando la violación de restricciones (Raghavjee & Pillay, 2013 Gen & Cheng, 1999). Existen dos tipos de soluciones: válidas y factibles. Las soluciones válidas son aquellas que cumplen con las restricciones fuertes, mientras que las soluciones factibles son aquellas que cumplen las restricciones fuertes y la mayor parte de las restricciones débiles (Abdullah *et al.*, 2007).

Planteamiento del problema

Hoy en día, la UESVV cuenta con cinco programas educativos (carreras): Licenciatura en Contaduría, Criminología, Informática, Enfermería y Psicología Industrial. La programación de horarios está basada en el plan de estudios de cada carrera y consiste en asignar alrededor de siete cursos por semestre por cada programa educativo. En un periodo escolar que consta de cinco grados por carrera, se tienen que programar 175 cursos o asignaturas para lo cual se cuenta con 41 profesores, 20 aulas de clase y 72 espacios de tiempo de

una hora de lunes a sábado. La asignación de horarios académicos se realiza de forma manual por el personal administrativo, el cual carece de las herramientas e información necesarias para satisfacer las necesidades académicas. Lo anterior ocasiona que los horarios académicos generados no sean óptimos (por ejemplo, que no se cumpla a tiempo con la carga académica docente, que haya empalmes, etc.), además del excesivo tiempo que se invierte en corregir los errores de los horarios generados (aproximadamente 200 horas/persona).

Modelado del problema

Para desarrollar el modelo matemático del problema se considera las siguientes restricciones:
a) Restricciones fuertes:
1. Las asignaturas deben ser programadas en aulas de clase que no estén ocupadas.
2. Un profesor no puede dar cátedra a más de un grupo a la vez.
3. Las aulas de clase deben tener espacio suficiente para acomodar el grupo de estudiantes asignado.
4. El horario de clase debe ser programado en horarios de 07:00 a 19:00 horas de lunes a sábado.
5. Se debe cumplir con la totalidad de asignaciones de las asignaturas de cada programa académico.

b) Restricciones débiles:
1. Los profesores imparten cátedra de las asignaturas acordes a su perfil y experiencia profesional.
2. Los profesores proponen las asignaturas que puede impartir.
3. El horario se basa en la disponibilidad de tiempo del profesor.
4. Se debe evitar horas muertas (sin actividad) para profesores y estudiantes.
5. Se debe evitar que un mismo profesor imparta clase de la misma asignatura en horas discontinuas.

Los recursos con los que cuenta la universidad se definen de la siguiente manera:
- $\boldsymbol{P}$ es el conjunto de un número de $\boldsymbol{N_p}$ Profesores ; Cada profesor se distingue con $\boldsymbol{p_i}$ donde $\boldsymbol{i}$ es un índice que toma valores [1, 2, ..., $\boldsymbol{N_p}$].
- $\boldsymbol{C}$ es el conjunto de un número $\boldsymbol{N_c}$ de cursos o asignaturas $\boldsymbol{c_i}$; donde $\boldsymbol{i}$ es un índice asociado a cada asignatura con valores [1, 2, ..., $\boldsymbol{N_c}$].
- $\boldsymbol{T}$ es el conjunto de un número $\boldsymbol{N_t}$ espacios de tiempo $\boldsymbol{t_m}$ (días de la semana y horas del día); donde $\boldsymbol{m}$ es un índice asociado a un espacio de tiempo que toma valores [1, 2, …, $\boldsymbol{N_t}$] y el total de tiempo de espacios, se pueden identificar en número de días $\boldsymbol{d_t}$ y número de horas de cada espacio por día $\boldsymbol{h_t}$.
- $\boldsymbol{A}$ es el conjunto de $\boldsymbol{N_a}$ Aulas de clase $\boldsymbol{n_a}$; donde n es un índice que toma valores [1, 2, …, $\boldsymbol{N_a}$] y $\boldsymbol{N_a}$ es el número total de aulas disponibles.
- $\boldsymbol{X}: \boldsymbol{C} \rightarrow \boldsymbol{P} \times \boldsymbol{T} \times \boldsymbol{A}$ se define como el horario $\boldsymbol{X}$ en donde para cada asignatura $\boldsymbol{c} \in \boldsymbol{C}$ se asigna un profesor $\boldsymbol{p} \in \boldsymbol{P}$, un espacio de tiempo $\boldsymbol{t} \in \boldsymbol{T}$ y una aula de clase $\boldsymbol{a} \in \boldsymbol{A}$.

El objetivo principal es minimizar la violación de restricciones fuertes y débiles. Las Ecuaciones 1, 2 y 3 muestran el modelo matemático del problema:

$$Minimizar\ f(X), \qquad X \in \mathbb{R}^n$$

$$f(X) = \sum_{i=1}^{5} RF_i(X) + \sum_{i=1}^{5} RD_i(X) \tag{1}$$

$Sujeto\ a$:

$$\sum_{i=1}^{5} RF_i(X) = 0 \tag{2}$$

$$\sum_{i=1}^{5} RD_i(X) \geq 0 \tag{3}$$

En donde, la Ecuación 1 define la función objetivo $f(X)$ a minimizar y X es un arreglo definido en $\mathbb{R}^n$. $RF(X)$ es una función que permite contabilizar la violación de las 5 restricciones fuertes y $RD(X)$ es una función que permite contabilizar el número de violaciones de las 5 restricciones débiles; la violación de cada restricción tiene asociado una ponderación o peso específico, en donde la ponderación de las restricciones fuertes es mayor que el de las restricciones débiles. La Ecuación 2 establece el cumplimiento de las restricciones fuertes y la Ecuación 3 establece el cumplimiento no obligatorio de las restricciones débiles.

Desarrollo del software

Para el desarrollo del software se emplearon cuatro fases iterativas: análisis, diseño, desarrollo, pruebas e implementación; en el proceso se incluyeron técnicas y herramientas de ingeniería de software y CE.

a) Fase de análisis
La Fig. 1 muestra los actores principales del proceso de creación y asignación de horarios académicos (basado en la extensión de Erikkson y Penker (Hans-Erik & Magnus, 2000)). En el modelo se puede observar que, para realizar el proceso de creación y asignación de horarios académicos, se requiere información referente a profesores, estudiantes y materias. En el caso de los profesores, se requiere conocer los resultados de la evaluación docente (desempeño en clase). Además, se requiere la disponibilidad de horario para conocer los días y horas disponibles de los profesores. Otra información importante es el perfil profesional del profesor para conocer las asignaturas que pueden impartir. En cuanto a los estudiantes, se necesita conocer la cantidad de estudiantes por grupo, así como las asignaturas que se ofrecerán para un determinado periodo escolar.

Fig. 1: Modelo para la creación y asignación de horarios académicos.

Se definen tres tipos de actores: el coordinador de licenciatura, académico y coordinador de la UESVV. El coordinador de licenciatura analiza los perfiles de los profesores, a continuación el coordinador de la UESVV revisa la propuesta realizada por los coordinadores y posteriormente entrega la propuesta al académico para que publique los horarios. La Fig. 2 muestra el diagrama de casos de uso correspondiente al proceso de creación y asignación de horarios en el sistema (Fontela, 2016).

b) Fase de diseño
La Fig. 3 muestra el diagrama de clases del sistema para poder crear un horario grupal e individual: aulas de clase, profesores, asignaturas, grupos, tiempos y asignaturas. Las clases Individuo, Población y Ag contienen la información y métodos necesarios para ser utilizados por el AG.

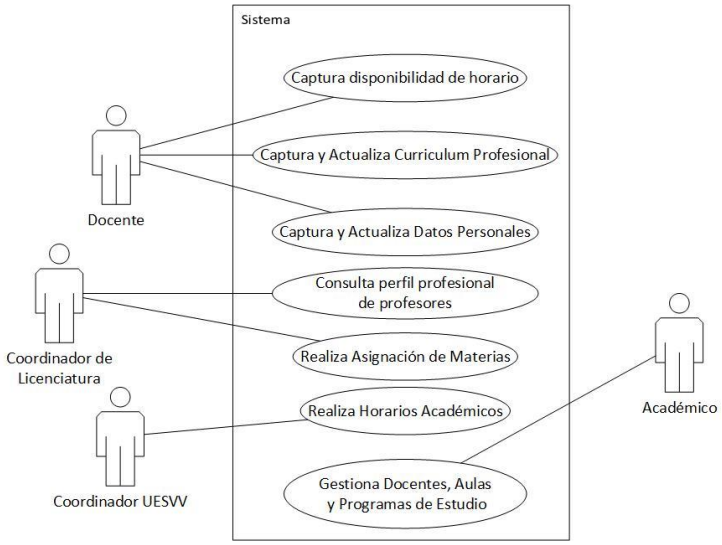

Fig. 2: Caso de uso del proceso de creación y asignación de horarios.

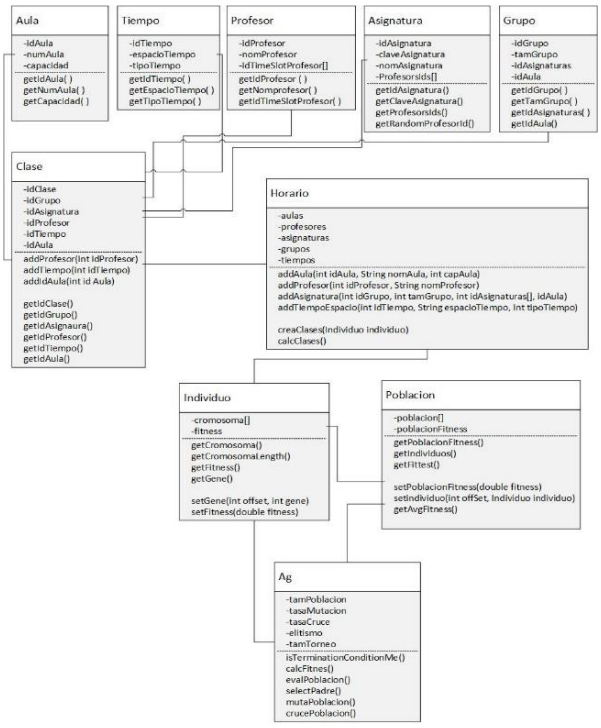

Fig. 3: Diagrama de clases.

c) Fase de desarrollo

Se emplean arreglos de objetos con las características principales de profesores, asignaturas, horas, días y aulas de clase. Un aula o salón de clase está compuesto por un identificador, un nombre de aula y la capacidad para albergar estudiantes (Tabla 1). Para la definición de los espacios de tiempo (de impartición de cursos) se utiliza un arreglo que contiene un número identificador clave y la especificación del periodo de tiempo (día/hora) (Tabla 2).

Tabla 1: Arreglo para un aula de clase

AULA		
ID_AULA	**NOM_AULA**	**CAP**
1	Aula 1	40
2	Aula 2	40
3	Aula 3	20
…	…	…

Tabla 2: Espacios de tiempo

TIEMPO	
ID_TIEMPO	**ESPACIO_TIEMPO**
1	Lunes 7:00 - 8:00
2	Lunes 8:00 - 9:00
…	…
48	Sábado 7:00 – 8:00
49	Sábado 8:00 – 9:00
…	…

Las asignaturas constan de identificador, clave, nombre de la asignatura y un identificador del profesor que impartirá el curso. La tabla profesor tiene definido un identificador, nombre y la disponibilidad de horario indicada por su id (Tabla 3).

Tabla 3: Asignaturas y profesores con disponibilidad de tiempos

ASIGNATURA			
ID_ASIG	**CLAVE_ASIG**	**NOM_ASIG**	**ID_PROF**
1	Clave 1	Matemáticas I	1
2	Clave 2	Probabilidad	2
3	Clave 3	Estadística	3
…	…	…	…

PROFESOR		
ID_PROF	**NOM**	**ID_TIEMPO**
1	Profesor x	1,2,3,4,5,10,11,12
2	Profesor y	29,30,31,32,33,34,35
3	Profesor z	1,2,3,4,5,10,11,12
…	…	…

Finalmente, a los grupos se les realiza una carga de asignaturas, para ello se tiene que asignar: un identificador al grupo, conocer el número de alumnos que lo conforman y las asignaturas que tendrán establecidas (Tabla 4).

Tabla 4: Grupo de alumnos

GRUPO		
ID_GRUPO	**TAM_GRUPO**	**ID_ASIG**
1	40	1,2,3,4
2	40	1,2,3,5
3 …	20 …	1,2,3,6
…	…	…

El cromosoma del AG se genera con los valores de las variables que representen a los objetos que forman parte de un horario de clases, definiendo a un individuo como se muestra en la Tabla 5; c_i corresponde al conjunto de cursos que se tiene necesidad de asignar recurso, p_x el profesor, t_y el espacio de tiempo y a_z el aula asignados.

Tabla 5: Cromosoma de un individuo

c_1	p_{x1}	t_{y1}	a_{z1}	c_2	p_{x2}	t_{y2}	a_{z2}	…	c_{Nc}	p_{xNp}	t_{yNt}	a_{zNa}

El proceso evolutivo inicia con la generación de una población definida de individuos. Posteriormente se calcula el valor de la función aptitud (Ecuación 1) de acuerdo al número de Violaciones de las Restricciones Fuertes (VRF) y al número Violaciones de las Restricciones Débiles (VRD) (Ecuaciones 2 y 3). Posteriormente se realiza la selección por torneo de los mejores individuos, los cuales son sometidos a un proceso de cruce uniforme que permitirá una combinación de genes para generar nuevos individuos. En el proceso de mutación se crea un nuevo individuo válido de forma aleatoria, se seleccionan algunos de sus genes los cuales son copiados al individuo a mutar (seleccionado); esta técnica permite asegurar que todos los individuos mutados sean válidos (véase Algoritmo 1).

Algoritmo 1. Procedimento de mutación

```
Inicio
    Inicializar una nueva población
    Para la población actual recorrer por aptitud
    Crear un individuo aleatorio para intercambiar genes
      Para un individuo recorrer sus genes
        Saltar la mutación si es un individuo de elite
        De lo contrario intercambiar un gen
      Fin para
    Agregar el nuevo individuo a la población
    Fin para
Fin
```

En el proceso de mutación se utiliza el Operador de Mutación Adaptativo (OMA); el OMA usa una tasa adaptativa que cambia en el tiempo para mejorar el proceso de búsqueda. El Algoritmo 3 muestra el procedimiento general que implementa OMA y el Algoritmo 2 calcula la aptitud media de la población.

Algoritmo 2: Aptitud promedio de la población.

```
Inicio
   Si la aptitudPoblación es igual a -1
      Entonces crear totalAptitud = 0
        Para toda la población
        totalAptitud = totalAptitud + Obtener la aptitud de cada individuo
        aptitudPoblación = totalAptitud
        retornar aptitudPoblación/tamaño de la población
    Fin
```

Algoritmo 3: Operador de Mutación Adaptativo

```
Inicio
    Inicializar una nueva población
    mejorAptitud = mejor aptitud de la población
    recorrer la población
    obtener el mejor individuo de la población
    parametroTasaMutación = tasaMutación
    Si la aptitud de un individuo es mayor a la aptitud promedio de la población
    entonces
       ap1 = mejorAptitud – aptitudIndividuo
      ap2 = mejorAptitud – aptitudPromedioPoblación
    parametroTasaMutación = (ap1 / ap2) * tasaMutación
      Para un individuo recorrer sus genes
        Saltar la mutación si es un individuo de elite
        De lo contrario intercambiar un gen
      Fin para
    Agregar el nuevo individuo a la población
    Fin para
    Retornar nuevaPoblación
Fin
```

Otra técnica utilizada en el proceso de mutación es el método de Recocido Simulado (RS). Con el uso de RS se busca reducir la tasa de mutación empezando con valores considerables los cuales van disminuyendo de forma paulatina para refinar la búsqueda (Gen & Cheng, 1999) (véase Algoritmo 4).

Algoritmo 4. Método de enfriamiento para tasa de mutación

```
Entrada: temperatura, tasaEnfriamiento
Inicio
    Temperatura = temperatura * (1.5 – tasaEnfriamiento)
Fin
```

RESULTADOS Y DISCUSIÓN

Para el desarrollo del prototipo de software se empleó el lenguaje de programación Java con el sistema operativo Microsoft® Windows 10 de 64 bits, en una computadora portátil con memoria RAM de 8gb, con procesador Intel Core i7 a una velocidad de reloj de 2.4 Ghz. En el proceso de experimentación se calibraron los parámetros del AG: tamaño de población, el método de selección (elitismo), tasa de cruce y mutación, y un tamaño de torneo para la selección. Para realizar la selección de los mejores parámetros de inicio, se tomó como referencia el desempeño del algoritmo conociendo de antemano que el óptimo global es 0.0. La configuración inicial del AG considerada es: tasa de mutación de 0.001 y tasa de cruce de 1.0. Todos los experimentos fueron repetidos 30 veces y se reporta la media de los valores obtenidos (por ejemplo configuraciones de AG, desempeño, entre otros.). La primera parte de la experimentación consistió en definir la tasa de cruza (con valores de 0.1 a 1.0) y mutación (con valores de 0.001 y 0.009) utilzando diferentes tamaños de población (de 100 a 500). La configuración que presentó mejor desempeño se muestra en la Tabla 6.

Tabla 6: Mejor configuración del AG

Parámetro	Valor
Tamaño de la Población	400
Tasa de Mutación	0.003
Tasa de Cruce	0.08
Elitismo	4
Tamaño del Torneo	12

La Fig. 4 muestra el desempeño del AG-OMA utilizando diferentes tamaños de población (desde 100 hasta 500 individuos). Como se puede observar, el desempeño del AG obtuvo mejores resultados con una población de 500 individuos, logrando alcanzar un desempeño de 0.1. La configuración utilizada en el AG-OMA se muestra en la Tabla 7.

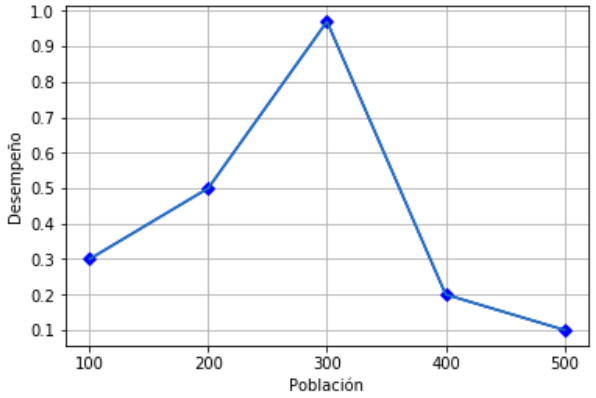

Fig. 4. Desempeño en contra del tamaño de la población del AG-OMA.

Tabla 7: Mejor configuración considerando el desempeño del AG-OMA.

Parámetro	Valor
Tamaño de la Población	500
Tasa de Mutación	0.009
Tasa de Cruce	1.0
Elitismo	5
Tamaño del Torneo	15

Para cada una de las estrategias se realizó el mismo proceso para seleccionar los mejores parámetros. La Tabla 8 concentra la mejor configuración de cada estrategia.

Tabla 8: Mejores configuraciones de AG, AG-OMA, AG-RS

Parámetro	AG	AG-OMA	AG-RS
Tamaño de la Población	400	500	400
Tasa de Mutación	0.003	0.009	0.008
Tasa de Cruce	0.08	1.0	0.08
Elitismo	4	5	4
Tamaño del Torneo	12	15	12

La Tabla 9 muestra el número de violaciones fuertes y débiles, el tiempo en segundo y el número de gereraciones utilizando las tres estrategias de optimización (AG, AG-OMA, AG-RS) cada estrategia con la mejor configuración.

Tabla 9: Resumen de desempeño de las estrategias de AG

Técnica	Violaciones de RF	Violaciones de RD	Tiempo en segundos	Generaciones
AG	0	61	50	169
AG-OMA	0	45	261	716
AG-RS	**0**	**41**	**163**	**431**

Como se puede observar todas las estrategias cumplieron con las restricciones de VRF y el AG-RS fue el que mejor desempeño mostró con las restricción de VRD. En referencia al tiempo de ejecución, el AG mostró mejor desempeño seguido de la estrategia AG-RS. La Fig. 5 muestra parte de un horario generado por el prototipo de software de acuerdo a los datos que fueron almacenados en la base de datos utilizando la estrategia AG-RS.

GRUPO	21CR121							
PROFESOR	MATERIA	SALON	LUNES	MARTES	MIERCOLES	JUEVES	VIERNES	SABADO
JAZMIN REYNA GUADARRAMA	DERECHO CONSTITUCIONAL	A13				16:00 -18:00	13:00 - 16:00	
SANDRA FLORES ATANACIO	EJECUCIÓN DE PENAS, BENEFICIOS Y SUSTITUTIVOS PENALES	A13			12:00 - 16:00			
SANDRA FLORES ATANACIO	HISTORIA Y TENDENCIAS DE LA CRIMINALIDAD	A13			17:00 -19:00	14:00 -16:00		
JOSE JUAN VALDES OLIVARES	INGLÉS II	A13		14:00-18:00			16:00 - 17:00	
ELEAZAR VALLE PINEDA	METODOLOGÍA DE LA INVESTIGACIÓN I	A13						
JOSE ANTONIO ALVA MEDINA	LABORATORIO DE QUÍMICA Y TOXICOLOGÍA	A13					09:00 - 13:00	
JOSE MANUEL RUIZ PADILLA	PROBABILIDAD Y ESTADISTICA	A13	15:00-18:00			12:00 - 14:00		

Fig. 5: Horario generado por el software.

Para la selección o adaptación de metodologías para el diseño y desarrollo de software se deben de tomar en cuenta las necesidades y características del problema a resolver con el fin de asegurar que el producto sea funcional y cumpla con las expectativas del usuario final; durante el desarrollo del software, que implementa un AG, se siguieron los procedimientos definidos en metodologías de ingeniería de software. El prototipo propuesto se puede extender, por ejemplo, se puede implementar el proceso de impresión de horarios y la manipulación de horarios después de haber sido generados. Además, se pueden agregar restricciones (fuertes o débiles), asignándoles un peso específico (definido) para que sean consideradas en el proceso de búsqueda. En comparación con otros prototipos de software, la presente propuesta se puede adaptar a distintas IES.

Un método para evitar los óptimos locales (horarios no óptimos) es realizar una adecuada configuración de los parámetros del AG; los principales parámetros son tasa de cruce, tasa de mutación y tamaño de la población. Se observó que con el uso de las estrategias adaptativas (AG-OMA y AG-RS) se mejora el proceso de búsqueda del óptimo global. Con el uso de RS se busca reducir la tasa de mutación para que el AG realice el proceso de búsqueda de manera controlada y pueda converger satisfactoriamente (la tasa de mutación disminuye hasta encontrar el mejor valor). Además, debe haber un compromiso entre el uso de los recursos computacionales, por ejemplo, con el uso de tamaños de población adecuada.

CONCLUSIONES

Con el desarrollo del software se logró minimizar el tiempo de diseño de horarios académicos para el caso de estudio propuesto; se cumplieron todas las restricciones fuertes y la mayoría de las restricciones débiles (al menos la mitad de las restricciones débiles definidas). El Algoritmo Genético implementa una configuración óptima de los parámetros de entrada y utiliza dos estrategias de mutación adaptativa (operador de mutación adaptativa y recocido simulado). El software fue implantado en la Universidad Mexiquense del Bicentenario (UMB) campus Unidad de Estudios Superiores Villa Victoria (UESVV) del Estado de México, México. Algunos de los trabajos futuros que se desprenden de esta investigación son la adaptación del prototipo para otras Unidades de Estudios Superiores de la UMB; realizar una revisión de todas las posibles restricciones (fuertes y débiles) que se aplican en las universidades, para generar una base de conocimiento que permita al usuario seleccionar y ponderar las restricciones fuertes y débiles; y la implementación de un módulo para la calendarización automática de exámenes.

AGRADECIMIENTOS

Los autores agradecen a las autoridades de la Unidad de Estudios Superiores Villa Victoria por facilitar la información relacionada en el proceso de creación y asignación de horarios académicos, necesaria para el desarrollo de esta investigación, así como al Consejo Mexiquense de Ciencia y Tecnología por el apoyo a través de sus programas de becas para llevar a cabo esta investigación.

REFERENCIAS

Abdullah, S., Burke, E.K. & McCollum, B. (2007). *A hybrid evolutionary approach to the university course timetabling problem*. In Evolutionary Computation 2007. CEC 2007. IEEE. pp. 1764-1768.

Arntzen, H. & Løkketangen, A. (2005). *A tabu search heuristic for a university timetabling problem*. In Metaheuristics: progress as real problem solvers (pp. 65-85). Springer, Boston, MA.

AscHorarios. *Generador de horarios escolares.* Disponible en: https://www.asctimetables.com/timetables_es.html [consultado en diciembre 2017].

Beligiannis, G.N., Moschopoulos, C.N., Kaperonis, G.P. & Likothanassis, S. D. (2008). Applying evolutionary computation to the school timetabling problem: The Greek case. *Computers & Operations Research, 35* (4), 1265-1280.

Chen, R.M. & Shih, H.F. (2013). Solving university course timetabling problems using constriction particle swarm optimization with local search. *Algorithms, 6* (2), 227–244.

Chmait, N. & Challita, K. (2013). Using simulated annealing and ant-colony optimization algorithms to solve the scheduling problem. *Computer Science and Information Technology, 1* (3), 208-224.

Coello, C.A. (2016). *Introducción a la Computación Evolutiva*. Instituto Politécnico Nacional no. 2508 México. Disponible en http://delta.cs.cinvestav.mx/~ccoello/compevol/apuntes.pdf [consultado en junio 2018].

Cubillos, M.A.G., Quiroga, E.H.P. & Ruiz, R.E.S. (2013). Problema del School Timetabling y algoritmos genéticos: una revisión. *Revista Vínculos, 10* (2), 259-276.

Daskalaki, S. & Birbas, T. (2005). Efficient solutions for a university timetabling problem through integer programming. *European Journal of Operational Research, 160* (1), 106-120.

Dino Matijas, V., Molnar, G., Cupic, M., Jakobovic, D. & Dalbelo Basic, B. (2010). University Course Timetabling Using ACO: A Case Study on Laboratory Exercises, volume 6276, chapter Knowledge-Based and Intelligent Information and Engineering Systems. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 100–110.

Flores, P., Brau, E., Monteverde, J., Salazar, N., Figueroa, J., Cadena, E., *et al*. (2004). Experimentos con Algoritmos Genéticos para resolver un problema real de Programación Maestros-Horarios-Cursos. *Revista iberoamericana de sistemas, cibernética e informática, 1*, 42-46.

Fontela, C. (2016). *UML Modelado de Software para Profesionales*. México. Alfaomega. ISBN: 978-987-1609-22-2

Gen, M. & Cheng, R. (1999). *Genetic Algorithms & Engineering Optimization*. John Wiley & Sons, Inc. ISBN:978-0471-3153-5

GOAL-*Group of optimization and algorithms*. Disponible en: http://goal.ufop.br/software/hstt/ [consultado en julio 2018].

Hans-Erik, E. & Magnus, P. (2000). *Business Modeling with UML*: Business Patterns at Work. OMG Press. John Wiley & Sons, 459 pag.

Haupt, R.L., Haupt, S.E. & Haupt, S.E. (2004). *Practical genetic algorithms*. Sec. Ed. New York: Wiley.

KHE. *A Software Library for High School Timetabling*. Disponible en: http://www.it.usyd.edu.au/~jeff/khe/ [consultado en julio 2018].

Lalescu L. *FET Free timetabling software.* Disponible en: https://lalescu.ro/liviu/fet/ [consultado en diciembre 2017].

Peñalara. *Software generador de horarios escolares automáticos.* Disponible en: https://www.penalara.com/es/mx [consultado en diciembre 2017].

Ponce, J.C., Torres, A., Quezada, F.S., Silva, A., Martínez, E., Casali, A., *et al*. (2014). *Inteligencia Artificial*. 1ra. Ed. Iniciativa Latinoamericana de Libros de Texto Abiertos. 225 pag.

Raghavjee, R. & Pillay, N. (2013). *A study of genetic algorithms to solve the school timetabling problem*. In Mexican International Conference on Artificial Intelligence (pp. 64-80). Springer, Berlin, Heidelberg.

Rodríguez, N., Martinez, J., Flores, J.J. & Graff, M. (2014). *Solving a scholar timetabling problem using a genetic algorithm-study case: Instituto Tecnológico de zitacuaro*. In Artificial Intelligence (MICAI), 2014 13th Mexican International Conference on, IEEE (pp. 197-202).

Unitime. *Comprehensive university timetabling system*. Disponible en: https://www.unitime.org/ [consultado en julio 2018].

UntisExpress. *Generador de horarios*. Disponible en: http://www.programahorario.com/index.php?lang=SP [consultado en diciembre 2017].