UNIVERSIDAD AUTÓNOMA DEL ESTADO DE MÉXICO

CENTRO UNIVERSITARIO UAEM VALLE DE MÉXICO

ALGORITMOS GENÉTICOS EN LA GENERACIÓN DE HORARIOS ESCOLARES

T E S I S

Que para obtener el Grado de

MAESTRO EN CIENCIAS DE LA COMPUTACIÓN

P r e s e n t a

Ing. Joan Manuel Broca

Tutor Académico:
Dr. Víctor Manuel Landassuri Moreno

Tutor(es) Adjunto(s):
Dr. Abel García Nájera
Dr. Héctor Rafael Orozco Aguirre

Atizapán de Zaragoza, Edo. de Méx. Noviembre 2016.

Dedicatoria

Dedico este trabajo a mi familia entera, deben saber que todos en alguna medida me han alentado a superarme.

Kevin, Hanzka y Alan, sirva este trabajo para que en algún momento entiendan que todo lo que se propongan con determinación, lo pueden llevar a cabo y evidencie que aunque haya malos momentos no se puede dejar de crecer.

Agradecimientos

Agradezco a la vida todos los recordatorios, expresados de muchas formas, día con día, de que el aprendizaje no tiene fin.

Agradezco enormemente a la planta docente de tiempo completo del Centro Universitario UAEM Valle de México por su dedicación y gusto de transmitir conocimiento de buen nivel, me siento muy orgulloso de formarme en esta Institución.

Quiero agradecer de manera muy especial al Dr. Víctor Landassuri por toda la ayuda, su tiempo y orientación durante mi estancia en esta Honorable Institución, también por encaminarme en un nivel de abstracción computacional superior pero aún más, por ampliar mis horizontes en cuanto al conocimiento a nivel global.

También agradecer al CONACyT por el apoyo económico que se me brindó durante los dos años de estudio, ya que sin él, sencillamente no estaría aquí. Al COMECyT por la ayuda concedida para la culminación de ésta tesis.

Resumen

La generación de horarios de clases en una Institución Educativa (IE) implica el problema de asignar recursos limitados a una serie de tareas relacionadas. Algunos de los recursos a considerar son: los periodos, la disponibilidad del profesor, las aulas, unidades de aprendizaje, entre otros. La complejidad radica en la cantidad de restricciones y los criterios con los que deben aplicarse. La búsqueda asistida de un horario que se ajuste a todas las condiciones se vuelve una tarea costosa en tiempo y recursos de cómputo que, al no resolverse, puede impactar a otras áreas de la Institución cuyas tareas dependen de la generación de horarios.

Los Algoritmos Evolutivos (AEs) han demostrado que pueden brindar solución a este tipo de problemas de una manera más eficiente que los procedimientos de búsqueda aleatoria. En particular, los Algoritmos Genéticos (AGs) proporcionan un enfoque metaheurístico haciendo uso de técnicas basadas en la teória de la evolución de las especies.

En este trabajo, se estudia la aplicación e implementación de un Algoritmo Genético, con el fin de realizar experimentos y comprobaciones acerca de la configuración de parámetros de ejecución en casos de estudio específicos.

Se pone en práctica la implementación de un Algoritmo Genético (AG) en un programa que sea capaz de generar a una solución factible con los recursos de cómputo disponibles y en un tiempo menor al que tardaría una búsqueda aleatoria o realización manual de un horario de clases en una IE.

Abstract

The generation of class schedules in an educational institution involves the problem of allocating limited resources to a number of related tasks. Some of the resources to consider are: periods, teacher availability, classrooms, learning units, among others. The complexity lies in the amount of restrictions and criteria that should be applied. Assisted search of a schedule that meets all conditions becomes a costly task in time and computing resources, if not resolved, can impact other areas of the institution whose tasks depends on the timetabling generation.

Evolutionary Algorithms (EA) have shown that they can provide solution to these problems in a more efficient way than random search procedures. In particular, Genetic Algorithms (GA) provide a meta-heuristic approach using techniques based on the theory of evolution of species.

In this paper, the application and implementation of a Genetic Algorithm is studied in order to perform experiments and checks about the configuration execution parameters in specific case studies.

The implementation of a simple Genetic Algorithm is performed in a computer program that is able to generate a feasible solution with available computing resources and in less time than it took a random search or manual realization schedule.

Índice general

Índice de figuras

Índice de tablas

Lista de Acrónimos

Capítulo 1

Introducción

1.1. Antecedentes

La solución de problemas del mundo real, se ha visto beneficiada cada vez más por algoritmos y técnicas computacionales, permitiendo tener resultados más precisos, más rápidos (al ser resueltos de forma más inteligente) o bien, para abrir nuevos paradigmas en la resolución de este tipo de problemas. Una de las áreas de las Ciencias Computacionales, es la Inteligencia Artificial (IA), la cual suele usar mecanismos matemáticos y algorítmicos inspirados en la naturaleza e inteligencia del ser humano.

La generación de horarios es un problema clásico en el ámbito de la computación ya que se trata de un problema NP-dificil [1] lo que significa que la cantidad de cómputo requerida para resolver el problema crece con el tamaño del mismo en forma exponencial. Este también es conocido como un problema Timetable (horario). Los timetable son problemas que tienen numerosas aplicaciones, por ejemplo pueden encontrarse en horarios de colegios y Universidades [2 - 9], actividades como el transporte [2], abastecimiento de agua [3], economía y finanzas [4].

En la literatura se ha demostrado que la primera forma de resolver problemas de horarios es con los algoritmos de búsqueda ya que éstos se pueden aplicar a casi cualquier problema, la desventaja principal radica en que este tipo de algoritmos requieren demasiado tiempo y recursos de cómputo cuando se trata de un problema NP-difícil en virtud de que el algoritmo clásico realizará una búsqueda exhaustiva por todas las posibles soluciones gastando exageradamente recursos de cómputo.

Los algoritmos de búsqueda informada que utilizan funciones heurísticas para dirigir la búsqueda y reducir el tiempo para encontrar una buena solución, transmiten el conocimiento acerca del dominio del problema. Las funciones heurísticas se aplican a un cierto espacio de búsqueda y regresan un resultado, el cual permitirá estimar el valor del estado con respecto al objetivo (o meta), en tal sentido son más eficientes ya que se aprovechan de la retroalimentación, además de que una buena heurística puede superar drásticamente el rendimiento de cualquier búsqueda desinformada. Por ejemplo, en el juego del sudoku, donde hay un tablero de 9 x 9 en el cual se deben acomodar los numeros del 1 al 9 sin que éstos se repitan a lo largo de las filas o las columnas, una búsqueda de fuerza bruta haría una ejecución por todas las posibilidades, por el contrario la búsqueda informada se acercaría al objetivo o meta haciendo uso del conocimiento propio del problema. Nótese que en problemas complejos el objetivo es encontrar buenas soluciones en un periodo de tiempo razonable y no soluciones óptimas. En la búsqueda heurística se utiliza el conocimiento específico del dominio del problema, por esta razón se puede considerar como débil, pero puede ser muy efectiva si se aplica correctamente.

En la mayoría de los problemas hay forma de evaluar cuál es una buena solución, es decir se puede encontrar un máximo, un mínimo o una evaluación que satisfaga una determinada **función objetivo**, esto se conoce como una tarea de optimización.

La optimización es un tema resuelto por la investigación de operaciones siempre y cuando se conozca con precisión la función a evaluar, cuando no se cuenta con ello se habla de que la función tiene que darse en términos de optimización parametrizada haciendo que la solución calculada se acerque lo más posible a la función objetivo.

Muchos problemas en el campo de la inteligencia Artificial (IA) pueden ser modelados como problemas de satisfacción de restricciones y son resueltos a través de búsquedas, donde las restricciones son relaciones lógicas entre variables y para encontrar una solución se deben evaluar los valores además de satisfacer todas las restricciones. Los problemas del mundo real son muy complejos dado que: el número de soluciones posibles es demasiado grande para una búsqueda exhaustiva, la función objetivo puede ser ruidosa o puede cambiar en el tiempo por lo que se requiere no solo una solución sino todo un conjunto de soluciones y las soluciones posibles son tan fuertemente restringidas que la construcción de una solución factible es muy difícil.

Inicialmente, se puede seguir una serie de procedimientos o estrategias de ataque al problema conocidas como heurísticas, Zankis y Evans [5] las definen como:

"procedimientos simples, a menudo basados en el sentido común, que se supone ofrecerán una buena solución (aunque no necesariamente la óptima) a problemas difíciles, de un modo fácil y rápido"

Las metaheurísticas consisten en un proceso repetitivo de tal forma que dada una solución conseguida con una heurística, ésta se evalua para comprobar si cumple con los objetivos, puede ser modificada y el proceso se repite hasta que se obtenga una solución que sea factible.

Entonces, las metaheurísticas son métodos para resolver una clase general de optimización de problemas, no requieren de conocimiento previo sobre la función objetivo ya que tratan a las funciones objetivo como "cajas negras", obtienen el conocimiento de un problema de optimización a partir de estadísticas obtenidas de las soluciones posibles.

Así, los Algoritmos Evolutivos (AEs) son técnicas metaheurísticas para la solución de problemas de búsqueda y optimización. Tienen su principal inspiración en la teoría de la evolución de las especies, por tanto hay que especificar dos interpretaciones para dicho término: primero, se usa con frecuencia para describir algo que cambia gradualmente con el tiempo y segundo, tienen una relación estrecha con conceptos biológicos donde se describe un sistema evolutivo, que cambia de generación en generación a través de la variación entre la selección y la reproducción. Esta noción Darwiniana del cambio evolutivo es la idea central de los AEs.

No es objetivo de la computación evolutiva encontrar óptimas soluciones, sin embargo sirve para aproximar a una **solución factible**, de hecho pueden encontrar soluciones muy cercanas al óptimo global con técnicas que simulan el comportamiento evolutivo de la naturaleza.

Esta investigación resolverá el problema de la generación de horarios en forma automática, para un conjunto de problemas encontrados en la lieratura los cuales se detallan en la siguiente sección, utilizando de forma puntual técnicas de algoritmos genéticos que proporcionen soluciones que sean factibles en un tiempo aceptable en comparación con una búsqueda desinformada y con los recursos de cómputo de que se disponga.

1.2. Planteamiento del problema

En toda Institución educativa al inicio de cada periodo escolar se presenta la necesidad de asignar los espacios físicos, recursos humanos y materiales a los diferentes grupos de

estudiantes que deben cumplir con un cierto plan curricular.

Esta tarea puede tomar de varios días de trabajo manual dependiendo de la cantidad de datos que se deben manejar y la complejidad que suponen las diversas variables involucradas en el proceso.

Para explicar a detalle el problema de la generación de horarios de clase, se puede comenzar con un ejemplo trivial: se consideran 2 Grupos, 2 Profesores y 2 Materias, éstas se van a calendarizar de lunes a viernes, en cuatro periodos posibles. En este caso, de forma arbitraria, se establece una relación entre Profesor-Grupo-Materia (P,G,M) la cual es una lista ordenada de elementos que se define como **tupla**. El requerimiento es que cada tupla se tiene que encontrar 3 veces durante la semana. La Fig. 1.1a muestra un posible horario válido ya que todas las condiciones se cumplen, la Fig. 1.1b también es una solución válida. Como se puede observar, este caso particular del problema es trivial en virtud de que los

	Lunes	Martes	Miércoles	Jueves	Viernes
Periodo 1	P1G1M1	P2G2M2	P1G1M1	P2G2M2	
Periodo 2	P1G1M1	P2G2M2			
Periodo 3					
Periodo 4					

(a) Un horario válido.

	Lunes	Martes	Miércoles	Jueves	Viernes
Periodo 1	P1G1M1	P2G2M2			
Periodo 2	P1G1M1	P2G2M2			
Periodo 3	P1G1M1	P2G2M2			
Periodo 4					

(b) Otro horario válido.

Figura 1.1: Ejemplos de horarios.

posibles horarios se pueden ajustar de muchas formas válidas dado que carece de restricciones. Inicialmente es importante mencionar que este problema es no determinista ya que dadas las mismas entradas hay varias soluciones posibles. Otro dato importante es que si se cree que las tuplas pueden acomodarse de forma aleatoria en los bloques y se supone que nunca habría choques entre tuplas, en ambos ejemplos el tiempo computacional seria calculado en función de la cantidad de bloques que se deben recorrer hasta terminar las posibles tuplas y se trataría de un problema lineal.

Sin embargo, los problemas reales son mucho más complicados que el ejemplo anterior ya que se pueden agregar varias restricciones al problema, e.g. una de ellas puede ser que un grupo no puede tomar dos clases en el mismo dia al mismo tiempo, si al generar un horario se evalua su factibilidad, se puede detectar si algun grupo causa alguna colisión o choque de alguna de las restricciones específicas.

En el ejemplo del horario trivial una búsqueda aleatoria seguramente provocaría choques y aún es posible calcular matemáticamente el polinomio de resolución. Al agregar más restricciones al problema se complicará el cálculo del tiempo computacional.

Cuando se calcula la complejidad de un problema mediante un polinomio y en la ejecución del algoritmo el tiempo de solución es mayor que el calculado por el polinomio, se considera que el problema es NP-Hard [1].

Para calcular las posibles permutaciones en el ejemplo trivial, primero: se tienen 5 días por 4 periodos igual a 20 espacios disponibles para asignar. Segundo: si se considera la asignación de una sola tupla entonces se tienen 20 posibilidades de asignarla, pero si hay 2 tuplas entonces la cantidad de posibilidades se calcula considerando que para la primera tupla tiene 20 posibilidades y, una vez acomodada, la siguiente tupla tendría cabida en alguno de los 19 espacios restantes, por lo tanto, con 2 tuplas habría: 20 x 19 = 380 permutaciones. Tercero: si se considera que existen igual número de tuplas que número de espacios (n) entonces el número de permutaciones es el factorial de dicho número de tuplas o de espacios (n!).

Luego entonces, si el ejemplo trivial contiene 20 espacios y se tienen que acomodar 20 tuplas resulta que la cantidad de permutaciones es: 20!. El resultado es cercano a 2.5 trillones de posibilidades, que en tiempo computacional es muy costoso. Además se hace notar

que se está considerando un solo salón de clases.

Bajo esta perspectiva, el espacio de búsqueda en un problema pequeño resulta muy extenso, adicionalmente se tiene que considerar el hecho de que se deben evaluar ciertas restricciones inherentes a cada problema ya que el cálculo anterior proporciona un panorama de todas las posibilidades, sin embargo, no todas ellas son soluciones factibles.

Algunos ejemplos de restricciones son: i) es imposible que un profesor pueda impartir clases en dos lugares al mismo tiempo, ii) que, debido a políticas en la Institución Educativa, un profesor puede tener un límite de horas a impartir o iii) que no puedan existir clases contiguas de 2, 3, o n horas, iv) también que el profesor no puede impartir cualquier materia, etc. El cumplimiento de las restricciones necesarias determina que éste es un problema de optimización y de manera más específica se trata de un problema de minimización, ya que se busca una solución que infrinja la menor cantidad de restricciones del problema con el fin de encontrar un horario factible.

Para la experimentación se considera conveniente utilizar un conjunto de datos público, éste ha sido objeto de estudio por otros trabajos encontrados en [6, 7], dicho conjunto es ajeno a esta investigación, es obtenido de la Universidad de Brunel en Londres y es están categorizados por el PhD. J.E. Beasley [8] como "Problemas difíciles de horarios". Se trata de cinco problemas de horarios con diferentes niveles de dificultad (Tipo 4, Tipo 5, Tipo 6, Tipo 7 y Tipo 8). Se explica a continuación, el más simple de ellos con el fin de comprender la configuración de todos los archivos que conforman el conjunto de datos.

Hacer referencia al problema Tipo 4 implica pensar en cuatro profesores (P1, P2, P3, P4), cuatro clases (C1, C2, C3 y C4) y cuatro salones (S1, S2, S3 y S4), el archivo de requerimientos viene dispuesto en una matriz donde las columnas representan a los profesores

y las filas a los grupos, al tratarse del problema Tipo cuatro se agrupan matrices de 4 x 4 para formar los cuatro salones. La Tabla 1.1 muestra el contenido de dicho archivo de requerimientos en una tabla que contiene los cuatro salones con sus respectivos cuatro Profesores y cuatro Clases, se hace hincapié en que la descripción de los problemas nos hablan de Clases-Materias, para efectos de la investigación se tratarán como Materias o Unidades de Aprendizaje (UDAs). El resto de los problemas se pueden observar en el apéndice A.

Tabla 1.1: Horario Tipo 4

HT4	P1	P2	P3	P4	
C1	2	2	1	2	
C2	1	1	1	2	
C3	1	1	1	6	
C4	2	2	3	2	S1
C1	2	5	1	2	
C2	0	4	3	2	
C3	1	2	1	0	
C4	2	2	1	2	S2
C1	2	1	1	2	
C2	0	0	5	1	
C3	2	1	4	1	
C4	6	1	2	1	S3
C1	3	1	2	1	
C2	1	4	1	4	
C3	3	3	2	1	
C4	2	0	1	1	S4

De la Tabla 1.1 se especifica que:

$$P = \text{Profesores}$$

$$C = \text{Clases}$$

$$S = \text{Salones}$$

El número que aparece en cada posición de la tabla es la cantidad de periodos (o sesiones) que la tupla se debe encontrar durante la semana. Además se especifica que para todos los problemas la cantidad de periodos a la semana (horas) es de 30, por ende, durante la semana de lunes a viernes se tienen 6 horas / clase cada día.

Por lo tanto el espacio de búsqueda para los ejemplos, que son objeto de esta investigación, se calcula de la siguiente forma:

$$EP = Per! * Prof! * C! * S!$$

Donde se especifica que Per = 30, Prof = 4, C = 4 y S = 4. Entonces, el caso del problema Tipo 4 el cálculo da como resultado: $EP = 3.6 \times 10^{36}$ un número muy considerable para realizar todas las combinaciones de forma exhaustiva.

Finalmente, se precisa que los problemas de Tipo 5, 6, 7 y 8, son de un espacio de búsqueda mucho mayor y por lo tanto mucho mas costosos en tiempo computacional. Tambien se hace hincapié en que de acuerdo a la descripción de los problemas en la página oficial, los renglones de la matriz estan denominados como “Clase”, para efectos de esta investigación se interpreta como materia o como una Unidad de Aprendizaje (UDA).

1.3. Objetivos

1.3.1. Objetivo General

Como se ha mencionado hasta este punto de la investigación, el reto de la generación de horarios es no tener ningún choque de horario dadas las restricciones de cada problema, por lo tanto, como objetivo general se persigue:

"Resolver mediante un algoritmo genético problemas de generación automática de horarios de clases en problemas considerados por la literatura [1, 6, 7, 8, 9] como difíciles (NP-hard)".

1.3.2. Objetivos Específicos

1. Minimizar cantidad de restricciones violentadas por los horarios generados como posibles soluciones, a través de las generaciones para los problemas estudiados en el conjunto de datos.

2. Encontrar soluciones factibles mediante la modelación y programación de un Algoritmo Genético, en una cantidad de tiempo conveniente en comparación con el tiempo que tardaría una búsqueda aleatoria.

3. Evaluar diversas configuraciones del algoritmo genético para establecer cual brinda soluciones factibles con menores tiempos de ejecución, así como ajustarlo de forma precisa para resolver el problema, en términos de: el tamaño de la población, el porcentaje de reproducción, de mutación y la cantidad de generaciones necesarias para que la solución converja a soluciones sin choques.

1.4. Hipótesis

"Mediante un algoritmo genético estándar se puede resolver el problema de la generación de horarios de clase ajustando correctamente sus parámetros de evolución como lo son: el tamaño de población, porcentajes de mutación - cruzamiento y la cantidad de generaciones".

1.5. Justificación

Debido a que el problema de horarios es un problema con un espacio de búsqueda complejo, que persigue un objetivo y que esta sujeto a restricciones (problema de optimización), se decide su solución mediante la implementación de un AG.

Las bases teóricas de los algoritmos genéticos son vastas, el estado del arte muestra su aplicación a una gran cantidad de problemas reales que aunque son muy concretos, muestran el potencial de esta técnica.

Además, los algoritmos genéticos brindan gran flexibilidad para su adaptación y aplicación en problemas del mundo real, anteriormente se encontraban limitados a que la modelación de los problemas se ajustara a la codificación binaria (genes). Esto ya no es así y para el caso de los problemas de horarios, los Algoritmos Genéticos (AGs) brindan la posibilidad de modelarlos e implementarlos mediante codificaciones enteras [10], es decir, sin tener que ajustar al problema a términos binarios.

También se considera que, al no encontrar fuentes que muestren los parámetros de ejecución de los AGs como lo son: el tamaño de la población, la cantidad de generaciones de ejecución, los porcentajes en la reproducción y técnicas adicionales como el uso de elitismo. En este documento se presenta información oportuna acerca de cómo dichos parámetros son de crucial importancia en la solución de problemas de horarios escolares.

1.6. Delimitación del problema

El algoritmo genético se diseñará para resolver un conjunto de datos encontrados en la Universidad de Brunel en Londres, que se explicaron en la sección 1.2. El uso de este conjunto se decide debido a que es una fuente referida los trabajos [6, 7]. Como se mencionó, la información se encuentra abierta al público y son problemas diseñados para este tipo de estudios con técnicas heurísticas.

El conjunto consta de 5 problemas fundamentales:

1. Problema de 4 profesores, 4 grupos y 4 materias a impartirse en 4 salones.
2. Problema de 5 profesores, 5 grupos y 5 materias a impartirse en 5 salones.
3. Problema de 6 profesores, 6 grupos y 6 materias a impartirse en 6 salones.
4. Problema de 7 profesores, 7 grupos y 7 materias a impartirse en 7 salones.
5. Problema de 8 profesores, 8 grupos y 8 materias a impartirse en 8 salones.

Los 5 problemas requieren la asignación de tuplas en 30 espacios (5 días por seis periodos). En todos los casos los profesores imparten 30 horas a la semana.

Cabe destacar que está fuera del alcance de esta investigación, implementar otro tipo de algoritmos evolutivos para los mismo efectos. Así mismo, no se incursionará en ningún otro tipo de técnicas matemáticas para optimizar los horarios, y el trabajo únicamente estará centrado en el desempeño de los algoritmos genéticos.

1.7. Distribución del trabajo

El presente trabajo está distribuido en 4 partes fundamentales:

1. El panorama teórico se detalla en el capitulo 2, además se muestra el estado del arte con investigaciones y sus técnicas aplicadas para la resolución del problema.

2. La elaboración del modelo matemático, codificación y funcionamiento del algoritmo genético que permitirá generar los horarios, son mostrados en el capítulo 3.

3. El capítulo 4 presentará experimentos preeliminares y los resultados de la experimentación, éste contendrá los métodos y técnicas estadísticas utilizados para fundamentar el funcionamiento del algoritmo genético, así como para mejorar su evolución.

4. Conclusiones y trabajo futuro se mostrarán en el capítulo 5.

Capítulo 2

Marco Téorico

2.1. Búsqueda y Optimización

De manera general existen dos tipos de problemas computables: la primera clase puede ser resuelta de forma determinista, en este caso el éxito está garantizado y se sabe que se trata de un problema de computación de un cierto orden polinomial, es decir, algoritmos que siempre producirán los mismos resultados dadas las mismas entradas y que invariablemente tardarán un número n de unidades de tiempo que podrán calcularse mediante un polinomio, como se explicó en la sección anterior.

Sin embargo, el mundo real tiene problemas que no pueden resolverse de forma determinista, estos problemas sólo se pueden resolver mediante la búsqueda de una solución, es decir, se tiene que generar un algoritmo para realizar la búsqueda de una solución que se aproxime al problema en cuestión, y este tipo de casos representan la segunda clase de problemas en computación. La Inteligencia Artificial (IA) se ocupa de este tipo de problemas.

La forma de obtener una solución de esta segunda clase, es mediante un proceso de generación de soluciones, las cuales se caracterizan por un conjunto de objetivos, de objetos

y de operaciones. Estos conjuntos pueden ser imprecisos y cambiantes durante la solución del problema, por ejemplo, el problema del sudoku (donde hay que rellenar un tablero de 9 x 9 de manera que cada renglón, columna y cajas de 3 x 3 contengan los números del uno al nueve), además dan lugar al espacio del problema a partir de la combinación de los operadores con cualquier combinación de objetos. Así, el espacio del problema puede contener una o más soluciones válidas.

De acuerdo a la literatura [11], la primera forma para atacar espacios de problemas muy grandes consiste en realizar búsquedas exhaustivas (también llamadas de fuerza bruta), éstas se realizan de manera no informada y por lo tanto no tienen ningún tipo de aprendizaje. Para problemas complejos, como los NP-completos, los algoritmos tradicionales no son capaces de encontrar una solución dentro de un cierto límite de tiempo.

Como se menciona anteriormente, los algoritmos de búsqueda informada que utilizan funciones heurísticas para dirigir la búsqueda y reducir el tiempo para encontrar una buena solución, transmiten el conocimiento acerca del dominio del problema. Al iniciar una función heurística, ésta se localiza en una región del espacio de búsqueda, al evaluarse, se regresa un valor de adaptabilidad el cual nos permite saber que tan lejos estamos del objetivo global a alcanzar, así se puede considerar una herramienta más eficiente, dado que se cuenta con una retroalimentación, además de que una buena heurística puede superar drásticamente el rendimiento de cualquier búsqueda desinformada. En el caso del sudoku, una búsqueda de fuerza bruta haría una ejecución por todas las posibilidades, por el contrario la búsqueda informada se acercaría al objetivo o meta haciendo uso del conocimiento propio del problema. Nótese que en problemas complejos el objetivo es encontrar buenas soluciones en un periodo de tiempo razonable y no soluciones óptimas. En la búsqueda heurística se utiliza el conocimiento específico del dominio del problema, por esta razón se puede considerar como débil, pero puede ser muy efectiva si se aplica correctamente.

En la mayoría de los problemas hay forma de evaluar cuál es una solución factible, es decir se encuentra un máximo, un mínimo o una evaluación que satisfaga una determinada función heurística, esto se conoce como una tarea de optimización.

La optimización es un tema resuelto por la investigación de operaciones siempre y cuando se conozca con precisión la función a evaluar, cuando no se cuenta con ello se dice que la función tiene que darse en términos de optimización parametrizada (restricciones) haciendo que la solución calculada se acerque lo más posible a la función objetivo.

Muchos problemas pueden ser modelados como problemas de satisfacción de restricciones y son resueltos a través de búsquedas, donde las limitaciones son relaciones lógicas entre variables y para encontrar solución se deben evaluar los valores y deben satisfacer todas las restricciones. Se entiende que los problemas del mundo real son más complejos dado que: el número de soluciones posibles es demasiado grande para una búsqueda exhaustiva, la función objetivo puede ser ruidosa o puede cambiar en el tiempo, lo cual implica no solo a una solución sino todo un conjunto de soluciones, y las soluciones posibles son tan fuertemente restringidas que la construcción de una solución factible es muy difícil.

Las metaheurísticas [5] son métodos para resolver una clase general de optimización de problemas, no requieren de conocimiento previo sobre la función objetivo ya que tratan a las funciones objetivo como "cajas negras", obtienen el conocimiento de un problema de optimización a partir de estadísticas obtenidas de las soluciones posibles.

Como se menciona en la sección anterior el problema de horarios es un problema con un espacio de búsqueda complejo, puede contar con varias restricciones y se trata de llegar a una función objetivo, por lo tanto, la aplicación de evolutivos para este tipo de problemas resulta muy conveniente. A continuación se presentan sus fundamentos teóricos.

2.2. Téoria de la Evolución Biológica

Los Algoritmos Evolutivos (AEs) parten de las ideas del modelo de evolución natural que fue propuesto por Charles Darwin en 1859 [12]. De acuerdo con la teoría de Darwin la evolución de las especies se debe al principio de selección natural, que favorece la supervivencia y multiplicación de aquellas especies que están mejor adaptadas a las condiciones de su entorno. Otro elemento que Darwin señaló como relevante para la evolución son las mutaciones, o pequeñas variaciones que introducen diferencias en las características físicas y tipos de respuesta de los padres y los hijos. El mecanismo que forza la actuación de la selección es la producción de descendencia. Mientras hay abundancia de recursos, la población crece exponencialmente. Este proceso lleva a situaciones de escasez de recursos en el entorno, en las que los individuos "mejor adaptados" al medio tienen mayor probabilidad de sobrevivir y de dejar descendencia.

La genética y las leyes de la herencia genética han complementado la teoría de Darwin con mecanismos relativos a la herencia de características físicas en la producción de descendencia, dando lugar a la teoría del neodarwinismo. De acuerdo con esta teoría, las características físicas de un individuo, su fenotipo, son la consecuencia de su información genética o genotipo, cadenas de genes con complejas interacciones, que constituyen las unidades de transferencia de la herencia. Los genes pueden modificarse puntualmente por mutaciones. La replicación de las cadenas de genes en la reproducción no siempre es perfecta. A veces, aunque con una frecuencia extremadamente baja, se producen errores en el proceso de copia que constituyen mutaciones. Sin embargo, existen mecanismos reparadores de estos errores, como enzimas codificadas en los propios genes, que reducen aún más el porcentaje de este tipo de mutaciones. También existen factores externos, como la radiación y ciertas sustancias químicas, que pueden incrementar de forma significativa las probabilidades de mutación.

La selección tiene lugar sobre los individuos, que son la consecuencia del genotipo y su interacción con el medio, constituyendo las unidades de selección. Lo que evoluciona es el conjunto de individuos que constituyen la población, que representa a un conjunto de genes comunes a sus individuos. La adaptación de su individuo es su tendencia, relativa al resto de los individuos de la población, para sobrevivir y dejar descendencia en unas condiciones ambientales específicas.

Estas ideas están en la base del diseño de los algoritmos evolutivos. Además, muchas de las propiedades de la evolución de los seres vivos, como la edad, la mayor o menor tendencia a la mutación según el estado de la evolución, etc., están siendo objeto de investigación para su incorporación a las técnicas de computación evolutiva. Sin embargo, los algoritmos evolutivos no tratan de ser un reflejo fiel de la evolución biológica. Se debe tener en cuenta que la naturaleza evoluciona a lo largo de millones de años, mientras que a nosotros nos interesa que nuestros algoritmos nos proporcionen una solución en un tiempo algo más corto.

2.3. Algoritmos Evolutivos (AEs)

De acuerdo con Cervigón [13] "Los Algoritmos Evolutivos (AEs) son técnicas metaheurísticas para la solución de problemas de búsqueda y optimización. Tienen su principal inspiración en la evolución, por tanto hay que especificar dos interpretaciones para dicho término: primero, se usa con frecuencia para describir algo que cambia gradualmente con el tiempo, y segundo, tienen una relación estrecha con conceptos biológicos donde se describe un sistema evolutivo, que cambia de generación en generación a través de la variación entre la selección y la reproducción". Esta noción Darwiniana del cambio evolutivo es la idea central de los AEs.

Desde un punto de vista convencional, un Algoritmo Evolutivo (AE) es un algoritmo que

simula un sistema evolutivo Darwiniano. De manera específica un AE incluye:

1. Una o más poblaciones de individuos compitiendo por recursos limitados.
2. Estas poblaciones cambian dinámicamente debido al nacimiento y muerte de individuos
3. Una noción de aptitud que refleja la habilidad de un individuo de sobrevivir y reproducirse.
4. Una noción de reproducción variada: los descendientes se parecen mucho a los padres pero no son idénticos.

Como en la biología, en el contexto de los AEs, la representación precedente es referida como genotipo y después como fenotipo. En la Figura 2.1, se presenta un ejemplo para una cadena genotipo donde cada gen puede tomar valores binario cero o uno, de acuerdo a su posición cada gen brinda un valor para conformar el fenotipo correspondiente.

Posición Gen	Gen 8	Gen 7	Gen 6	Gen 5	Gen 4	Gen 3	Gen 2	Gen 1	Gen 0
Potencias de 2	256	128	64	32	16	8	4	2	1
Genotipo	**1**	**0**	**0**	**1**	**1**	**0**	**0**	**1**	**1**
suma encendidos	256	+	32	+	16	+	2	+	1
Fenotipo	**307**								

Figura 2.1: Ejemplo de un fenotipo generado a partir de un genotipo.

En la evolución de las especies se comprende que los genes tienen propensión a la reproducción (cruzamiento), los AEs implementa dicha característica como se ilustra en la Figura 2.2, donde dados dos individuos, con sus respectivos genotipos, se establece un punto de cruce y se lleva a cabo el intercambio de información genética dando como resultado la generación de nuevos individuos que preservan la información genética a traves de las generaciones.

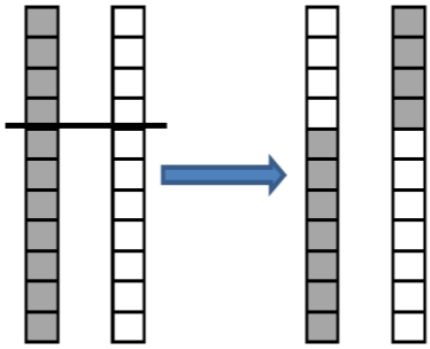


Figura 2.2: Ejemplo de cruzamiento de genes entre dos individuos.

También es indicado por Darwin, que los genes son suceptibles de las mutaciones en sus genes como se muestra en la Figura 2.3, donde un gen determinado de forma aleatoria puede cambiar el valor de su gen y por lo tanto afectar a su fenotipo.

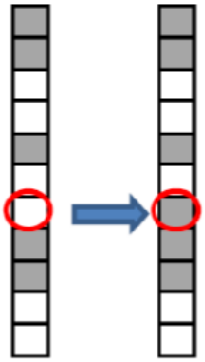


Figura 2.3: Ejemplo de una posible mutación a nivel de genes.

2.3.1. Esquema de un AE

Los distintos algoritmos evolutivos que se pueden formular responden a un esquema básico común, y comparten una serie de propiedades:

- Procesan simultáneamente, no una solución al problema, sino todo un conjunto de ellas. Estos algoritmos trabajan con alguna forma de representación de soluciones po-

tenciales al problema, que se denominan individuos. El conjunto de todos ellos forma la población con la que trabaja el algoritmo.

- La composición de la población se va modificando a lo largo de las iteraciones del algoritmo que se denominan generaciones. De generación en generación, además de variar el número de copias de un mismo individuo en la población, también pueden aparecer nuevos individuos generados mediante operaciones de transformación sobre individuos de la población anterior. Dichas operaciones se conocen como operadores genéticos.

Cada generación incluye un proceso de selección, que da mayor probabilidad de permanecer en la población y participar en las operaciones de reproducción a los mejores individuos.

Los mejores individuos son aquellos que dan lugar a los mejores valores (ya sean mínimos o máximos) de la función de adaptación del algoritmo. Es fundamental para el funcionamiento de un algoritmo evolutivo que este proceso de selección tenga un componente aleatorio, de forma que individuos con baja adaptación también tengan oportunidades de sobrevivir, aunque su probabilidad sea menor.

Es este componente aleatorio el que dota a los algoritmos evolutivos de capacidad para escapar de óptimos locales y de explorar distintas zonas del espacio de búsqueda. En la Figura 2.4 se muestra el esquema general de un algoritmo evolutivo de acuerdo con Blum [14]:

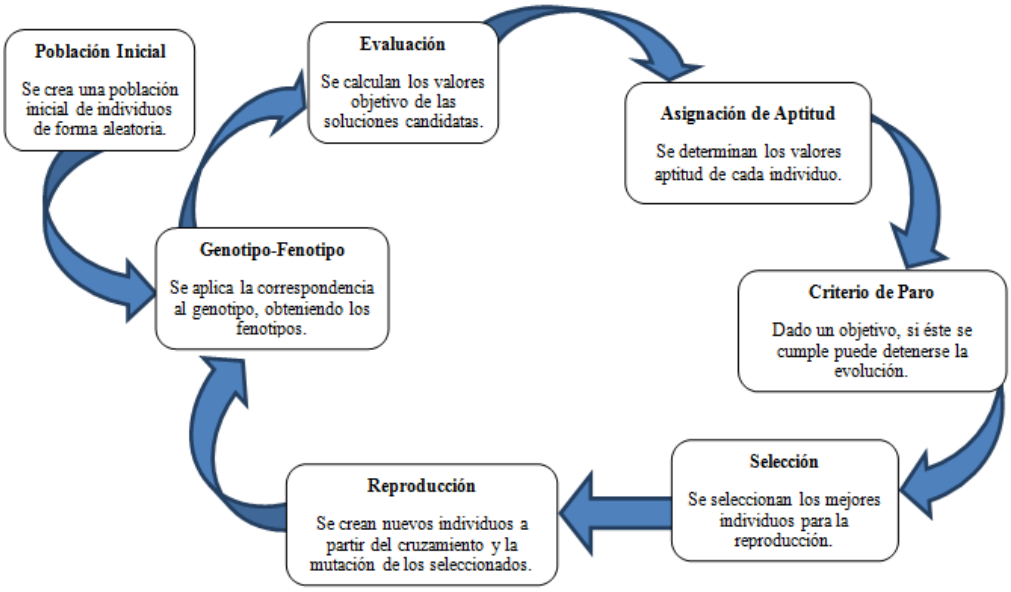


Figura 2.4: Ciclo básico de los AE's.

Históricamente, científicos de la computación e ingenieros han considerado los principios de la evolución para resolver problemas de optimización. A continuación se enlistan algunos de los AEs más utilizados en aplicaciones del mundo real [15]:

1. Algoritmos Genéticos (AG) [16], que se abordan con detalle en la sección 2.4.

2. Programación Genética (PG) [17]: tiene dos significados posibles. En primer lugar, se puede ver como AEs que producen programas, algoritmos y construcciones similares. En segundo lugar, se usa para incluir AEs que usan estructuras de datos árboles como genotipos.

3. Estrategias Evolutivas (EE) [10, 18, 19]: Es una técnica de optimización heurística basada en ideas de adaptación y evolución, los espacios de búsqueda usualmente consiste en vectores de números reales, aunque las cadenas de bits y cadenas de enteros también son comunes. La mutación y la selección son los principales operadores y la reproducción se usa con menos frecuencia. La desviación estándar de un conjunto de números aleatorios distribuidos se utiliza como parámetro para la mutación.

4. Programación Evolutiva (PE) [20, 21]: Se encuentra menos definida en comparación con otros AEs. Hay una diferencia semántica, mientras que en otros AEs los individuos de una especie son soluciones candidatas, en PE una solución candidato es una especie en si misma. Por lo tanto la mutación y selección son los únicos operadores utilizados en PE, el cruzamiento no se utiliza. El esquema de selección utilizado en PE es muy similar al de EE.

5. Algoritmos Meméticos (AM) [22]: Es un algoritmo de optimización hibrido compuesto por un marco evolutivo y un algoritmo de búsqueda local, activados en el ciclo de la generación del marco evolutivo. El "meme" es una idea, una "unidad de transmisión cultural", la unidad básica del conocimiento. La idea de estos algoritmos es transmitir esta información cultural entre generaciones, en el entorno computacional ésta transferencia se lleva a cabo mediante búsquedas locales combinadas con el marco evolutivo, a través de su interacción armónica se pueden obtener mejores soluciones que cooperan a la detección del óptimo global.

6. Inteligencia de Enjambre (IDE) [23, 24]: Se ocupa del diseño de algoritmos para solucionar problemas distribuidos inspirados en el comportamiento colectivo de insectos o animales sociales. Dos de los más populares son: la optimización por partículas y la optimización por colonia de hormigas [14, 25]. Otros representantes son: los basados en el forrajeo de las abejas y los de asignación de tareas en colonias de avispas.

2.4. Algoritmos Genéticos (AGs)

A principios de los sesentas, John H. Holland propone los "planes reproductivos", más tarde en 1975 denominados AGs [16], en los que la motivación principal fue el aprendizaje de máquina. Su AG enfatiza la importancia de la cruza sexual (operador principal) sobre el de la mutación (operador secundario) y usa selección probabilística.

Gran parte de los problemas que surgen en el desarrollo industrial y en la investigación pueden formularse como una búsqueda o como una optimización: dado un sistema, se busca un conjunto de valores que permiten llevarlo a una determinada configuración o bien un conjunto de valores que permiten optimizar su comportamiento (rendimiento, calidad, costo, etc.).

El método clásico de optimización para problemas cuyo espacio de búsqueda de soluciones es continuo es la técnica de escalada, que consiste en determinar la pendiente de la vecindad del punto actual y seleccionar el punto de mayor pendiente en dicha vecindad. Si el valor de la función a optimizar en el nuevo punto es mejor que en el anterior, el nuevo punto se convierte en el punto actual. El proceso continúa hasta que no es posible realizar ninguna mejora. Una limitación de este método es su incapacidad para escapar de óptimos locales. Considérese la Figura 2.5, si la exploración comienza en un punto como p, sólo será capaz de llegar al máximo a, pero no podrá llegar a b, ya que para ello tendrá que atravesar la región c de valores peores que a. En este caso el espacio de búsqueda ofrece la solución en el mismo "plano" por lo que se habla de una función con espacio convexo.

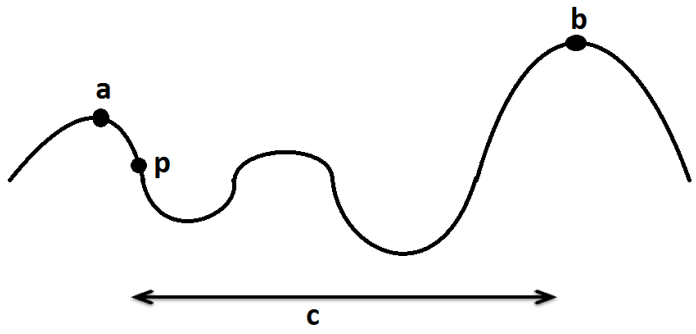


Figura 2.5: Gráfica clásica de optimización, usando la técnica de la escalada.

Por otro lado se debe considerar la existencia de espacios no convexos, como el de la Figura 2.6, en este caso la tarea de búsqueda no es tan sencilla, sin embargo los AGs han probado que funcionan sin importar el tipo de espacio de búsqueda.

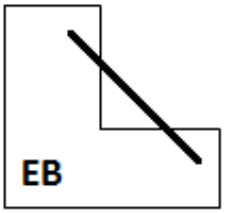


Figura 2.6: Espacios de búsqueda no convexos.

Los algoritmos genéticos (AGs) son una técnica de resolución de problemas de búsqueda y optimización inspirada en la teoría de la evolución de las especies y la selección natural propuesta por Charles Darwin, mencionada en la sección 2.2. El pensamiento evolutivo actual gira en torno al Neo-Darwinismo el cual establece que toda vida puede explicarse mediante cuatro procesos: Reproducción, Mutación, Competencia y Selección. Además, las características físicas de un individuo, su fenotipo, son la consecuencia de su información genética o genotipo, cadenas de genes con complejas interacciones, que constituyen las unidades de transferencia de la herencia. Los genes pueden modificarse puntualmente por mutaciones.

La selección tiene lugar sobre los individuos, que son la consecuencia del genotipo y su interacción con el medio, constituyendo las unidades de selección. Lo que evoluciona es el conjunto de individuos que constituyen la población, que representa a un conjunto de genes comunes a sus individuos. La adaptación de un individuo es su tendencia, relativa al resto de los individuos de la población, para sobrevivir y dejar descendencia en unas condiciones ambientales específicas.

Sin embargo, los algoritmos evolutivos no tratan de ser un reflejo fiel de la evolución biológica. Se debe tener en cuenta que la naturaleza evoluciona a lo largo de millones de años, mientras que en los AGs interesa que los algoritmos proporcionen una solución en un tiempo mucho más corto.

Para poder aplicar un algoritmo genético se requiere de los componentes siguientes:

- Una representación de las soluciones potenciales del problema.
- Una forma de crear una población inicial de posibles soluciones (normalmente un proceso aleatorio).
- Una función de evaluación que desempeñe el papel del medio ambiente, donde se desarrollan los individuos, ponderando las soluciones en términos de su "aptitud", es decir, que tan adaptado está cada individuo para solucionar el problema en forma correcta.
- Operadores genéticos que alteren la composición de los hijos que se producirán para las siguientes generaciones mediante la reproducción (cruzamiento y mutación).
- Valores para los diferentes parámetros que utiliza el Algoritmo Genético (tamaño de la población, porcentaje de cruzamiento, porcentaje de mutación, número máximo de generaciones).

La forma canónica de los genéticos se muestra en la Figura 2.7.

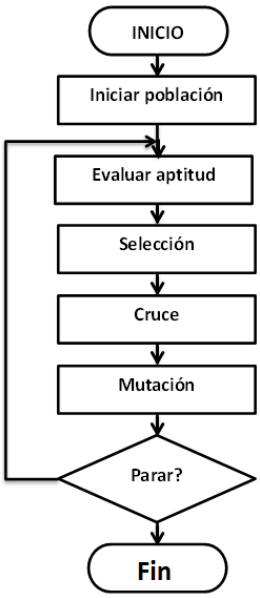


Figura 2.7: Forma canónica de un AG.

2.4.1. Representación de los individuos

Como se mostró en la Figura 2.1, en el AG simple los individuos son cadenas binarias (también pueden ser números enteros), que se denotarán por **bits**, éstos representan a puntos **x** del espacio de búsqueda del problema. Tomando la nomenclatura de la biología, los bits conforman el genotipo del individuo y a **x** se le denomina fenotipo. La nomenclatura biológica a veces se adopta también para otros datos del individuo. Así, se usa gen para referirse a la codificación de una determinada característica del individuo. En los AGs se suele identificar un gen con cada posición de la cadena binaria, aunque esto no tiene por qué ser siempre así. Se usa alelo para los distintos valores que puede tomar un gen y locus para referirse a una determinada posición de la cadena binaria.

2.4.2. Generación de la población inicial

Los individuos de la población inicial de un AG suelen ser cadenas de ceros y unos generadas de forma completamente aleatoria, es decir, se va generando cada gen con una función que devuelve un cero o un uno con igual probabilidad. En algunos problemas en los que se disponga de información adicional que permita saber de antemano que determinadas cadenas tienen más probabilidades de llegar a ser solución, es posible favorecer su generación al crear la población inicial. Sin embargo, es imprescindible para el buen funcionamiento del AG dotar a la población de suficiente variedad para poder explorar todas las zonas del espacio de búsqueda.

2.4.3. Grado de adaptación de los individuos

La evolución de la población depende de la calidad relativa de los individuos que compiten por aumentar su presencia en la población y por participar en las operaciones de reproducción. En un problema de búsqueda u optimización, dicha calidad se mide por la adecuación o adaptación de cada individuo a ser solución al problema. Es frecuente que

los problemas se presenten como la optimización de una función matemática explícita. En dichos casos la función de adaptación coincide con la función a optimizar.

Sin embargo, a veces se realizan algunas transformaciones a la función a optimizar o función de evaluación $\mathbf{g(x)}$ para transformarla en una función de adaptación adecuada $\mathbf{f(x)}$. Se denominará adaptación bruta de un individuo $\mathbf{x}$ a $\mathbf{g(x)}$, y simplemente adaptación a $\mathbf{f(x)}$.

2.4.4. Condiciones de terminación

Es necesario especificar las condiciones en las que el algoritmo deja de evolucionar y se presenta la mejor solución encontrada. La condición de terminación más sencilla es alcanzar un determinado número de generaciones de evolución. Otras condiciones, que a veces se utilizan de forma combinada, son alcanzar una solución de una determinada calidad o detectar que la mayor parte de la población ha convergido a una forma similar, careciendo de la suficiente diversidad para que tenga sentido continuar con la evolución.

2.4.5. El proceso de selección

La población del algoritmo genético se somete a un proceso de selección que debe tender a favorecer la cantidad de copias de los individuos más adaptados. Este proceso se puede realizar de formas muy diferentes como se presentan a continuación.

2.4.5.1. Selección proporcional o por ruleta

El porcentaje de selección p_i de un individuo con este método es proporcional a su adapta-

ción relativa:

$$p_i = \frac{f(i)}{\bar{f}}$$

siendo $\bar{f}$ la adaptación media de la población y $f(i)$ la adaptación del individuo.

Es necesario generar un número aleatorio de acuerdo con la distribución de probabilidad dada por las p_i. Si se cuenta con un generador de números aleatorios que genera números de forma uniformemente distribuida a lo largo de un intervalo como [0,1], se puede seguir el siguiente procedimiento [13]:

- Se definen las puntuaciones acumuladas de la siguiente forma:

$$q_0 := 0$$

$$q_i := p_1 + ... + p_i \qquad (\forall i = 1, ..., n)$$

  Donde $q_0 := 0$ es el primer individuo, q_i es cada uno de los individuos subsecuentes hasta el enésimo elemento y p_i es la puntuación calculada de cada individuo.

- Se genera un número aleatorio $a \in [0, 1]$
- Se selecciona al individuo i que cumpla:

$$q_{i-1} < a < q_i$$

- Este proceso se repite para cada individuo que se desee seleccionar.

2.4.5.2. Muestreo estocástico universal

Es un procedimiento similar al de muestreo por ruleta, pero en este caso se genera un solo número aleatorio a y a partir de él se generan los k números que se necesitan (para generar k individuos) espaciados de igual forma. Los números se calculan de la siguiente

forma:

$$a_j := \frac{a+j-1}{k}(_{\forall j=1,\ldots,k)}$$

Una vez generados estos números, el método funciona de la misma forma que la selección por ruleta.

2.4.6. El proceso de reproducción: los operadores genéticos

En cada nueva generación se crean algunos individuos que no estaban presentes en la población anterior. De esta forma el algoritmo genético va accediendo a nuevas regiones del espacio de búsqueda. Los nuevos individuos se crean aplicando ciertos operadores genéticos a individuos de la población anterior. Los operadores que suelen estar presentes en todo algoritmo genético son los operadores de cruce y mutación. El operador de cruce combina propiedades de dos individuos de la población anterior para crear nuevos individuos. El operador de mutación crea un nuevo individuo realizando algún tipo de alteración, usualmente pequeña, en un individuo de la población anterior. Estos operadores pueden adoptar formas muy distintas. Se verá a continuación la forma más simple de estos operadores.

Para cada operador genético se establece una tasa o frecuencia, de manera que el operador sólo se aplica si un valor generado aleatoriamente está por encima de la tasa especificada. Por ejemplo, si la tasa de aplicación del operador de cruce es del 40 %, entonces si al generar un número aleatorio entre 0 y 1 se obtiene 0.56, que es mayor que 0.4, se aplica el operador de cruce a los dos individuos seleccionados a tal efecto.

2.4.6.1. Operador de cruce monopunto

Como se explicó en la sección 2.3, la forma más simple del operador de cruce en los algoritmos genéticos es el cruce monopunto. Consiste en seleccionar al azar una única posición en la cadena de ambos padres e intercambiar las partes de los padres divididas por

dicha posición, como muestra la Figura 2.2. Este operador produce dos hijos que combinan propiedades de ambos padres, lo que puede llevar a una mejora de la adaptación de los hijos respecto a la de los padres.

2.4.6.2. Operador de mutación aleatoria

La forma más sencilla de mutación que se puede aplicar a un individuo de un algoritmo genético consiste en cambiar el valor de una de las posiciones de la cadena: si es cero pasa a uno, y si es uno pasa a cero. Como se explicó en la Figura 2.3. Para el caso de cromosomas con números enteros se elige aleatoriamente un alelo y el gen es remplazado por por un número generado aleatoriamente dentro del rango de los valores válidos para el cromosoma.

Para cada posición por individuo se comprueba la tasa de mutación, y si se cumple se aplica el operador. Habitualmente la tasa de aplicación del operador de mutación es bastante pequeña (en torno al 0.1 %) comparada con el operador de cruce. Sin embargo, esto no es una norma general.

En muchos casos la mutación produce individuos con peor adaptación que los individuos originales, ya que la mutación puede romper las posibles correlaciones entre genes que se hayan formado con la evolución de la población. Sin embargo, contribuyen a mantener la diversidad de la población, que es fundamental para el buen funcionamiento del algoritmo.

2.4.6.3. Operador de mutación por intercambio

Para problemas de tipo combinatorio, este es uno de los métodos mas utilizados, consiste en seleccionar dos puntos al azar del individuo e intercambiar los valores de dichas posiciones.

2.4.7. El proceso de reemplazo

Habitualmente los AGs mantienen un tamaño de población constante, aunque existen otras posibilidades. Para mantener el tamaño de la población, los nuevos individuos creados mediante los operadores genéticos deben reemplazar a otros de la población anterior. En función de la cantidad de individuos reemplazados en la población anterior se consideran distintos tipos de AGs:

- AGs generacionales: en estos algoritmos la población se renueva por completo de una generación a otra.
- AGs con estado estacionario: la descendencia de los individuos seleccionados en cada generación se incluye en la población, reemplazando a algunos de los individuos de la población anterior. En este caso se conserva parte de la población de generación en generación.

Para los AGs con estado estacionario existen los siguientes criterios de reemplazo:

- Reemplazo de los padres: los hijos sustituyen a sus padres.
- Reemplazo aleatorio: los individuos a eliminar se eligen aleatoriamente. El número de individuos a eliminar viene dado por el tamaño de la descendencia, que a su vez queda definido por las tasas de cruces y mutaciones y el tamaño de la población.
- Reemplazo de los individuos peor adaptados: los individuos a eliminar se eligen aleatoriamente, pero sólo entre los que tiene el valor de adaptación más bajo. Valores bajos de adaptación se suelen considerar por debajo del 10 % de la adaptación media.

2.5. Tratamiento de problemas con restricciones

Los problemas de satisfacción de restricciones se formulan sobre un conjunto de variables, cada una de las cuales toma valor en un rango específico y sobre las que se definen

un conjunto de restricciones. El objetivo de un problema de este tipo puede ser hallar un valor para cada variable, dentro de un rango específico, de forma que se satisfagan todas las restricciones. También puede ser el buscar los valores para las variables y que, además de satisfacer las restricciones, optimice alguna función objetivo (problema de optimización). Dependiendo del problema, los dominios de las variables pueden ser continuos o discretos.

Dado que los algoritmos evolutivos tienen aplicaciones con muchos problemas del mundo real, se han desarrollado diversos trabajos para atacar los problemas que suponen la satisfacción de restricciones, que buscan alguna serie de valores con una función objetivo y que pueden además perseguir una función de optimización.

2.5.1. Técnicas básicas para el tratamiento de restricciones

Tsang y Rossi [26, 27] proponen algunas técnicas para el tratamiento de problemas con restricciones, a continuación se describen 3 técnicas que aplican a los AEs.

2.5.1.1. Técnicas de penalización

Son las más generales, ya que pueden aplicarse a cualquier problema con restricciones. Consisten en generar soluciones para el problema ignorando las restricciones y penalizar después en la evaluación a aquellas soluciones que no cumplan con las restricciones del problema. A menudo la función de penalización depende del grado de la violación de la restricción, es decir, es alguna función (logaritmo, exponencial, etc.) del grado de la violación. En ocasiones también se hace que la penalización cambie a medida que avanza la evolución, de manera que al comienzo del proceso haya más permisividad de soluciones que violan las restricciones, y a medida que se acerca al final de la evolución la penalización se incremente.

2.5.1.2. Técnicas de reparación

Son aquellas en las que se busca algún mecanismo para corregir las soluciones que violan las restricciones del problema. Estas técnicas son específicas para cada problema y en general son difíciles de encontrar.

2.5.1.3. Técnicas de codificación

Consisten en buscar una representación especial y adecuada para el problema que garantice que se cumplen las restricciones. Al igual que las técnicas de reparación, son específicas de cada problema y son difíciles de encontrar.

2.6. Estado del Arte

Para efectos de esta investigación, se han buscado trabajos en la literatura que ya han estudiado la generación de horarios y se han encontrado diversas publicaciones cuyos resultados generales se presentan a continuación.

La mayoría de las investigaciones encontradas tratan como sinónimo el concepto de algoritmo evolutivo con el de algoritmo genético, también se dedican a resolver problemas de Instituciones específicas. Se observa que cuando los problemas implican un conjunto de restricciones a optimizar se tiende a usar la solución por Algoritmos Genéticos (AGs).

En un trabajo de Suárez[28], se encuentra la implementación de un AG para la generación de horarios en una Universidad, se modeló el problema en términos de las restricciones propias de la Institución utilizando la técnica clásica de algoritmos genéticos. Se refieren resultados aceptables que cumplen con las necesidades descritas, no hay contribución alguna a las técnicas actuales y la conclusión general es que se disminuyó el tiempo dedicado a la generación de horarios en la Universidad, comparado contra la generación manual de

dichos horarios.

Por otro lado, en [29] de Colorni, se realiza una comparación contra un producto de software comercial de generación de horarios cuyo nombre no se menciona. Tras la implementación de la técnica clásica de AG, el autor concluye que su algoritmo es 12.3 % más eficiente que el producto de software comercial y con un resultado de adaptabilidad (fitness) del 92 %, en este caso el trabajo está totalmente apegado a las necesidades de organización de horarios en escuelas de educación básica.

Otra aplicación de un AE para horarios de examen de Universidad [9] de Raghavjee, se utiliza también la técnica clásica de AGs, el autor concluye que su implementación encuentra soluciones aceptables al optimizar horarios, resalta el hecho de que el tiempo de ejecución de su sistema es en promedio de 10 minutos por lo que su uso puede resultar muy conveniente si se requiere obtener resultados muy rápidos.

La investigación de Von Lücken [30], no está basada en generación de horarios, pero se trata de un estudio de Algoritmos Evolutivos MultiObjetivo (MOEAs) cuya mayor aportación es el estudio de dichos algoritmos en ambientes paralelos. El autor concluye que los algoritmos que utilizan elitismo muestran mejor desempeño en comparación con lo que no utilizan elitismo (algo que es de esperarse). Además se realiza una comparación de implementaciones de algoritmos reconocidos y el resultado es que los algoritmos CNSGA-II y NSGA-II son mayormente beneficiados por el paralelismo. Propone un algoritmo nuevo MOEA que evita la pérdida o el remplazo de la peor solución, permitiendo el mantenimiento de puntos alejados y previniendo la convergencia en un frente Pareto óptimo local. Por último menciona que la incorporación del método de búsqueda por islas mejora las soluciones obtenidas pero menciona que es necesario encontrar una manera óptima de incorporar la información en las diferentes islas.

También se observan trabajos de generación de horarios con otras técnicas evolutivas. En [31] Granada presenta la implementación de un algoritmo memético para la generación de horarios el cual se compara contra otras técnicas evolutivas como son: Simulated Annealing y Búsqueda tabú. La conclusión del autor es que su algoritmo memético registra mejores resultados en tiempo y afirma mejores resultados (factibles) en comparación con el algoritmo memético propuesto por Chu-Beasley y deja abierta la puerta a la incorporación de técnicas multiobjetivo para su algoritmo.

Por otro lado se revisa [32] de Öner para horarios en una Universidad utilizando un algoritmo híbrido mediante la estrategia de coloración de nodos y un algoritmo de colonia artificial de abejas, los resultados demuestran que el algoritmo hibrido proporciona resultados eficientes para el caso particular de la Universidad donde se llevó a cabo el estudio.

En el mercado hay numerosas aplicaciones que brindan la posibilidad de generar horarios por software, entre las más populares:

- FET (Free Timetabling Software) [33]. Un programa de código abierto para generación de horarios, se menciona que ante horarios complejos el programa podría tomar horas en resolverlo, no se menciona la implementación de alguna técnica de IA. La interfaz de usuario no es amigable y por lo tanto complica la introducción de problemas.

- ASC Horarios [34]. Se trata de un software con costo desde $250 usd para primarias, hasta $1995 usd para la versión professional. No se menciona la implementación de alguna técnica heurística, el sitio oficial menciona que la búsqueda se realiza entre miles de millones de posibilidades por lo que, se infiere que realiza búsqueda exhaustiva.

- MIMOSA Scheduling Software [35]. Es una aplicación con costo por licencia desde 500 euros, tampoco menciona la aplicación de alguna técnica especial.

- Scientia Timetabling Scheduling [36] y Wise Timetable [37] son productos que no mencionan el costo del producto, solo permiten la descarga del demo para después contactar la compra. No se menciona la aplicación de técnicas de IA.

Se puede observar que la mayoría de las investigaciones encontradas hasta el momento, estudian casos particulares y hay pocas aportaciones a las técnicas existentes. De hecho, no fue posible encontrar información precisa acerca de los parámetros de configuración del evolutivo. Se busca dicha información con respecto al tamaño de la población, el número de generaciones, cantidades para el porcentaje de cruza y mutación. Todos ellos parámetros necesarios para la ejecución del algoritmo y no se pudo encontrar información clara del detalle fino de las ejecuciones.

Es por ello que uno de los objetivos en este trabajo es proporcionar el detalle de los parámetros recomendados que arroja la experimentación.

Capítulo 3

Modelado del problema

3.1. Definición de restricciones

Inicialmente, se debe definir una serie de restricciones obligatorias con las que el algoritmo debe cumplir para brindar soluciones factibles. En tal sentido Raghavjee [9] y Larrosa [38] define que hay dos tipos de condiciones que se deben satisfacer en los horarios generados, la violación de alguna de las condiciones origina un horario no válido.

- Restricciones obligatorias: que son aquellas que deben de cumplirse.
- Restricciones deseables: son aquellas que denotan preferencias del usuario y que se desea que se cumplan en la medida de lo posible.

En este caso el conjunto de datos referido en la sección 1.2 exige las siguientes restricciones:

- Un curso o UDA debe tener asignado a lo más un docente en un aula en un periodo específico.
- Un docente debe tener asignado a lo más un curso en un aula en un periodo específico.
- Un aula puede tener a lo más un curso asignado y un docente en un periodo específico.
- Una UDA debe cumplir con una cantidad de horas semanales requeridas.

- Un docente debe impartir una determinada cantidad de horas semanales señalada en los requerimientos.

3.1.1. Modelo General

Como se mencionó en la sección 1.1 se utilizarán AGs como técnica metaheurística para resolver el problema de generación de horarios de un período escolar, para ello, es necesario plantear el problema en términos de un modelo del problema.

3.1.1.1. Conjunto de datos del modelo

Sea $C = \{1, ..., c\} \subseteq \mathbb{Z}$ el conjunto de unidades de aprendizaje, donde cada número está asociado a una unidad de aprendizaje.

Sea $T = \{1, ..., t\} \subseteq \mathbb{Z}$ el conjunto de periodos de tiempo en el que se puede dictar un curso cualquiera $c \in C$, donde cada número esta asociado a un horario a lo largo del día.

Sea $D = \{1, ..., 7\} \subseteq \mathbb{Z}$ el conjunto de días de la semana. De antemano se sabe que, los problemas objetos del estudio solo consideran los días de lunes a viernes, sin embargo, el modelo esta preparado para considerar clases en cualquier día de la semana.

Sea $P = \{1, ..., p\} \subseteq \mathbb{Z}$ el conjunto de profesores que puede dictar un curso cualquiera $c \in C$ donde cada número está asociado a un profesor.

Sea $S = \{1, ..., s\} \subseteq \mathbb{Z}$ el conjunto de salones donde se puede dictar un curso cualquiera $c \in C$ donde cada número también esta asociado a un salón.

3.1.1.2. Parámetros

Se deben definir los siguientes parámetros que corresponden a cada uno de los recursos involucrados:

$$i = \text{cantidad de salones},$$

$$j = \text{cantidad de días},$$

$$k = \text{cantidad de periodos}$$

$$l = \text{cantidad de profesores},$$

$$m = \text{cantidad de unidades de aprendizaje},$$

De estas definiciones se detalla que:

1. La cantidad de salones, profesores y unidades de aprendizaje estará en función del tipo de problema que se esté ejecutando.
2. Para el conjunto de datos estudiado la cantidad de días son 5 (Lunes - Viernes).
3. La cantidad de periodos para todos los casos es de 6 periodos por día.

3.1.1.3. Variables de decisión

Entonces, se tienen variables conformadas por valores de tipo entero, de tal manera que:

$$\begin{aligned} X_{ijklm} &\in \{\ S\ x\ D\ x\ T\ x\ P\ x\ C\ \} \\ &\forall\ i \in S,\ j \in D,\ k \in T,\ l \in P\ y\ m \in C \end{aligned} \tag{3.1}$$

Donde, x denota el producto cartesiano de los conjuntos.

Mediante un arreglo, se almacena el valor entero de cada una de las variables formando tuplas las cuales reciben el tratamiento de cromosomas, éste se detallará más adelante

en la sección 3.1.2.1

3.1.1.4. Modelado de las restricciones del problema

Este modelo está planteado en función de las restricciones obligatorias dadas en la sección 3.1 y se definen a continuación:

$$x_{ijklm} = \{\ (i,j,k,i,m) \in X \mid i \in S,\ j \in D,\ k \in T,\ l \in P\ y\ m \in C\ \} \tag{3.2}$$

Donde x_{ijklm} es un elemento del conjunto X definido en la formula 3.1.

Todo curso (UDA) m que se dicta en un día j y periodo específico k debe tener asignado solo un docente l en un aula i:

$$|x_{ijklm}| = 1 \tag{3.3}$$

Todo profesor l que enseña en un día j y en un periodo específico k debe tener asignada solo una aula i, donde se especifica que, esta restricción es independiente de la UDA a impartir por lo que se omite la variable m.

$$|x_{ijkl}| = 1 \tag{3.4}$$

Todo curso (UDA) m debe cumplir con una cantidad de horas semanales H, en este caso la restricción es independiente del profesor y del salón a impartir, por ende, se omite la variable l y la variables i respectivamente.

$$|x_{jkm}| = H \tag{3.5}$$

Todo profesor l debe cumplir con una cantidad de horas semanales HP, esta restricción es independiente de la UDA m y del salón i por lo cual se omiten.

$$|x_{jkl}| = HP \tag{3.6}$$

3.1.1.5. Función objetivo global

La función global (fg) depende de los resultados de las funciones locales (fl) de cada individuo.

$$fg(x) = \sum_{x=1}^{N} fl(x) \tag{3.7}$$

Donde x representa cada elemento de la matriz y N representa el numero total de elementos en una matriz de 4 dimensiones.

3.1.1.6. Función objetivo local

$$fl(x) = CAPMSDP(x) + CALHSP(x) + CALHSU(x) \tag{3.8}$$

Donde $CAPMSDP$ representa los conflictos entre las tuplas salón, día y periodo con el mismo profesor, $CALHSP$ representa los conflictos en el límite de horas por semana para los profesores y $CALHSU$ representa problemas en el límite de periodos por UDA.

3.1.1.7. Criterio de optimización

Una vez definidas la función objetivo global y local, se establece la optimización de la siguiente forma:

$$Min(fg(x) = 0) \tag{3.9}$$

Por tanto, se establece que la función objetivo busca minimizar la cantidad de conflictos que pueden ocurrir en un horario, el criterio de horario factible se da cuando un individuo no tiene conflictos es decir que los conflictos son igual a cero sin embargo, el AG puede brindar algún horario que no este minimizado a cero pero que sea lo más factible encontrado.

3.1.2. Diseño del algoritmo

3.1.2.1. Representación del problema

Para la programación de los horarios es necesaria la definición del problema en términos de un AG, primero se define el cromosoma que estará representado como se muestra en la siguiente figura:

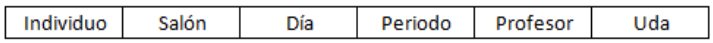


Figura 3.1: Representación del cromosoma para el problema de horarios.

Como se puede apreciar el cromosoma tiene la información necesaria para la conformación de las tuplas, sin embargo, se debe considerar que las parejas formadas a partir del Profesor-UDA tienen que almacenarse en un arreglo que pueda almacenar los k periodos del problema de acuerdo a los requerimientos, así mismo, se debe contemplar la creación de una matriz que logre el almacenamiento de los k periodos en los j días, a su vez, toda la información debe almacenarse por cada uno de los i salones.

Hasta este punto se tiene un arreglo de tres dimensiones y aún falta considerar que se trata de un solo individuo, por ende, se considera la creación de una dimensión adicional en la matriz para almacenar a todos los individuos de la población del AG.

De acuerdo con Goldberg [39], el AG simple puede codificarse con cromosomas que utilicen variables de decisión de tipo binario, pero también es posible hacerlo mediante números enteros, en este caso, para el cromosoma descrito, la codificación se lleva a cabo con enteros.

3.1.2.2. Evaluación de la aptitud

De acuerdo a la fórmula para la función de optimización 3.9, lo que se busca con el AG es minimizar la cantidad de restricciones que se violentan durante la evolución.

Se aplica la técnica de penalización descrita en la sección 2.5.1.1, por tanto, se establece que cada vez que se encuentre conflicto con alguna restricción, el fitness se incrementará, dado que se busca minimizar, provocando que el más apto sea aquel individuo que tenga menos violaciones a las restricciones.

3.1.2.3. Operadores genéticos

Durante la etapa evolutiva del algoritmo genético se utilizan los dos operadores clásicos de reproducción: la cruza y mutación. Adicionalmente se utiliza el elitismo en virtud de que Michalewicz [17] asegura que es posible obtener mejores resultados durante la evolución ya que se preserva el(los) mejor(es) individuo(s) durante toda la evolución.

Los operadores funcionan de acuerdo a los porcentajes que tienen asociados y a un mecanismo de selección. Cuando la población de individuos se evalúa, se realiza un proceso de selección por ruleta donde los más adaptados tienen mayor oportunidad de dejar descendencia. Para el cruzamiento se eligen dos padres dando lugar a dos hijos que reemplazan a los padres seleccionados. La mutación no intercambia genes, ni tiene selección, en este caso únicamente se elige algún individuo probabilísticamente, se fijan dos puntos de intarcambio y se intercambian sus genes, como se dijo en la sección 2.4.6.3.

Cabe mencionar que dichos operadores no modifican a aquellos individuos que se seleccionaron para el elitismo y que el AG es de tipo generacional donde los individuos recién creados, mas los que vienen del elitismo, remplazan por completo a la población actual.

3.1.2.4. Criterio de paro

Se establece al AG la posibilidad de detenerse tras un número de generaciones de evolución y también existe la posibilidad de detenerlo cuando la adaptación de algún individuo

cumple con el criterio de optimización que se encuentra estrechamente ligado a la función objetivo global, cabe recordar que la función de optimización debe minimizar la cantidad de conflictos en las restricciones.

3.1.3. Algoritmo

Para la solución del problema de generación de horarios se utiliza el siguiente procedimiento: Primeramente la función iniciar población, realiza la carga los requerimientos de

Algoritmo 1 Algoritmo para la generación de horarios

Entrada: Parámetros del horario: Tamaño de la población, Número máximo de generaciones, Tipo de horario, Probabilidad de reproducción, Probabilidad de mutación.
Salida: Mejor solución encontrada (Horario)

```
1: Iniciar población
2: Evaluar población
3: mientras no sea criterio de paro hacer
4:     Elitismo
5:     Seleccionar individuos para cruza.
6:     Cruzamiento(Probabilidad de reproducción)
7:     Mutación(Probabilidad de mutación)
8:     Sustituir Población
9:     Evaluar Población
10: fin mientras
11: devolver Mejor individuo encontrado
```

un archivo de texto y genera las estructuras cromosómicas para la cantidad de individuos que se haya especificado en la llamada al algoritmo, además de forma aleatoria, se realizan las asignaciones iniciales en todos los individuos de acuerdo a requerimientos.

Posteriormente, la población generada se somete al proceso de evaluación de la aptitud para cada individuo con base en la función objetivo general y local, además se realiza la ponderación de las aptitudes correspondientes con fines de establecer proporciones para los individuos de manera que los más aptos tengan mayores posibilidades que los demás de trascender. También se determina si es que algún individuo alcanzó la función de optimización y en su caso provocar la condición de paro.

Mediante un ciclo repetitivo se controla la evolución para las siguientes funciones. En caso de continuar la evolución se llevará a cabo el proceso de conservar al mejor adaptado (elitismo), para ello se realiza una copia de su cromosoma y se reemplaza al peor individuo por éste, además de que el mejor individuo se mueve a la primera posición de la población para que los subsecuentes procedimientos no lo afecten.

El siguiente paso es generar de manera probabilística, una población temporal de aquellos individuos que tendrán la oportunidad de realizar intercambio de genes, este procedimiento se lleva a cabo mediante la selección por ruleta, se decide el uso de este método dado que es el que brinda oportunidad de reproducción a aquellos individuos que son más aptos consiguiendo con ello una mejor explotación del espacio de búsqueda. La experimentación considera el uso de porcentajes para el cruzamiento y la mutación de acuerdo con [10] y [17], de manera que los operadores sexuales solo se aplican en caso de que, el número aleatorio generado para tal fin, se encuentre dentro del rango del porcentaje, en caso contrario los padres se conservan sin modificación en la población temporal.

Tras la selección se lleva a cabo el intercambio sexual de genes produciendo nuevos individuos en la población temporal, también se lleva a cabo el proceso de mutación en la población temporal. Este conjunto de individuos de nueva generación reemplazarán a la población anterior.

Finalmente, la nueva población se debe someter al proceso de evaluación y la evolución se repetirá mientras no se cumpla la condición de paro.

Capítulo 4

Experimentación y resultados

En este capítulo se detalla la metodología de investigación seguida para la elaboración de esta investigación. Se plantean los pasos a seguir para conseguir que el algoritmo genético funcione para el conjunto de datos y recursos disponibles.

Se analiza de manera inicial el funcionamiento del AG con las diferentes restricciones implicadas en la construcción de los horarios de clases, los resultados dan pauta al mejoramiento de la técnica y poder afinar la puesta en marcha para responder a los planteamientos iniciales de este trabajo.

Para analizar las diferentes ejecuciones de los problemas del conjunto de datos y los comportamientos que arrojan los resultados de los mismos, se determina repetir 30 veces las ejecuciones del AG como un parámetro estadístico significativo. De esta forma se puede sustentar el funcionamiento del programa mediante la búsqueda de las mejores ejecuciones en promedio.

4.1. Recursos empleados

Para la codificación del algoritmo genético diseñado en pos de la resolución del problema de horarios, se utiliza el lenguaje de programación Java, debido a sus posibilidades multiplataforma, el JDK utilizado es la versión 8 para 64 bits. Se desarrolla en el entorno NetBeans versión 8.

La computadora utilizada para el desarrollo cuenta con: un procesador Intel i5, 4GB de RAM y disco duro de 360 GB. Sistema operativo Windows 8.1 pro.

Tras la codificación del algoritmo genético, se lleva a cabo la experimentación con los problemas propuestos, se observa la necesidad de ejecutar las corridas en un equipo dispuesto únicamente para tal fin y que sea convenientemente mas potente, en virtud de que en el equipo donde se lleva a cabo el desarrollo las prácticas tardan cada vez más tiempo en terminar. En consecuencia, los experimentos se ejecutan en el clúster del Centro Universitario Valle de México, éste cuenta con 4 servidores Dell, cada uno equipado con 2 procesadores Xeon de 6 nucleos HT y 32 GB de memoria RAM. Sistema operativo Centos 6.6 y Java 1.7.0_51.

Como se menciona en la sección 1.2 los datos empleados para los experimentos son los que brinda el conjunto de datos de la mencionada Universidad de Brunel.

4.2. Experimentos preliminares

Para los fines de comparación mencionados durante este trabajo, de manera inicial se presenta la Figura 4.1 que muestra comportamiento del error (choques) promedio en el problema Tipo 4, que es el más sencillo del conjunto de datos, a lo largo de doscientas mil repeticiones. Es posible ver que, si bien la cantidad choques varia de manera considerable,

también se observa que dicha cantidad no tiende a bajar en la linea del tiempo. Es importante mencionar también que dicha ejecución ha tomado 52 horas de ejecución sin lograr minimizar el error promedio.

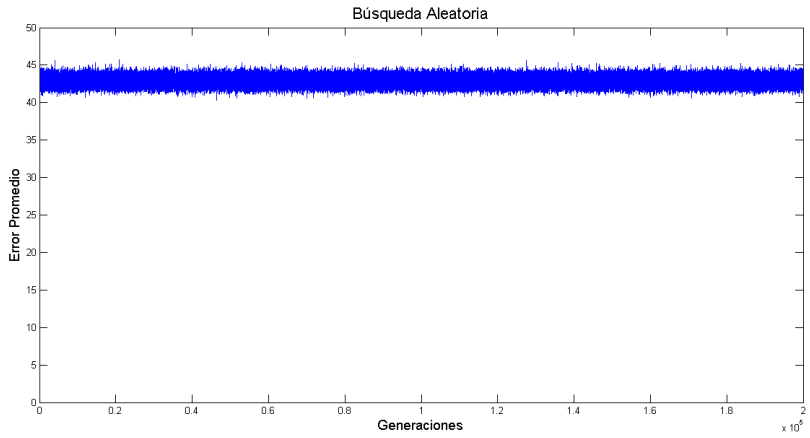


Figura 4.1: Error promedio en una búsqueda aleatoria.

Durante la etapa de la programación del AG se realizan diversas pruebas con el objetivo de contar con un programa cuya ejecución considere 3 funcionalidades fundamentales: primero que pueda leer la información de las necesidades almacenadas en archivos de texto, segundo que pueda ejecutar el algoritmo genético en búsqueda del horario que cumpla con todas las restricciones y tercero que pueda brindar las soluciónes posibles (horarios), en un archivo de salida como entregable para su revisión.

Una vez se cuenta con las funcionalidades básicas, se enfrenta al programa a los cinco tipos de problemas y se observan sus resultados antes de la ejecución definitiva.

Para los cinco tipos de problemas, los parámetros iniciales son:

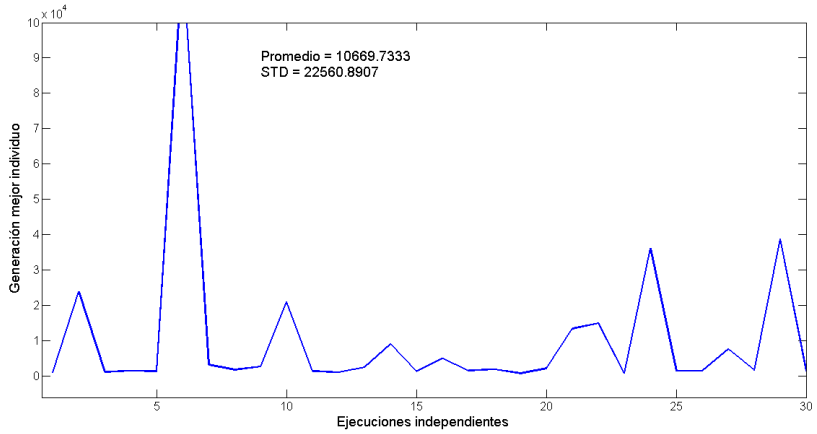
- Población de 100 individuos.
- Un total de cien mil generaciones.
- Para el porcentaje de cruce se ejecutan experimentos con los siguientes valores: 10 %,

20 %, 30 %, 40 %, 50 %, 60 %, 70 %, 80 %, 90 % y 100 %.

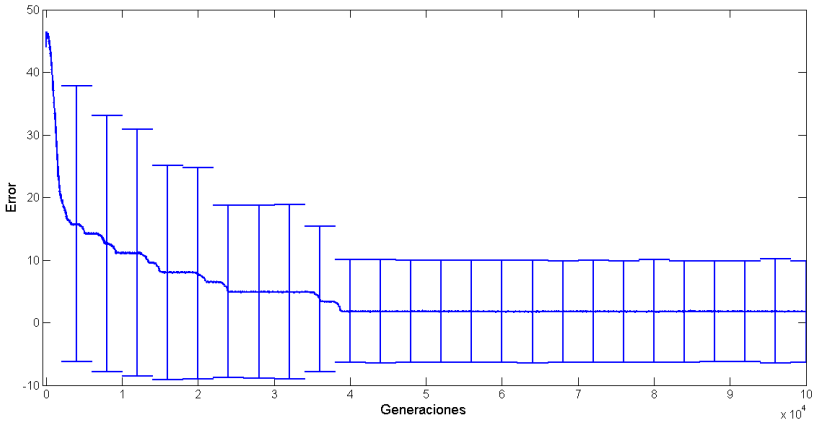
- En el caso del porcentaje de mutación los valores igualemente son: 10 %, 20 %, 30 %, 40 %, 50 %, 60 %, 70 %, 80 %, 90 % y 100 %.
- Se aplica elitismo en el AG.
- El criterio de paro es que algún individuo llegue a un horario factible.

En tal sentido se ejecutan experimentos para evidenciar el hecho de que el AG puede obtener horarios factibles, se observa que las ejecuciones con bajos porcentajes de cruce no tienen la suficiente capacidad de explotación en el espacio de búsqueda, sin embargo, las ejecuciones con altos porcentajes puede brindar horarios sin choques a costa de que consumen más tiempo computacional, también se observa que el porcentaje de mutación resulta muy importante para la exploración en el espacio de búsqueda, en este tema, una alta tasa en la mutación no beneficia a la convergencia del AG y una baja tasa tampoco lo hace debido a que la población difícilmente sale de mínimos locales. Por tanto, se determina para esta fase de observación utilizar porcentajes de 60 % de cruce y de 40 % en mutación.

Se obtienen resultados preliminares como sigue: la Figura 4.2 muestra los resultados de la ejecución para el problema Tipo 4, en el inciso *a* se observa el número de generaciones en el que el AG encuentra un horario viable así como su desviación estándar, el inciso *b* muestra el progreso de la disminución de las generaciones a través de la ejecución.



(a) Generaciones para el mejor individuo

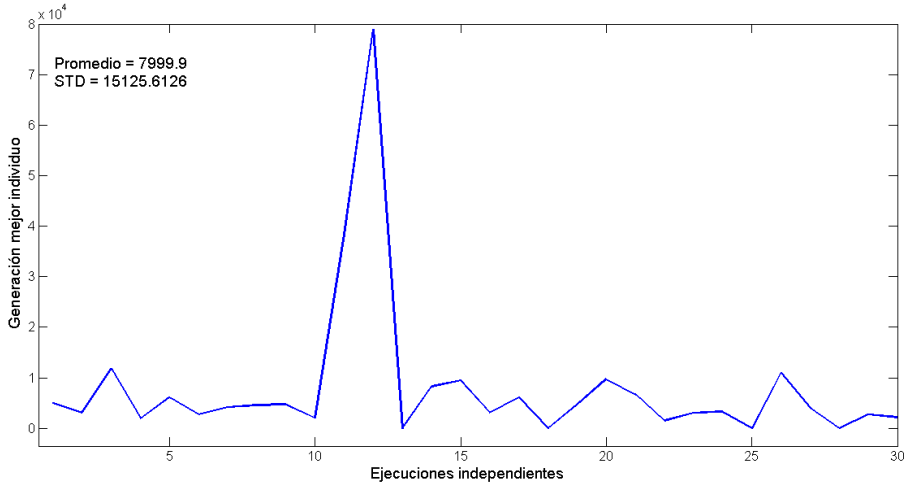


(b) Disminución de error

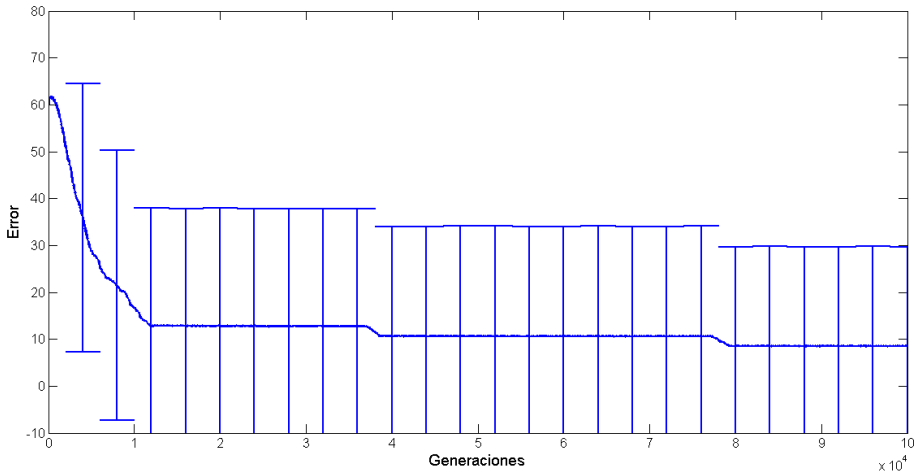
Figura 4.2: Gráficas iniciales para el Tipo 4

Se da cuenta de que el tamaño de la población y la cantidad de generaciones son vastos para el problema Tipo 4 dado que todas las ejecuciones logran obtener un horario factible.

Para el caso del Tipo 5, se obtiene de acuerdo a la Figura 4.3 que la población y la cantidad de generaciones son suficientes aunque hay algunos casos en los que las ejecuciones no logran un horario viable (valores que tocan el cero), en consecuencia, en el promedio de los errores se obtienen valores un poco más elevados con respecto al Tipo 4.



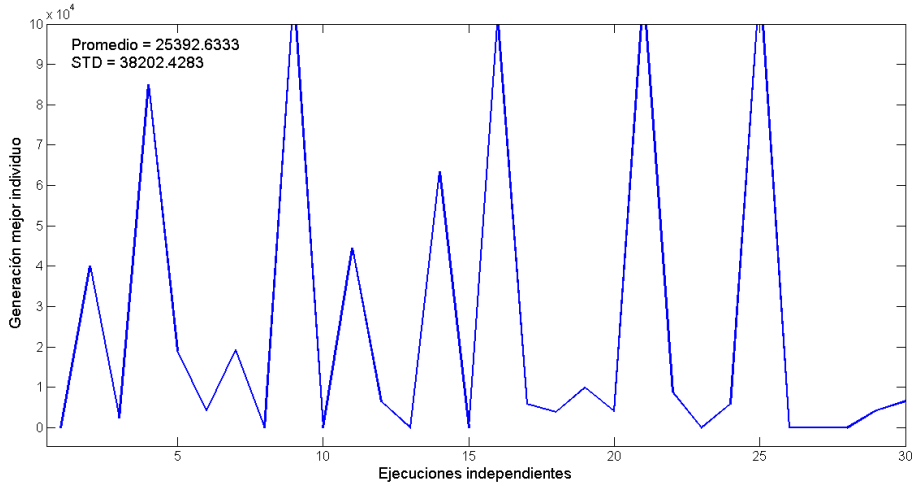
(a) Generaciones para el mejor individuo



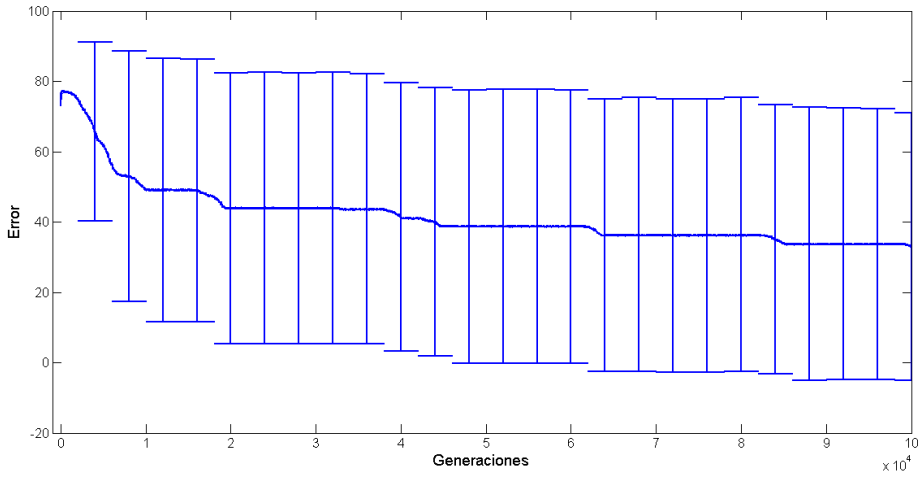
(b) Disminución de error

Figura 4.3: Gráficas iniciales para el Tipo 5

En el Tipo 6 se confirma la tendencia de que, a mayor complejidad, los parámetros iniciales propuestos el AG da resultados de menor calidad, ya que de acuerdo a la Figura 4.4 se ve que hay más ejecuciones que no llegan al óptimo y que los valores promedio son más elevados.



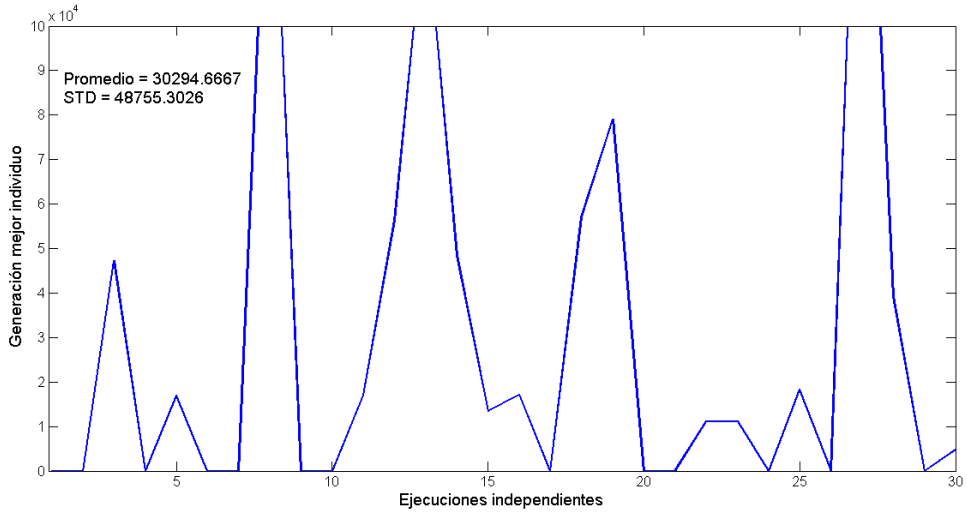
(a) Generaciones para el mejor individuo



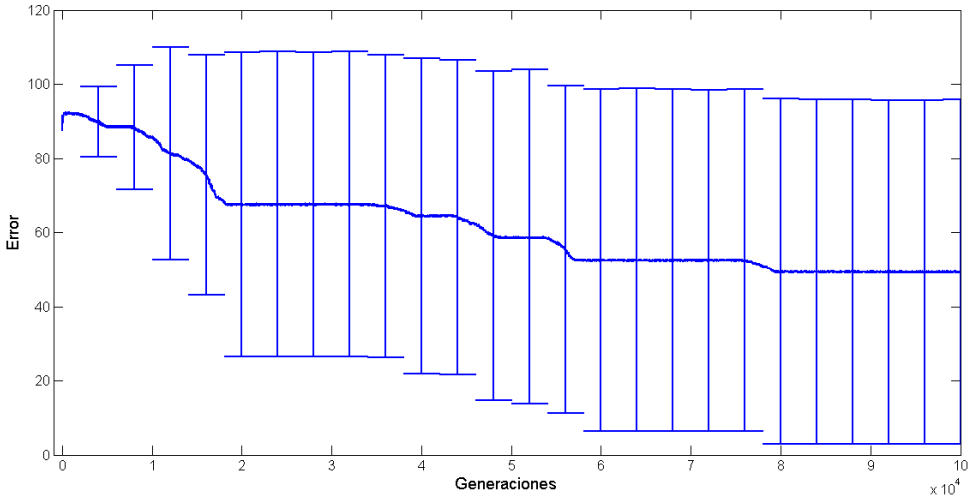
(b) Disminución de error

Figura 4.4: Gráficas iniciales para el Tipo 6

La Figura 4.5 demuestra que para el Tipo 7 la población y cantidad de generaciones ya no son suficientes dado que hay más casos en los que no se encuentra al óptimo además de que el error se mantiene en promedios muy altos durante las 30 ejecuciones.



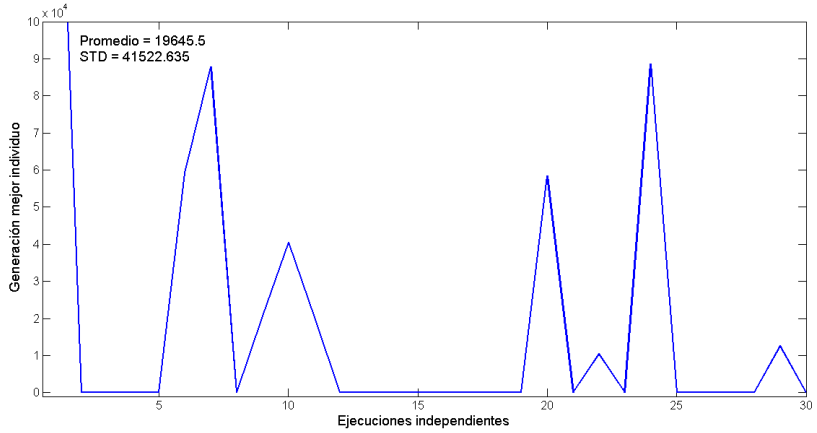
(a) Generaciones para el mejor individuo



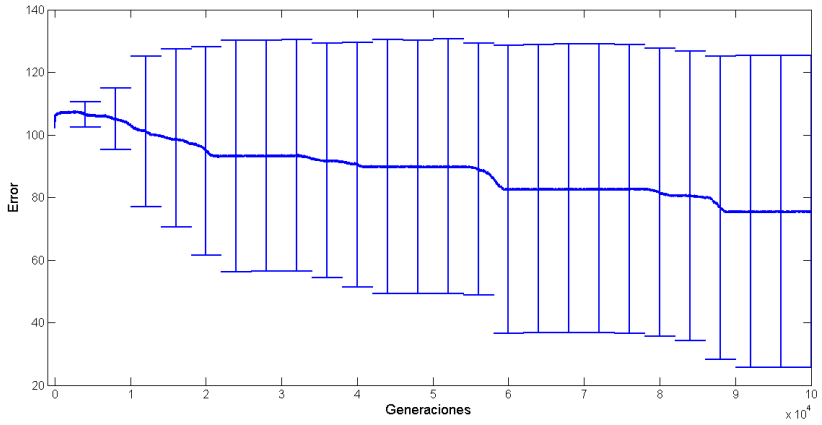
(b) Disminución de error

Figura 4.5: Gráficas iniciales para el Tipo 7

Finalmente para el caso de mayor complejidad de este trabajo, el Tipo 8, los resultados mostrados en la Figura 4.6 son de muy baja calidad en estas ejecuciones iniciales.



(a) Generaciones para el mejor individuo



(b) Disminución de error

Figura 4.6: Gráficas iniciales para el Tipo 8

A partir de los resultados anteriores se determinan algunos ajustes para la ejecución final del AG con la configuración más adecuada de acuerdo a lo observado:

- Tamaño de la población: se establece en 300 individuos, se considera que es un tamaño vasto para el problema Tipo 4 y suficiente para el Tipo 8.
- Número de generaciones: se determina en doscientas mil, dado el incremento en el tamaño de la población, se considera que es suficiente para todos los casos.
- Probabilidad de cruce: se deja en 60 % dada la observación, aunque en las ejecuciones definitivas se realizarán comparaciones adicionales con porcentajes distintos.

- Probabilidad de mutación: se fija en 40 %.
- Elitismo: se preserva la técnica del elitismo, conservando al mejor individuo en cada generación ya que demuestra brindar mejores resultados.
- Criterio de paro: para las corridas finales el criterio de paro es la finalización de todas las generaciones debido a que se realizan comparaciones adicionales donde es necesario contar con la ejecución completa.

Los resultados mostrados a continuación son los definitivos después de los ajustes con experimentos previos. De cada Tipo de problema, para la reproducción al 60 % se realiza un barrido por los porcentajes del 10 % al 100 % y éstos se repiten 30 veces, mismo caso para la mutación fijada en 40 %, se realiza un barrido para porcentajes del 10 % al 100 %. Por lo tanto, la experimentación definitiva consta de 3000 experimentos, los cuales toman un tiempo aproximado de 4 meses de ejecución en el clúster del Centro Universitario UAEM Valle de México.

4.3. Resultados

A continuación se examinan los resultados despúes del tiempo empleado para la búsqueda de soluciones a los problemas propuestos, en términos generales se muestran buenos resultados, el AG logra converger a una solución donde no hay choques en la mayoría de los casos, i.e. algunas ejecuciones independientes pueden no llegar, pero otras si obtienen un resultado adecuado.

Para todas aquellas gráficas que muestran comparativas en porcentaje de cruce y de mutación, se han eliminado algunas líneas que resultan ruidosas para el entendimiento de los comportamientos.

En primer lugar se analiza el tiempo de las ejecuciones completas, éstas varían en función de la complejidad del problema, de tal forma que los tiempos para el problema Tipo 4 se consideran cortos y los tiempos para el problema Tipo 8 se consideran largos, estos se muestran en la Figura 4.7. Se aclara que la Figura corresponde a la configuración que se determinó en los resultados previos, específicamente con porcentajes de cruce y mutación del 60 % y 40 % respectivamente.

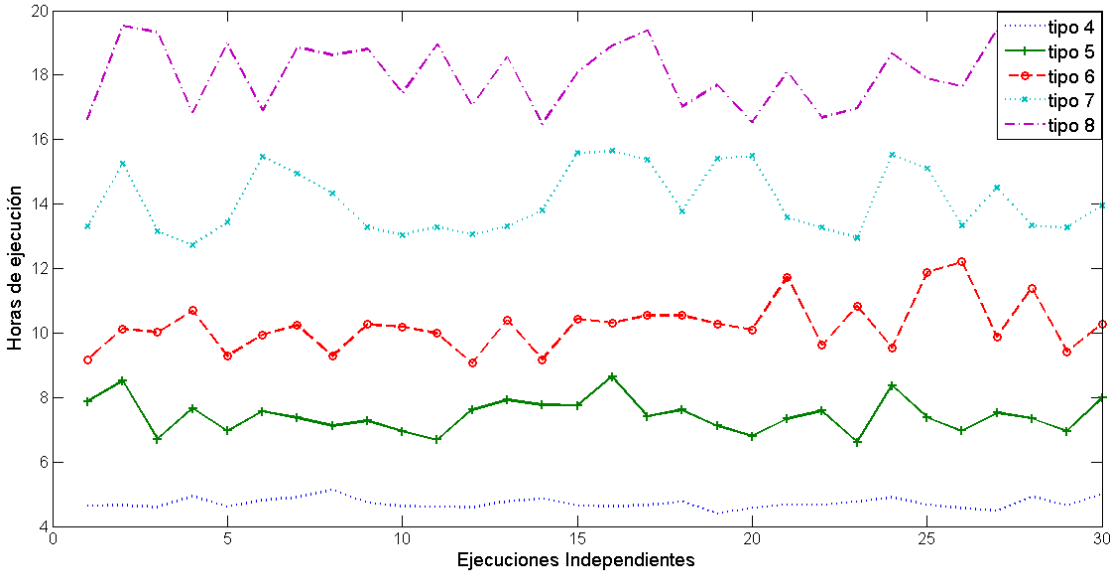


Figura 4.7: Comparativa del tiempo de todos los tipos en todas las ejecuciones.

Para el mismo caso se presenta también la tabla 4.1 donde se realizan comparativas entre el cálculo del promedio de tiempo (en horas), el cálculo de la desviación estándar, el mínimo y el máximo tiempo para todos los tipos de problemas.

Tabla 4.1: Comparativa en horas de los tiempos de ejecución para todas ejecuciones

	Media	STD	Min	Max
Tipo 4	4.766887	0.244307	4.206985	5.280438
Tipo 5	7.449537	0.436541	6.399144	8.653907
Tipo 6	10.28743411	0.667197	9.220105	11.862730
Tipo 7	13.80722993	0.944536	12.586236	15.830783
Tipo 8	17.23894024	0.833102	16.114529	19.068261

Como es de esperar el elitismo resulta de mucha utilidad para asegurar la convergencia más rápida de la población debido a que se conservan a los mejores individuos de la población durante la evolución del algoritmo, apoyando fuertemente a que la búsqueda pueda encontrar nuevos mínimos locales que reemplacen al que se encuentra en turno, si se maneja el criterio de paro por generaciones, se asegura que al menos se tenga el mejor mínimo local encontrado.

También se puede observar que a mayor número de individuos en la población (mayor diversidad al inicio de la evolución) se obtienen convergencias más rápidas en virtud de que se dota al AG de una capacidad mayor de exploración, en contraparte el uso de una mayor población también aumenta el uso de memoria (aunque con las capacidades actuales de hardware no representa un gran problema) y exige una mayor capacidad de procesamiento y de tiempo computacional, en este caso los tiempos si pueden variar considerablemente con respecto al tamaño de la población.

Con respecto a la determinación del porcentaje de cruzamiento y mutación, los resultados arrojan que a mayor porcentaje de cruzamiento se beneficia la obtención de convergencias más rápidas, sin embargo, esto se logra a costa de una mayor exigencia en el procesamiento y por lo tanto se ve mermado el tiempo de solución del problema. En este caso, se busca aquel porcentaje que brinda un equilibrio entre la rapidez para llegar a una solución sin penalizaciones, el uso de memoria y el procesamiento exigido por el programa. Como ejemplo, se presenta la Figura 4.8 del problema Tipo 5, la comparación es entre una ejecución con el 10 % de reproducción, una del 60 % y otra con el 100 %. De acuerdo a la imagen, se reafirma el porcentaje de cruzamiento en 60 % ya que se puede observar que los tiempos del 10 % distan mucho del 100 % y que los tiempos del 60 % se equiparan a los del 100 %. Aunque pareciera que el porcentaje del 100 % brinda mejores resultados, en [13] se afirma que no es recomendable, ya que en otros problemas los resultados experimentales afirman que no

beneficia al funcionamiento del AG, esto podría dar lugar a un sobrecruzamiento lo cual exige mayor cómputo, además de mayores tiempos de ejecución. Las gráficas en extenso se pueden observar en el apéndice B.

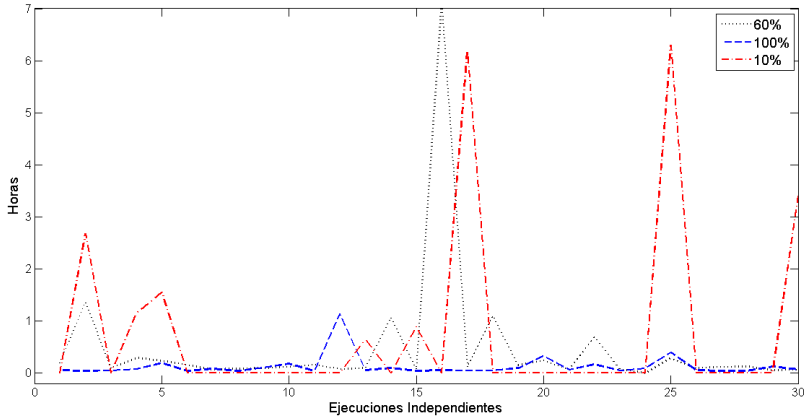


Figura 4.8: Comparativa de los tiempos de convergencia con 3 porcentajes de cruzamiento en el problema Tipo 5

En el tema de la mutación también se demuestra que el 40 % es un porcentaje adecuado al presentar un comportamiento donde los tiempos para encontrar una solución viable se equiparan con los porcentajes 60 % y 100 % en contraste con el 10 %. La Figura 4.9 muestra la comparativa para los diferentes porcentajes de mutación.

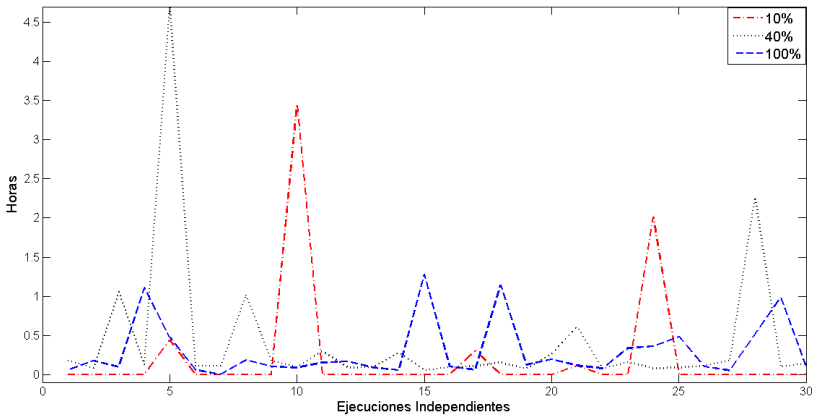


Figura 4.9: Comparativa de los tiempos de convergencia para las diferentes mutaciones en el problema Tipo 5

Con base en la Figura 4.10, se puede ver las diferencias en tiempo que el AG toma para llegar a un horario factible, en este caso se muestra una comparativa para los cinco tipos de problemas, se observa que, para los problemas de menor complejidad del conjunto, el AG encuentra algún horario factible de manera muy rápida y para los mas complicados la búsqueda toma más tiempo, se recuerda que las lineas que tocan el cero son aquellas ejecuciones que no logran generar ningun horario viable. Las gráficas completas se pueden ver en el apéndice B.

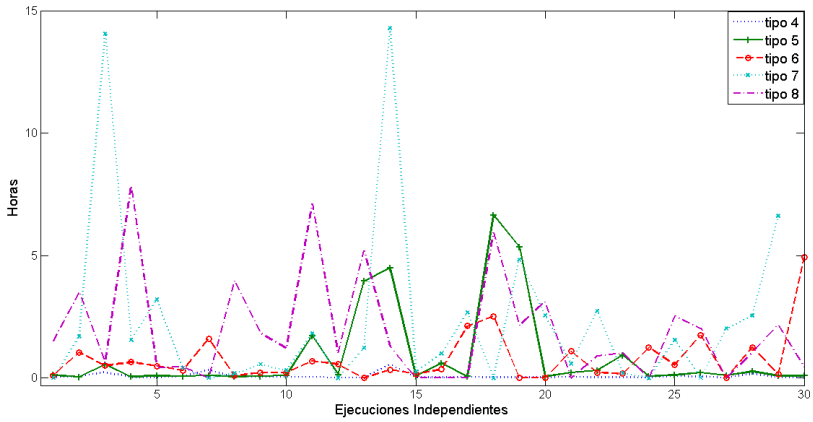


Figura 4.10: Cantidad de generaciones necesarias para llegar a un horario factible con porcentajes del 60 % en cruzamiento y 40 % mutación

También se presenta la tabla 4.2 donde se comparan los tiempos para que algún individuo logre un horario factible.

Tabla 4.2: Comparativa de los tiempos (en horas) para para lograr un horario factible

	Media	STD	Min	Max
Tipo 4	0.077797	0.114466	0.015684	0.567472
Tipo 5	0.892769	1.756324	0.047732	6.644672
Tipo 6	0.898583	1.052522	0.094218	4.921423
Tipo 7	2.906403	3.878130	0.183523	14.276760
Tipo 8	2.496143	2.162690	0.436449	7.859636

Para el caso de las generaciones transcurridas para lograr un horario factible, los resultados para los cinco tipos de problemas se comparan en la Figura 4.11, se puede apreciar que el

problema Tipo 4 se resuelve en muy pocas ejecuciones en la mayoría de los experimentos y que a medida que se presentan los problemas de mayor complejidad, se puede percibir que las líneas toman mayor cantidad de generaciones para llegar a una solución sin penalizaciones, donde cabe acotar que aquellas líneas que se tienen un valor de cero, indican que ningún individuo de la población (a lo largo de todas las generaciones) llego a una solución deseada, i.e. no se llego al mínimo de la función objetivo.

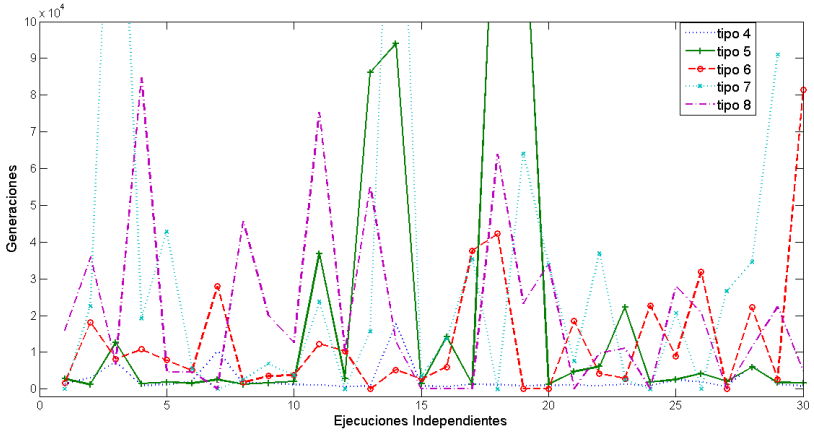


Figura 4.11: Cantidad de generaciones para llegar a un horario factible.

Se muestra también la tabla 4.3 donde se compara el número de generaciones transcurridas para que algún individuo logre un horario factible.

Tabla 4.3: Comparativa del número de generaciones para para lograr un horario factible

	Media	STD	Min	Max
Tipo 4	2511.966667	3623.865982	680.0	17781.0
Tipo 5	20457.700000	41219.191602	1224.0	155595.0
Tipo 6	15415.076923	17752.307199	1687.0	81437.0
Tipo 7	38707.478261	51948.678287	2546.0	190201.0
Tipo 8	26677.304348	23297.646889	4639.0	84775.0

Asimismo, se puede examinar en la Figura 4.12 el comportamiento de la aptitud (fitness) a lo largo de las generaciones, cabe mencionar que es el promedio de 30 ejecuciones y, éste se obtiene con el promedio de sus individuos. Se hace hincapié en que este compartamiento nos muestra que la población entera va mejorando su aptitud, es decir, tiene una tendencia a la

minimización a lo largo de las generaciones y que no hay incremento de los choques durante toda la ejecución. También se ve que, los problemas de menor complejidad se minimizan con mayor facilidad en promedio y que conforme aumenta la complejidad del problema, las líneas quedan en valores de error más altos en promedio, dado que hay algunas ejecuciones en los experimentos que no logran minimizar a cero.

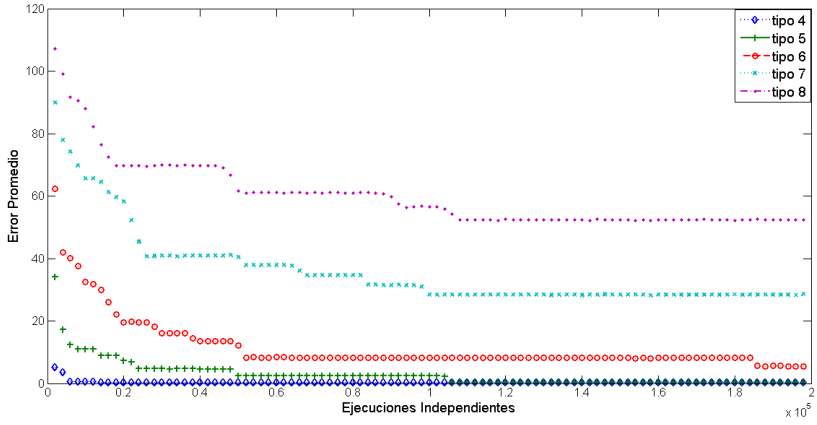


Figura 4.12: Error promedio a tráves de las generaciones.

De igual manera se expone la tabla 4.4 donde se compara el comportamiento del error promedio a través de todas las generaciones.

Tabla 4.4: Comparativa del error promedio a través de las generaciones

	Media	STD	Min	Max
Tipo 4	0.395000	0.229238	0.080000	0.930000
Tipo 5	0.286333	0.164201	0.060000	0.690000
Tipo 6	10.640667	26.551554	0.120000	78.550000
Tipo 7	17.063793	35.496021	0.260000	94.610000
Tipo 8	25.724000	46.359801	0.280000	108.880000

Finalmente, se presentan algunos ejemplos de horarios resultantes para los cinco tipos de problemas. En la tabla 4.5 se presenta la solución para el Tipo 4, en la tabla 4.6 para el Tipo 5, la tabla 4.7 para el Tipo 6, en la tabla 4.8 para el Tipo 7 y en la tabla 4.9 para el Tipo 8.

Tabla 4.5: Un ejemplo de horario resultante para el problema Tipo 4.

Salon 1 Dia 1 Per 1: P3-U4	Per 2: P2-U1	Per 3: P3-U2	Per 4: P1-U3	Per 5: P4-U4	Per 6: P4-U2
Dia 2 Per 1: P4-U3	Per 2: P3-U4	Per 3: P1-U2	Per 4: P2-U4	Per 5: P3-U4	Per 6: P2-U4
Dia 3 Per 1: P2-U1	Per 2: P4-U3	Per 3: P1-U1	Per 4: P2-U2	Per 5: P4-U3	Per 6: P1-U1
Dia 4 Per 1: P1-U4	Per 2: P4-U4	Per 3: P4-U3	Per 4: P4-U3	Per 5: P4-U1	Per 6: P1-U4
Dia 5 Per 1: P3-U3	Per 2: P4-U2	Per 3: P4-U1	Per 4: P3-U1	Per 5: P4-U3	Per 6: P2-U3
Salon 2 Dia 1 Per 1: P2-U2	Per 2: P3-U4	Per 3: P1-U3	Per 4: P3-U1	Per 5: P2-U1	Per 6: P1-U4
Dia 2 Per 1: P3-U3	Per 2: P2-U3	Per 3: P2-U1	Per 4: P3-U2	Per 5: P2-U2	Per 6: P4-U1
Dia 3 Per 1: P3-U2	Per 2: P2-U1	Per 3: P4-U4	Per 4: P4-U4	Per 5: P3-U2	Per 6: P2-U1
Dia 4 Per 1: P2-U3	Per 2: P1-U1	Per 3: P2-U1	Per 4: P2-U4	Per 5: P2-U2	Per 6: P4-U2
Dia 5 Per 1: P4-U2	Per 2: P2-U4	Per 3: P1-U1	Per 4: P1-U4	Per 5: P2-U2	Per 6: P4-U1
Salon 3 Dia 1 Per 1: P1-U4	Per 2: P1-U1	Per 3: P4-U3	Per 4: P4-U1	Per 5: P3-U2	Per 6: P3-U3
Dia 2 Per 1: P1-U4	Per 2: P4-U4	Per 3: P3-U2	Per 4: P4-U2	Per 5: P1-U4	Per 6: P3-U4
Dia 3 Per 1: P1-U4	Per 2: P1-U1	Per 3: P2-U4	Per 4: P3-U1	Per 5: P1-U4	Per 6: P3-U3
Dia 4 Per 1: P4-U1	Per 2: P3-U2	Per 3: P1-U3	Per 4: P3-U2	Per 5: P1-U3	Per 6: P3-U2
Dia 5 Per 1: P2-U3	Per 2: P3-U3	Per 3: P3-U4	Per 4: P2-U1	Per 5: P3-U3	Per 6: P1-U4
Salon 4 Dia 1 Per 1: P4-U2	Per 2: P4-U3	Per 3: P2-U3	Per 4: P2-U3	Per 5: P1-U1	Per 6: P2-U2
Dia 2 Per 1: P2-U2	Per 2: P1-U3	Per 3: P4-U4	Per 4: P1-U4	Per 5: P4-U2	Per 6: P1-U1
Dia 3 Per 1: P4-U1	Per 2: P3-U1	Per 3: P3-U3	Per 4: P1-U1	Per 5: P2-U2	Per 6: P4-U2
Dia 4 Per 1: P3-U2	Per 2: P2-U1	Per 3: P3-U1	Per 4: P1-U3	Per 5: P3-U3	Per 6: P2-U3
Dia 5 Per 1: P1-U2	Per 2: P1-U3	Per 3: P2-U2	Per 4: P4-U2	Per 5: P1-U4	Per 6: P3-U4

Tabla 4.6: Un ejemplo de horario resultante para el problema Tipo 5.

Salon 1 Dia 1 Per 1: P4-U5	Per 2: P4-U2	Per 3: P2-U4	Per 4: P2-U5	Per 5: P5-U1	Per 6: P1-U2
Continua...					

Dia 2					
Per 1: P4-U5	Per 2: P4-U3	Per 3: P5-U3	Per 4: P3-U5	Per 5: P5-U1	Per 6: P3-U4
Dia 3					
Per 1: P4-U2	Per 2: P4-U1	Per 3: P4-U4	Per 4: P2-U5	Per 5: P1-U1	Per 6: P3-U4
Dia 4					
Per 1: P4-U5	Per 2: P4-U3	Per 3: P2-U4	Per 4: P5-U2	Per 5: P1-U4	Per 6: P3-U4
Dia 5					
Per 1: P4-U5	Per 2: P4-U2	Per 3: P5-U5	Per 4: P3-U3	Per 5: P3-U2	Per 6: P1-U2
Salon 2					
Dia 1					
Per 1: P1-U1	Per 2: P1-U5	Per 3: P1-U5	Per 4: P3-U1	Per 5: P3-U1	Per 6: P3-U1
Dia 2					
Per 1: P1-U1	Per 2: P2-U3	Per 3: P4-U5	Per 4: P5-U2	Per 5: P3-U4	Per 6: P5-U1
Dia 3					
Per 1: P1-U1	Per 2: P5-U2	Per 3: P5-U3	Per 4: P3-U3	Per 5: P3-U4	Per 6: P5-U1
Dia 4					
Per 1: P1-U1	Per 2: P2-U5	Per 3: P4-U5	Per 4: P3-U3	Per 5: P3-U4	Per 6: P5-U2
Dia 5					
Per 1: P1-U4	Per 2: P2-U2	Per 3: P4-U1	Per 4: P4-U2	Per 5: P4-U3	Per 6: P5-U1
Salon 3					
Dia 1					
Per 1: P5-U1	Per 2: P5-U2	Per 3: P5-U3	Per 4: P4-U3	Per 5: P1-U4	Per 6: P2-U2
Dia 2					
Per 1: P5-U4	Per 2: P1-U3	Per 3: P3-U3	Per 4: P2-U1	Per 5: P1-U4	Per 6: P1-U2
Dia 3					
Per 1: P5-U1	Per 2: P3-U2	Per 3: P3-U3	Per 4: P4-U5	Per 5: P4-U2	Per 6: P1-U2
Dia 4					
Per 1: P5-U4	Per 2: P1-U5	Per 3: P5-U5	Per 4: P2-U1	Per 5: P4-U2	Per 6: P1-U2
Dia 5					
Per 1: P5-U4	Per 2: P1-U1	Per 3: P3-U3	Per 4: P1-U1	Per 5: P1-U4	Per 6: P2-U2
Salon 4					
Dia 1					
Per 1: P3-U2	Per 2: P3-U3	Per 3: P4-U2	Per 4: P5-U3	Per 5: P2-U1	Per 6: P5-U3
Dia 2					
Per 1: P3-U2	Per 2: P5-U1	Per 3: P1-U1	Per 4: P4-U4	Per 5: P2-U1	Per 6: P4-U5
Dia 3					
Per 1: P3-U3	Per 2: P1-U3	Per 3: P1-U3	Per 4: P5-U3	Per 5: P2-U2	Per 6: P2-U4
Dia 4					
Per 1: P2-U5	Per 2: P5-U4	Per 3: P1-U4	Per 4: P4-U4	Per 5: P2-U2	Per 6: P4-U5
Dia 5					
Per 1: P2-U4	Per 2: P5-U5	Per 3: P1-U5	Per 4: P5-U3	Per 5: P2-U4	Per 6: P3-U2
Salon 5					
Dia 1					
Per 1: P2-U2	Per 2: P2-U3	Per 3: P3-U1	Per 4: P1-U1	Per 5: P4-U5	Per 6: P4-U1
Dia 2					
Per 1: P2-U2	Per 2: P3-U3	Per 3: P2-U5	Per 4: P1-U3	Per 5: P4-U5	Per 6: P2-U4
Dia 3					
Per 1: P2-U2	Per 2: P2-U3	Per 3: P2-U5	Per 4: P1-U4	Per 5: P5-U1	Per 6: P4-U3
Dia 4					
Per 1: P3-U5	Per 2: P3-U3	Per 3: P3-U4	Per 4: P1-U5	Per 5: P5-U4	Per 6: P2-U4
Dia 5					
Per 1: P3-U5	Per 2: P3-U3	Per 3: P2-U5	Per 4: P2-U1	Per 5: P5-U5	Per 6: P4-U4

Tabla 4.7: Un ejemplo de horario resultante para el problema Tipo 6.

Salon 1 Dia 1 Per 1: P6-U1	Per 2: P4-U4	Per 3: P2-U2	Per 4: P3-U1	Per 5: P4-U1	Per 6: P6-U1
Dia 2 Per 1: P6-U4	Per 2: P2-U3	Per 3: P2-U5	Per 4: P4-U5	Per 5: P1-U5	Per 6: P5-U6
Dia 3 Per 1: P4-U6	Per 2: P6-U6	Per 3: P6-U6	Per 4: P6-U1	Per 5: P3-U6	Per 6: P2-U2
Dia 4 Per 1: P6-U5	Per 2: P4-U3	Per 3: P3-U4	Per 4: P4-U5	Per 5: P1-U1	Per 6: P3-U5
Dia 5 Per 1: P1-U4	Per 2: P5-U5	Per 3: P1-U5	Per 4: P5-U5	Per 5: P2-U6	Per 6: P3-U4
Salon 2 Dia 1 Per 1: P4-U6	Per 2: P5-U5	Per 3: P6-U2	Per 4: P1-U6	Per 5: P6-U2	Per 6: P5-U2
Dia 2 Per 1: P1-U1	Per 2: P5-U5	Per 3: P4-U5	Per 4: P3-U5	Per 5: P5-U1	Per 6: P2-U1
Dia 3 Per 1: P5-U3	Per 2: P1-U6	Per 3: P2-U5	Per 4: P5-U6	Per 5: P4-U5	Per 6: P6-U2
Dia 4 Per 1: P1-U6	Per 2: P5-U2	Per 3: P1-U1	Per 4: P3-U6	Per 5: P2-U5	Per 6: P1-U4
Dia 5 Per 1: P5-U1	Per 2: P6-U5	Per 3: P3-U2	Per 4: P3-U1	Per 5: P5-U3	Per 6: P6-U3
Salon 3 Dia 1 Per 1: P2-U4	Per 2: P3-U2	Per 3: P3-U6	Per 4: P5-U3	Per 5: P2-U1	Per 6: P1-U1
Dia 2 Per 1: P2-U6	Per 2: P1-U5	Per 3: P5-U6	Per 4: P6-U4	Per 5: P3-U3	Per 6: P3-U1
Dia 3 Per 1: P3-U4	Per 2: P3-U5	Per 3: P3-U6	Per 4: P2-U5	Per 5: P1-U6	Per 6: P1-U1
Dia 4 Per 1: P3-U2	Per 2: P1-U4	Per 3: P4-U3	Per 4: P6-U4	Per 5: P6-U3	Per 6: P5-U4
Dia 5 Per 1: P3-U2	Per 2: P1-U2	Per 3: P5-U3	Per 4: P4-U1	Per 5: P4-U4	Per 6: P5-U2
Salon 4 Dia 1 Per 1: P3-U5	Per 2: P6-U2	Per 3: P5-U1	Per 4: P6-U1	Per 5: P5-U5	Per 6: P3-U4
Dia 2 Per 1: P3-U2	Per 2: P3-U4	Per 3: P6-U2	Per 4: P1-U4	Per 5: P2-U3	Per 6: P4-U2
Dia 3 Per 1: P6-U5	Per 2: P4-U6	Per 3: P5-U3	Per 4: P1-U5	Per 5: P2-U3	Per 6: P5-U1
Dia 4 Per 1: P5-U3	Per 2: P2-U1	Per 3: P6-U6	Per 4: P5-U3	Per 5: P5-U6	Per 6: P4-U4
Dia 5 Per 1: P2-U4	Per 2: P4-U2	Per 3: P2-U1	Per 4: P1-U6	Per 5: P6-U5	Per 6: P2-U2
Salon 5 Dia 1 Per 1: P1-U5	Per 2: P1-U4	Per 3: P4-U2	Per 4: P2-U6	Per 5: P3-U4	Per 6: P4-U3
Dia 2 Per 1: P4-U3	Per 2: P6-U1	Per 3: P1-U6	Per 4: P2-U1	Per 5: P4-U2	Per 6: P6-U1
Continua...					

Dia 3					
Per 1: P1-U3	Per 2: P5-U2	Per 3: P1-U5	Per 4: P3-U3	Per 5: P6-U3	Per 6: P4-U3
Dia 4					
Per 1: P4-U3	Per 2: P6-U1	Per 3: P5-U6	Per 4: P1-U3	Per 5: P4-U2	Per 6: P2-U2
Dia 5					
Per 1: P6-U3	Per 2: P2-U5	Per 3: P6-U4	Per 4: P6-U6	Per 5: P1-U4	Per 6: P1-U3
Salon 6					
Dia 1					
Per 1: P5-U5	Per 2: P2-U4	Per 3: P1-U6	Per 4: P4-U3	Per 5: P1-U5	Per 6: P2-U3
Dia 2					
Per 1: P5-U3	Per 2: P4-U4	Per 3: P3-U2	Per 4: P5-U1	Per 5: P6-U6	Per 6: P1-U4
Dia 3					
Per 1: P2-U6	Per 2: P2-U4	Per 3: P4-U4	Per 4: P4-U4	Per 5: P5-U1	Per 6: P3-U4
Dia 4					
Per 1: P2-U2	Per 2: P3-U1	Per 3: P2-U2	Per 4: P2-U2	Per 5: P3-U2	Per 6: P6-U3
Dia 5					
Per 1: P4-U2	Per 2: P3-U6	Per 3: P4-U3	Per 4: P2-U1	Per 5: P3-U4	Per 6: P4-U6

Tabla 4.8: Un ejemplo de horario resultante para el problema Tipo 7.

Salon 1					
Dia 1					
Per 1: P7-U1	Per 2: P4-U3	Per 3: P2-U2	Per 4: P6-U1	Per 5: P5-U3	Per 6: P3-U5
Dia 2					
Per 1: P7-U2	Per 2: P4-U4	Per 3: P1-U1	Per 4: P6-U2	Per 5: P5-U4	Per 6: P3-U2
Dia 3					
Per 1: P7-U5	Per 2: P7-U7	Per 3: P1-U5	Per 4: P6-U4	Per 5: P5-U1	Per 6: P3-U5
Dia 4					
Per 1: P7-U6	Per 2: P4-U6	Per 3: P2-U3	Per 4: P6-U5	Per 5: P5-U7	Per 6: P3-U7
Dia 5					
Per 1: P7-U1	Per 2: P4-U7	Per 3: P2-U6	Per 4: P6-U7	Per 5: P5-U1	Per 6: P3-U5
Salon 2					
Dia 1					
Per 1: P1-U4	Per 2: P1-U1	Per 3: P5-U1	Per 4: P7-U5	Per 5: P6-U1	Per 6: P4-U1
Dia 2					
Per 1: P1-U6	Per 2: P1-U5	Per 3: P5-U3	Per 4: P3-U6	Per 5: P6-U3	Per 6: P4-U2
Dia 3					
Per 1: P1-U6	Per 2: P4-U7	Per 3: P5-U4	Per 4: P5-U6	Per 5: P6-U7	Per 6: P4-U6
Dia 4					
Per 1: P1-U6	Per 2: P1-U7	Per 3: P5-U3	Per 4: P2-U2	Per 5: P3-U2	Per 6: P4-U6
Dia 5					
Per 1: P1-U6	Per 2: P1-U1	Per 3: P5-U1	Per 4: P2-U6	Per 5: P3-U2	Per 6: P4-U6
Salon 3					
Dia 1					
Per 1: P3-U1	Per 2: P6-U2	Per 3: P3-U5	Per 4: P5-U2	Per 5: P7-U5	Per 6: P1-U6
Dia 2					
Per 1: P3-U4	Per 2: P3-U3	Per 3: P2-U7	Per 4: P5-U2	Per 5: P7-U6	Per 6: P1-U6
Dia 3					
Per 1: P3-U7	Per 2: P5-U3	Per 3: P2-U2	Per 4: P2-U5	Per 5: P7-U5	Per 6: P1-U2
Continua...					

Dia 4					
Per 1: P3-U4	Per 2: P6-U4	Per 3: P4-U1	Per 4: P5-U7	Per 5: P6-U1	Per 6: P1-U6
Dia 5					
Per 1: P3-U7	Per 2: P6-U5	Per 3: P4-U4	Per 4: P5-U7	Per 5: P6-U1	Per 6: P1-U3
Salon 4					
Dia 1					
Per 1: P4-U7	Per 2: P7-U1	Per 3: P7-U1	Per 4: P3-U5	Per 5: P2-U7	Per 6: P7-U3
Dia 2					
Per 1: P4-U4	Per 2: P6-U1	Per 3: P6-U5	Per 4: P1-U5	Per 5: P2-U7	Per 6: P7-U3
Dia 3					
Per 1: P4-U2	Per 2: P6-U3	Per 3: P7-U1	Per 4: P3-U5	Per 5: P2-U2	Per 6: P7-U5
Dia 4					
Per 1: P4-U3	Per 2: P5-U3	Per 3: P7-U1	Per 4: P3-U4	Per 5: P2-U3	Per 6: P7-U5
Dia 5					
Per 1: P4-U3	Per 2: P5-U7	Per 3: P7-U4	Per 4: P3-U3	Per 5: P1-U6	Per 6: P7-U7
Salon 5					
Dia 1					
Per 1: P5-U6	Per 2: P5-U6	Per 3: P6-U1	Per 4: P2-U6	Per 5: P1-U5	Per 6: P6-U3
Dia 2					
Per 1: P5-U7	Per 2: P2-U1	Per 3: P7-U5	Per 4: P2-U6	Per 5: P1-U5	Per 6: P6-U3
Dia 3					
Per 1: P5-U7	Per 2: P1-U6	Per 3: P4-U2	Per 4: P7-U7	Per 5: P4-U4	Per 6: P6-U2
Dia 4					
Per 1: P5-U6	Per 2: P2-U4	Per 3: P1-U7	Per 4: P7-U7	Per 5: P4-U7	Per 6: P6-U3
Dia 5					
Per 1: P5-U6	Per 2: P2-U1	Per 3: P1-U7	Per 4: P7-U2	Per 5: P4-U3	Per 6: P6-U3
Salon 6					
Dia 1					
Per 1: P6-U1	Per 2: P2-U6	Per 3: P4-U3	Per 4: P1-U2	Per 5: P4-U4	Per 6: P2-U2
Dia 2					
Per 1: P6-U2	Per 2: P5-U5	Per 3: P4-U5	Per 4: P7-U1	Per 5: P4-U4	Per 6: P5-U6
Dia 3					
Per 1: P2-U4	Per 2: P2-U6	Per 3: P3-U4	Per 4: P1-U4	Per 5: P3-U4	Per 6: P5-U4
Dia 4					
Per 1: P6-U3	Per 2: P7-U4	Per 3: P3-U2	Per 4: P1-U5	Per 5: P7-U4	Per 6: P2-U3
Dia 5					
Per 1: P6-U5	Per 2: P7-U4	Per 3: P3-U5	Per 4: P1-U5	Per 5: P2-U3	Per 6: P2-U2
Salon 7					
Dia 1					
Per 1: P2-U1	Per 2: P3-U3	Per 3: P1-U7	Per 4: P4-U4	Per 5: P3-U2	Per 6: P5-U7
Dia 2					
Per 1: P2-U3	Per 2: P7-U6	Per 3: P3-U2	Per 4: P4-U4	Per 5: P3-U2	Per 6: P2-U2
Dia 3					
Per 1: P6-U5	Per 2: P3-U4	Per 3: P6-U1	Per 4: P4-U1	Per 5: P1-U4	Per 6: P2-U7
Dia 4					
Per 1: P2-U3	Per 2: P3-U5	Per 3: P6-U2	Per 4: P4-U6	Per 5: P1-U4	Per 6: P5-U2
Dia 5					
Per 1: P2-U1	Per 2: P3-U7	Per 3: P6-U3	Per 4: P4-U7	Per 5: P7-U4	Per 6: P5-U2

Tabla 4.9: Un ejemplo de horario resultante para el problema Tipo 8.

Salon 1					
Dia 1					
Per 1: P4-U6	Per 2: P4-U4	Per 3: P1-U1	Per 4: P1-U5	Per 5: P7-U7	Per 6: P7-U1
Dia 2					
Per 1: P4-U1	Per 2: P2-U2	Per 3: P6-U1	Per 4: P3-U6	Per 5: P8-U1	Per 6: P7-U2
Dia 3					
Per 1: P4-U6	Per 2: P2-U7	Per 3: P6-U3	Per 4: P3-U7	Per 5: P8-U5	Per 6: P7-U4
Dia 4					
Per 1: P4-U2	Per 2: P2-U8	Per 3: P6-U5	Per 4: P5-U3	Per 5: P8-U7	Per 6: P7-U5
Dia 5					
Per 1: P4-U6	Per 2: P4-U5	Per 3: P6-U8	Per 4: P5-U7	Per 5: P7-U8	Per 6: P7-U6
Salon 2					
Dia 1					
Per 1: P1-U7	Per 2: P2-U2	Per 3: P6-U7	Per 4: P7-U1	Per 5: P6-U5	Per 6: P5-U7
Dia 2					
Per 1: P1-U1	Per 2: P1-U6	Per 3: P2-U7	Per 4: P7-U2	Per 5: P6-U5	Per 6: P5-U7
Dia 3					
Per 1: P1-U2	Per 2: P4-U2	Per 3: P8-U5	Per 4: P7-U4	Per 5: P7-U6	Per 6: P5-U3
Dia 4					
Per 1: P1-U5	Per 2: P4-U7	Per 3: P8-U8	Per 4: P7-U5	Per 5: P6-U1	Per 6: P5-U8
Dia 5					
Per 1: P1-U7	Per 2: P1-U8	Per 3: P3-U2	Per 4: P7-U1	Per 5: P6-U2	Per 6: P5-U8
Salon 3					
Dia 1					
Per 1: P3-U1	Per 2: P8-U1	Per 3: P7-U4	Per 4: P6-U2	Per 5: P4-U2	Per 6: P1-U1
Dia 2					
Per 1: P3-U3	Per 2: P8-U3	Per 3: P5-U3	Per 4: P5-U4	Per 5: P4-U2	Per 6: P1-U7
Dia 3					
Per 1: P3-U6	Per 2: P8-U4	Per 3: P7-U2	Per 4: P6-U3	Per 5: P1-U8	Per 6: P1-U1
Dia 4					
Per 1: P3-U7	Per 2: P8-U5	Per 3: P2-U6	Per 4: P2-U7	Per 5: P4-U4	Per 6: P1-U7
Dia 5					
Per 1: P3-U8	Per 2: P8-U7	Per 3: P7-U4	Per 4: P6-U8	Per 5: P4-U4	Per 6: P1-U3
Salon 4					
Dia 1					
Per 1: P2-U1	Per 2: P6-U6	Per 3: P5-U3	Per 4: P5-U7	Per 5: P1-U5	Per 6: P4-U3
Dia 2					
Per 1: P2-U2	Per 2: P6-U4	Per 3: P7-U4	Per 4: P1-U3	Per 5: P7-U1	Per 6: P4-U3
Dia 3					
Per 1: P2-U8	Per 2: P5-U1	Per 3: P4-U8	Per 4: P4-U7	Per 5: P4-U5	Per 6: P4-U1
Dia 4					
Per 1: P2-U3	Per 2: P6-U5	Per 3: P7-U4	Per 4: P3-U2	Per 5: P7-U3	Per 6: P4-U3
Dia 5					
Per 1: P2-U8	Per 2: P6-U6	Per 3: P5-U4	Per 4: P3-U3	Per 5: P8-U7	Per 6: P4-U1
Salon 5					
Dia 1					
Per 1: P8-U1	Per 2: P5-U7	Per 3: P8-U8	Per 4: P4-U5	Per 5: P3-U6	Per 6: P6-U8
Dia 2					
Per 1: P8-U2	Per 2: P5-U7	Per 3: P8-U8	Per 4: P4-U8	Per 5: P3-U8	Per 6: P6-U3
Dia 3					
Per 1: P8-U4	Per 2: P7-U4	Per 3: P5-U4	Per 4: P1-U5	Per 5: P5-U5	Per 6: P2-U4
Continua...					

Dia 4					
Per 1: P8-U1	Per 2: P7-U6	Per 3: P5-U3	Per 4: P1-U6	Per 5: P3-U2	Per 6: P6-U4
Dia 5					
Per 1: P8-U6	Per 2: P5-U5	Per 3: P8-U8	Per 4: P1-U8	Per 5: P3-U6	Per 6: P6-U8
Salon 6					
Dia 1					
Per 1: P7-U2	Per 2: P3-U3	Per 3: P3-U4	Per 4: P3-U5	Per 5: P2-U1	Per 6: P3-U3
Dia 2					
Per 1: P6-U5	Per 2: P7-U4	Per 3: P1-U2	Per 4: P8-U1	Per 5: P1-U6	Per 6: P2-U2
Dia 3					
Per 1: P7-U5	Per 2: P3-U3	Per 3: P3-U7	Per 4: P5-U5	Per 5: P3-U8	Per 6: P3-U7
Dia 4					
Per 1: P6-U6	Per 2: P5-U4	Per 3: P1-U3	Per 4: P4-U4	Per 5: P1-U2	Per 6: P2-U6
Dia 5					
Per 1: P6-U8	Per 2: P7-U4	Per 3: P1-U5	Per 4: P4-U7	Per 5: P2-U1	Per 6: P8-U8
Salon 7					
Dia 1					
Per 1: P5-U7	Per 2: P7-U3	Per 3: P4-U5	Per 4: P8-U5	Per 5: P5-U3	Per 6: P8-U2
Dia 2					
Per 1: P5-U6	Per 2: P3-U4	Per 3: P4-U6	Per 4: P6-U6	Per 5: P5-U4	Per 6: P8-U1
Dia 3					
Per 1: P5-U7	Per 2: P1-U6	Per 3: P2-U2	Per 4: P8-U4	Per 5: P2-U5	Per 6: P8-U6
Dia 4					
Per 1: P7-U5	Per 2: P3-U8	Per 3: P4-U6	Per 4: P8-U2	Per 5: P2-U6	Per 6: P8-U5
Dia 5					
Per 1: P7-U8	Per 2: P3-U4	Per 3: P2-U3	Per 4: P2-U3	Per 5: P1-U7	Per 6: P2-U1
Salon 8					
Dia 1					
Per 1: P6-U1	Per 2: P1-U4	Per 3: P2-U1	Per 4: P2-U2	Per 5: P8-U2	Per 6: P2-U3
Dia 2					
Per 1: P7-U1	Per 2: P4-U6	Per 3: P3-U1	Per 4: P2-U3	Per 5: P2-U8	Per 6: P3-U2
Dia 3					
Per 1: P6-U2	Per 2: P6-U3	Per 3: P1-U8	Per 4: P2-U5	Per 5: P6-U6	Per 6: P6-U3
Dia 4					
Per 1: P5-U2	Per 2: P1-U8	Per 3: P3-U6	Per 4: P6-U2	Per 5: P5-U5	Per 6: P3-U2
Dia 5					
Per 1: P5-U4	Per 2: P2-U1	Per 3: P4-U6	Per 4: P8-U4	Per 5: P5-U8	Per 6: P3-U7

Capítulo 5

Conclusiones

5.1. Conclusiones

Despúes del análisis de la experimentación y resultados se concluye lo siguiente:

- En cuanto minimizar la cantidad de restricciones violentadas por los horarios generados y encontrar una solución factible mediante la modelación y programación de un Algoritmo Genético, se ratifica que el Algoritmo Genético Simple (AGS) propuesto obtiene buenos resultados en problemas de horarios escolares logrando la función objetivo y con tiempos de ejecución menores en comparación de una búsqueda aleatoria. Utilizar parámetros diversos a los aquí mencionados podría significar más tiempo de cómputo, aunque podría seguir llegando a una solución adecuada.

- En el tema de estudiar diversas configuraciones del algoritmo genético se afirma que con la correcta configuración de parámetros el AG puede minimizar las violaciones a las restricciones del problema mediante la técnica de la penalización y mejorar sus tiempos de ejecución en comparación con otras configuraciones.

- Se determina que para el conjunto de datos estudiado se puede utilizar la siguiente configuración:

1. una población de 300 individuos para dotar al AG de suficiente diversidad.
2. una probabilidad de cruce del 60 % para dar al AG la posibilidad de explotar ciertas áreas del espacio de búsqueda.
3. una probabilidad de mutación de 40 % para proporcionar suficiente capacidad de explorar en todo el espacio de búsqueda, sobre todo si es que el AG se encuentra en un minímo local.
4. configurar 200 mil generaciones de evolución para realizar la búsqueda, también es muy recomendable que el criterio de paro se de al encontrar un horario factible.

- Por último, y a diferencia de otras publicaciones en está área, se muestra el comportamiento del algoritmo evolutivo, en este caso un algoritmo genético simple, en donde es posible notar la dificultad de los problemas y los tiempos empleados, así empíricamente es posible argumentar que un AGS resulta ser una herramienta adecuada para la generación y solución de horarios escolares

5.2. Trabajo futuro

Este trabajo puede ser base para el estudio de problemas de horarios de clases más complejos, es decir, que contemplen más variables como lo son: grupos de alumnos, carreras o áreas de estudio, turnos en la Institución, etc.

Además es posible incluir más restricciones al problema como por ejemplo: disponibilidad horaria del profesor, elección de materias del profesor, profesores suplentes, perfiles del plan de estudio, considerar el tipo de aula (teórica o práctica), el cupo de las aulas, etc.

Se puede pensar en agregar técnicas de programación dinámica que puedan acelerar la generación de un horario viable, éstas pueden mejorar el proceso de exploración del AG, o bien, se puede buscar el mejoramiento del curso de explotación, por medio de evaluaciones

heurísticas adicionales a través de las generaciones.

Se considera que el problema de horarios tiene una gran continuidad con el uso de técnicas mas recientes de AEs como lo son las técnicas multiobjetivo, específicamente los Algoritmos Genéticos Multi-Objetivo (AGMO) permiten atacar problemas con espacios de búsqueda mas complejos, que contemplan mas restricciones y el cumplimiento de más de una función objetivo, haciendo uso de conceptos como el frente de Pareto y la dominancia de Pareto.

También se contempla la aplicación del AG al problema de generación de horarios del Centro Universitario UAEM Valle de México.

Apéndice A

Conjunto de datos

En esta apéndice se presentan las tablas para todos los tipos de problemas utilizados en este trabajo, los cuales fueron obtenidos de la Universidad del Brunel, del autor Beasley [8].

Tabla A.1: Horario Tipo 5.

HT5	P1	P2	P3	P4	P5	
C1	1	0	0	1	2	
C2	2	0	1	3	1	
C3	0	0	1	2	1	
C4	1	2	3	1	0	
C5	0	2	1	4	1	S1
C1	4	0	3	1	3	
C2	0	1	0	1	3	
C3	0	1	2	1	1	
C4	1	0	3	0	0	
C5	2	1	0	2	0	S2
Continua...						

C1	2	2	0	0	2	
C2	3	2	1	2	1	
C3	1	0	3	1	1	
C4	3	0	0	0	3	
C5	1	0	0	1	1	S3
C1	1	2	0	0	1	
C2	0	2	3	1	0	
C3	2	0	2	0	4	
C4	1	3	0	2	1	
C5	1	1	0	2	1	S4
C1	1	1	1	1	1	
C2	0	3	0	0	0	
C3	1	2	3	1	0	
C4	1	2	1	1	1	
C5	1	3	2	2	1	S5

Tabla A.2: Horario Tipo 6.

HT6	P1	P2	P3	P4	P5	P6	
C1	1	0	1	1	0	3	
C2	0	2	0	0	0	0	
C3	0	1	0	1	0	0	
C4	1	0	2	1	0	1	
C5	2	1	1	2	2	1	
Continua...							

C6	0	1	1	1	1	2	S1
C1	2	1	1	0	2	0	
C2	0	0	1	0	2	3	
C3	0	0	0	0	2	1	
C4	1	0	0	0	0	0	
C5	0	2	1	2	2	1	
C6	3	0	1	1	1	0	S2
C1	2	1	1	1	0	0	
C2	1	0	3	0	1	0	
C3	0	0	1	1	2	1	
C4	1	1	1	1	1	2	
C5	1	1	1	0	0	0	
C6	1	1	2	0	1	0	S3
C1	0	2	0	0	2	1	
C2	0	1	1	2	0	2	
C3	0	2	0	0	3	0	
C4	1	1	2	1	0	0	
C5	1	0	1	0	1	2	
C6	1	0	0	1	1	1	S4
C1	0	1	0	0	0	3	
C2	0	1	0	3	1	0	
C3	3	0	1	4	0	2	
C4	2	0	1	0	0	1	
C5	2	1	0	0	0	0	
C6	1	1	0	0	1	1	S5
Continua...							

C1	0	1	1	0	2	0	
C2	0	3	2	1	0	0	
C3	0	1	0	2	1	1	
C4	1	2	2	3	0	0	
C5	1	0	0	0	1	0	
C6	1	1	1	1	0	1	S6

Tabla A.3: Horario Tipo 7.

HT7	P1	P2	P3	P4	P5	P6	P7	
C1	1	0	0	0	2	1	2	
C2	0	1	1	0	0	1	1	
C3	0	1	0	1	1	0	0	
C4	0	0	0	1	1	1	0	
C5	1	0	3	0	0	1	1	
C6	0	1	0	1	0	0	1	
C7	0	0	1	1	1	1	1	S1
C1	2	0	0	1	2	1	0	
C2	0	1	2	1	0	0	0	
C3	0	0	0	0	2	1	0	
C4	1	0	0	0	1	0	0	
C5	1	0	0	0	0	0	1	
C6	4	1	1	3	1	0	0	
C7	1	0	0	1	0	1	0	S2
Continua...								

C1	0	0	1	1	0	2	0	
C2	1	1	0	0	2	1	0	
C3	1	0	1	0	1	0	0	
C4	0	0	2	1	0	1	0	
C5	0	1	1	0	0	1	2	
C6	3	0	0	0	0	0	1	
C7	0	1	2	0	2	0	0	S3
C1	0	0	0	0	0	1	4	
C2	0	1	0	1	0	0	0	
C3	0	1	1	2	1	1	2	
C4	0	0	1	1	0	0	1	
C5	1	0	2	0	0	1	2	
C6	1	0	0	0	0	0		
C7	0	2	0	1	1	0	1	S4
C1	0	2	0	0	0	1	0	
C2	0	0	0	1	0	1	1	
C3	0	0	0	1	0	4	0	
C4	0	1	0	1	0	0	0	
C5	2	0	0	0	0	0	1	
C6	1	2	0	0	4	0	0	
C7	2	0	0	1	2	0	2	S5
C1	0	0	0	0	0	1	1	
C2	1	2	1	0	0	1	0	
C3	0	2	0	1	0	1	0	
C4	1	1	2	2	1	0	3	

Continua...

C5	2	0	1	1	1	1	0	
C6	0	2	0	0	1	0	0	
C7	0	0	0	0	0	0	0	S6
C1	0	2	0	1	0	1	0	
C2	0	1	3	0	2	1	0	
C3	0	2	1	0	0	1	0	
C4	2	0	1	2	0	0	1	
C5	0	0	1	0	0	1	0	
C6	0	0	0	1	0	0	1	
C7	1	1	1	1	1	0	0	S7

Tabla A.4: Horario Tipo 8.

HT8	P1	P2	P3	P4	P5	P6	P7	P8	
C1	1	0	0	1	0	1	1	1	
C2	0	1	0	1	0	0	1	0	
C3	0	0	0	0	1	1	0	0	
C4	0	0	0	1	0	0	1	0	
C5	1	0	0	1	0	1	1	1	
C6	0	0	1	3	0	0	1	0	
C7	0	1	1	0	1	0	1	1	
C8	0	1	0	0	0	1	1	0	S1
C1	1	0	0	0	0	1	2	0	
C2	1	1	1	1	0	1	1	0	
Continua...									

C3	0	0	0	0	1	0	0	0	
C4	0	0	0	0	0	0	1	0	
C5	1	0	0	0	0	2	1	1	
C6	1	0	0	0	0	0	1	0	
C7	2	1	0	1	2	1	0	0	
C8	1	0	0	0	2	0	0	1	S2
C1	2	0	1	0	0	0	0	1	
C2	0	0	0	2	0	1	1	0	
C3	1	0	1	0	1	1	0	1	
C4	0	0	0	2	1	0	2	1	
C5	0	0	0	0	0	0	0	1	
C6	0	1	1	0	0	0	0	0	
C7	2	1	1	0	0	0	0	1	
C8	1	0	1	0	0	1	0	0	S3
C1	0	1	0	2	1	0	1	0	
C2	0	1	1	0	0	0	0	0	
C3	1	1	1	3	1	0	1	0	
C4	0	0	0	0	1	1	2	0	
C5	1	0	0	1	0	1	0	0	
C6	0	0	0	0	0	2	0	0	
C7	0	0	0	1	1	0	0	1	
C8	0	2	0	1	0	0	0	0	S4
C1	0	0	0	0	0	0	0	2	
C2	0	0	1	0	0	0	0	1	
C3	0	0	0	0	1	1	0	0	
Continua...									

C4	0	1	0	0	1	1	1	1	
C5	1	0	0	1	2	0	0	0	
C6	1	0	2	0	0	0	1	1	
C7	0	0	0	0	2	0	0	0	
C8	1	0	1	1	0	2	0	3	S5
C1	0	2	0	0	0	0	0	1	
C2	1	1	0	0	0	0	1	0	
C3	1	0	3	0	0	0	0	0	
C4	0	0	1	1	1	0	2	0	
C5	1	0	1	0	1	1	1	0	
C6	1	1	0	0	0	1	0	0	
C7	1	0	2	1	0	0	0	0	
C8	0	0	1	0	0	1	0	1	S6
C1	0	1	0	0	0	0	0	1	
C2	0	1	0	0	0	0	0	2	
C3	0	2	0	0	1	0	1	0	
C4	0	0	2	0	1	0	0	1	
C5	0	1	0	1	0	0	1	2	
C6	1	1	0	2	1	1	0	1	
C7	1	0	0	0	2	0	0	0	
C8	0	0	1	0	0	0	1	0	S7
C1	0	2	1	0	0	1	1	0	
C2	0	1	2	0	1	2	0	1	
C3	0	2	0	0	0	2	0	0	
C4	1	0	0	0	1	0	0	1	
Continua...									

C5	0	1	0	0	1	0	0	0	
C6	0	0	1	2	0	1	0	0	
C7	0	0	1	0	0	0	0	0	
C8	2	1	0	0	1	0	0	0	S8

Apéndice B

Gráficas en extenso

B.1. Gráficas completas de la experimentación

En este apéndice se presentan las gráficas en extenso, con la configuración encontrada como recomendada en la sección 4.2 y para todos los tipos de problemas exceptuando aquellos que se presentaron durante el desarrollo del trabajo y para los siguientes casos:

B.1.1. Número de generaciones para encontrar algún óptimo

En las gráficas siguientes se puede observar en el eje x la repetición de las 30 ejecuciones, en el eje y la cantidad de generaciones y en el cuadrante los diversos valores que toma cada una de las ejecuciones para llegar a un horario viable, se puede ver que en la mayoria de los casos éste se obtiene, sin embargo, aquellas lineas que tocan el cero en el eje y significa que no lograron un horario con cero choques. En la figuras B.1, B.2, B.3, B.4 se muestran los resultados para los problemas Tipo 4, Tipo 5, Tipo 7 y Tipo 8 respectivamente.

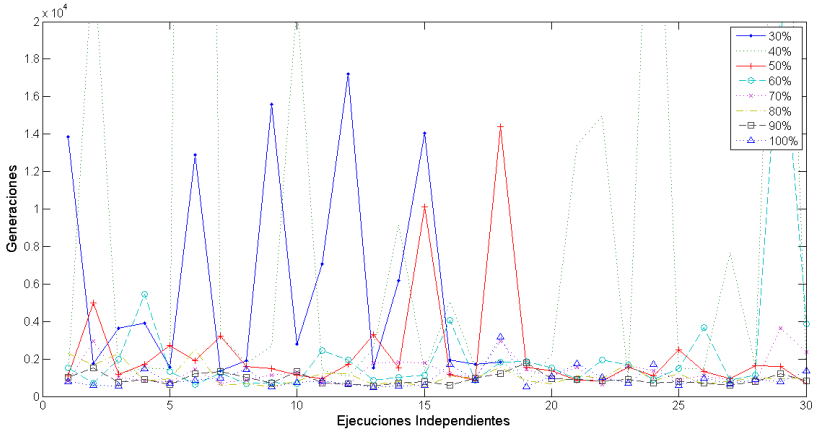


Figura B.1: Número de generaciones para el Tipo 4 con los diferentes porcentajes.

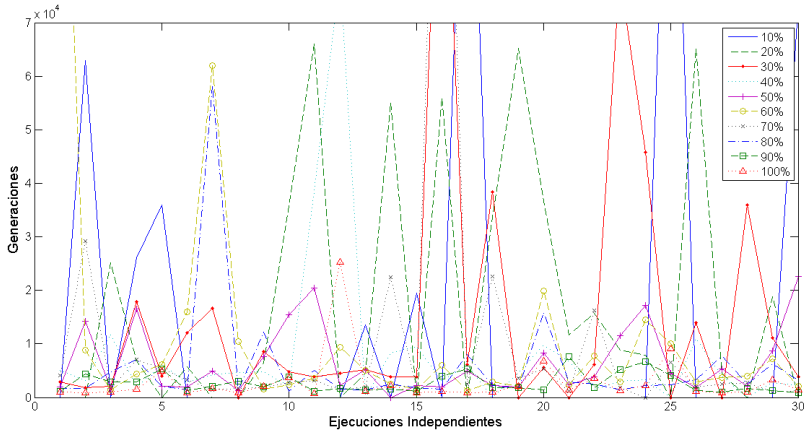


Figura B.2: Número de generaciones para el Tipo 5 con los diferentes porcentajes.

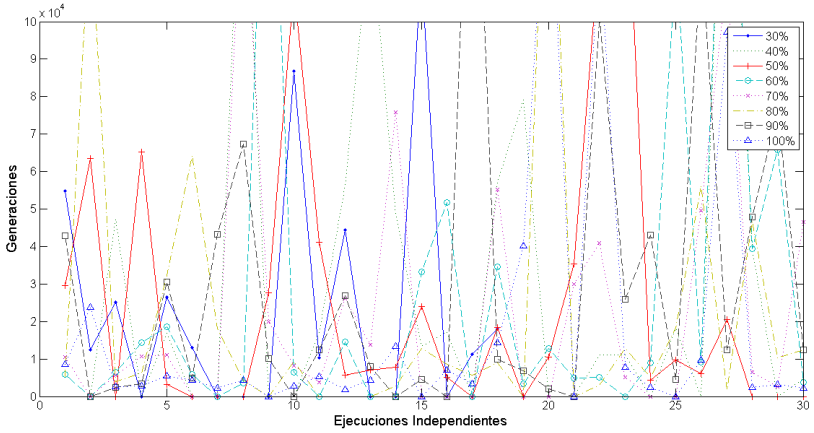


Figura B.3: Número de generaciones para el Tipo 7 con los diferentes porcentajes.

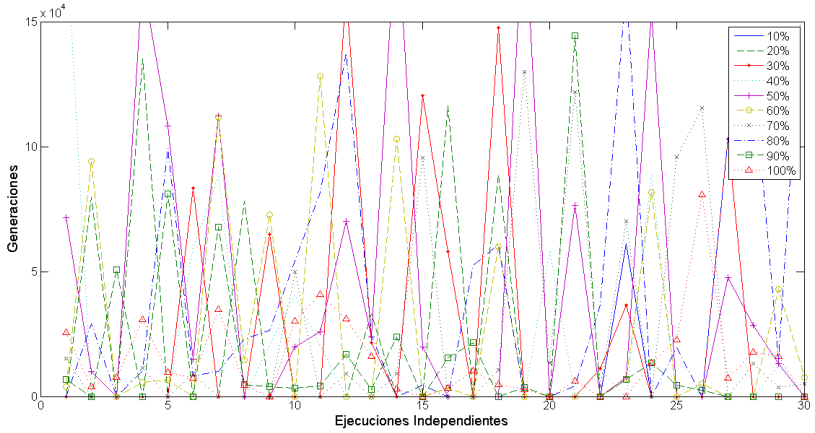


Figura B.4: Número de generaciones para el Tipo 8 con los diferentes porcentajes.

B.1.2. Error promedio durante las ejecuciones

En esta sección se puede observar el comportamiento del error promedio (cantidad de choques) en el eje y, a través de las generaciones en el eje x. Se puede ver que el AG tiende a bajar el error promedio incluso en porcentajes bajos de cruzamiento y que, con porcentajes altos lo hace de una manera más rápida. La Figura B.5 muestra la tendencia para el problema Tipo 4, para el Tipo 5 se muestra la Figura B.6 y respectivamente para los problemas 6, 7 y 8 se muestran las figuras B.7,B.8 y B.9.

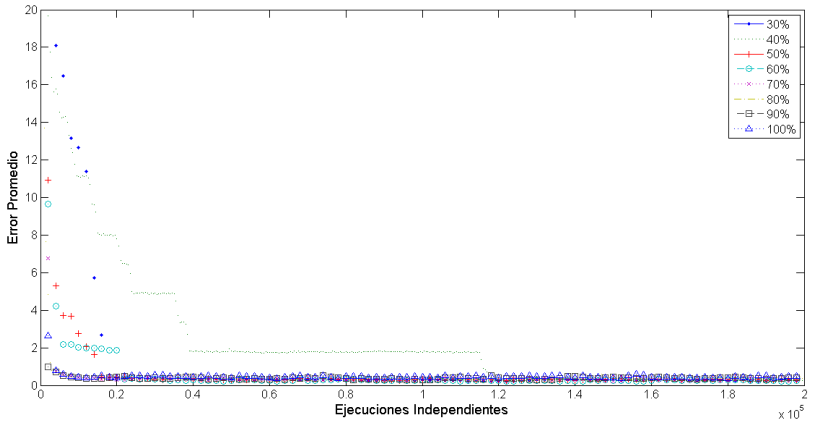


Figura B.5: Error promedio para el Tipo 4 con los diferentes porcentajes.

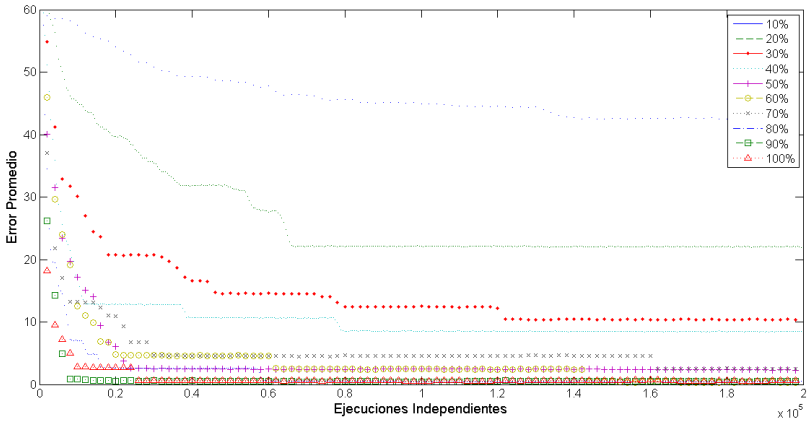


Figura B.6: Error promedio para el Tipo 5 con los diferentes porcentajes.

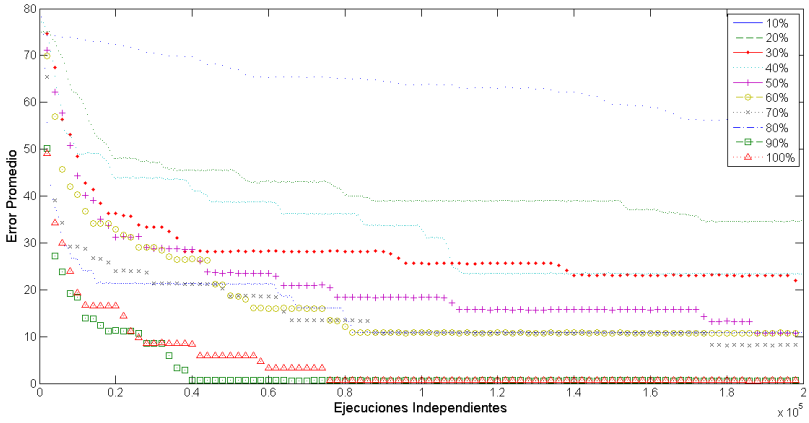


Figura B.7: Error promedio para el Tipo 6 con los diferentes porcentajes.

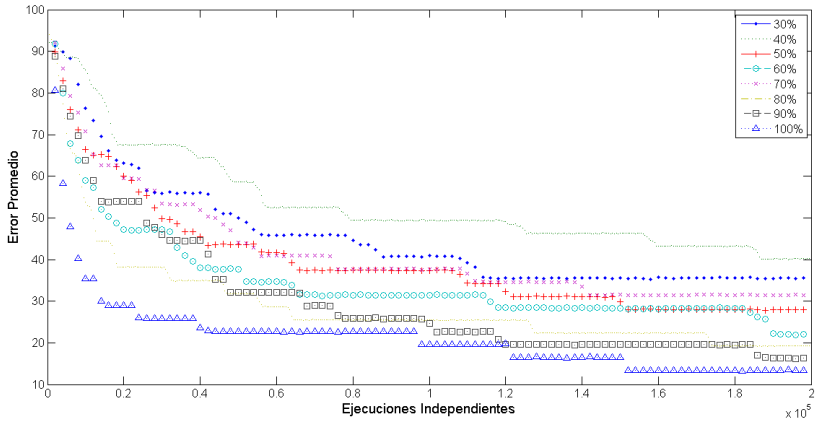


Figura B.8: Error promedio para el Tipo 7 con los diferentes porcentajes.

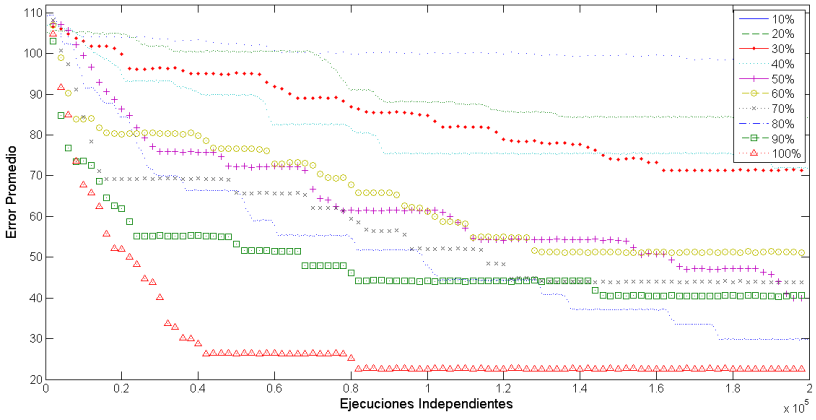


Figura B.9: Error promedio para el Tipo 8 con los diferentes porcentajes.

B.1.3. Tiempo total promedio de todas las ejecuciones

A continuación se pueden ver las gráficas del tiempo que toma al AG alcanzar un horario viable, es decir, que cualquiera de los individuos genera un horario sin choques, se muestra en el eje x las repeticiones de los experimentos y en el eje y la cantidad de tiempo (en horas), las lineas representan el comportamiento de las ejecuciones con los diferentes porcentajes de cruzamiento. Se observa que en la mayoria de los casos se logra un horario factible y que en aquellos con altos porcentajes lo hacen en menor tiempo. Para el Tipo 4 se muestra la Figura B.10, y para los tipos 5, 6, 7 y 8 se encuentran las figuras B.11, B.12, B.13 y B.14, respectivamente.

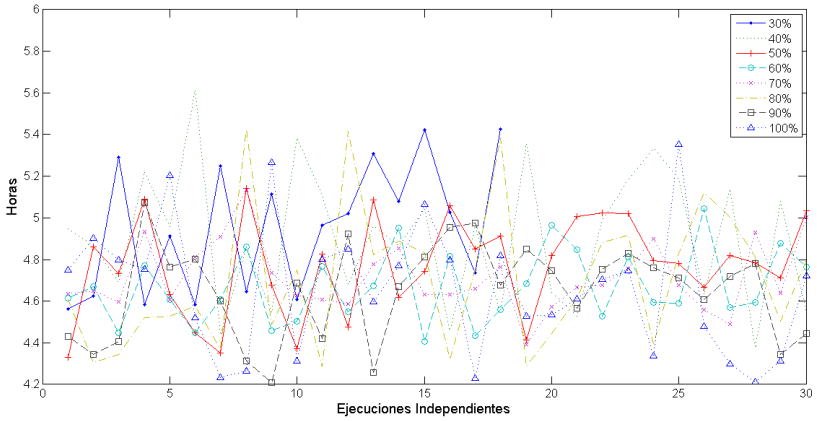


Figura B.10: Tiempo total para el Tipo 4 con los diferentes porcentajes.

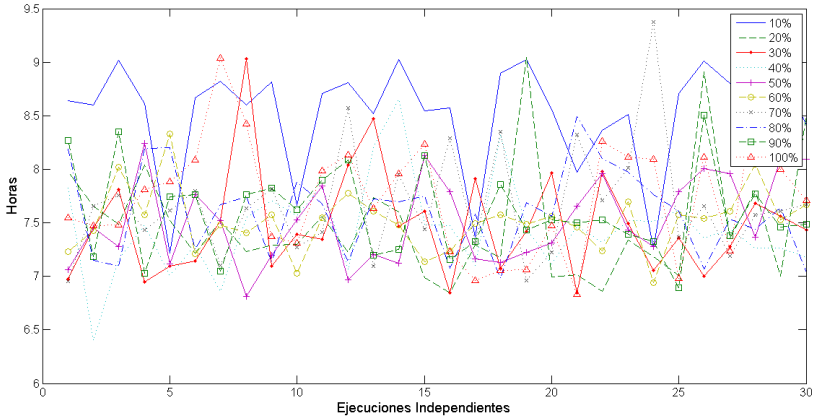


Figura B.11: Tiempo total para el Tipo 5 con los diferentes porcentajes.

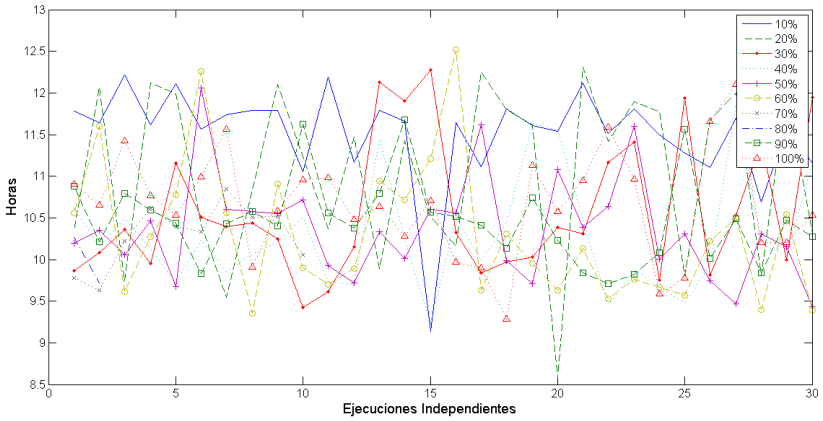


Figura B.12: Tiempo total para el Tipo 6 con los diferentes porcentajes.

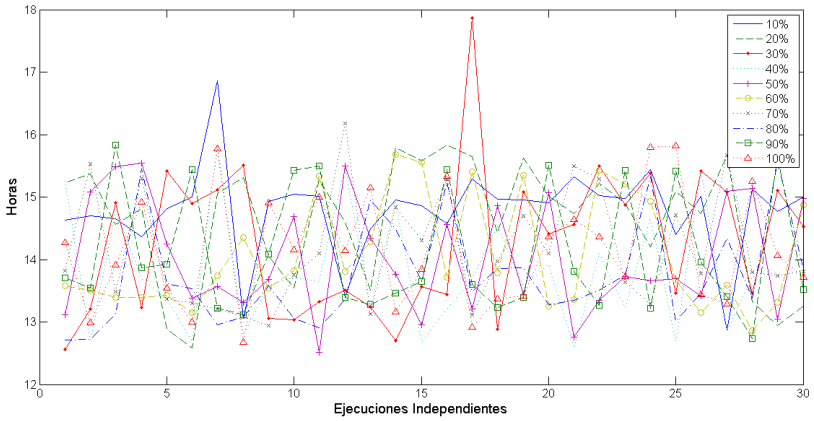


Figura B.13: Tiempo total para el Tipo 7 con los diferentes porcentajes.

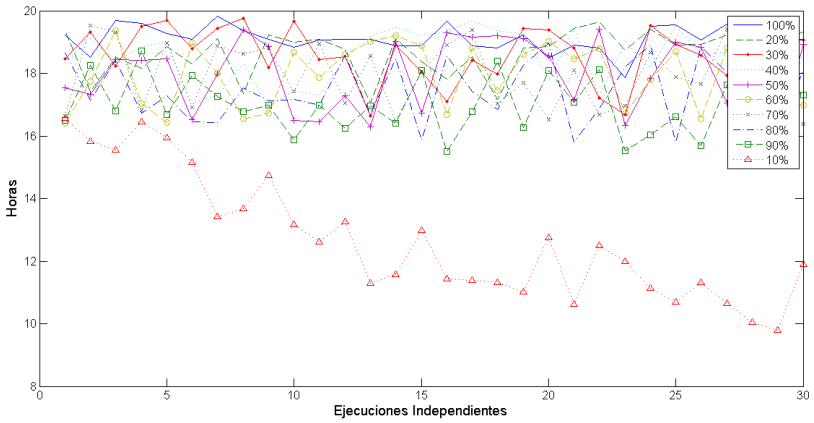


Figura B.14: Tiempo total para el Tipo 8 con los diferentes porcentajes.

B.1.4. Tiempo total para que algún individuo llegue a un óptimo

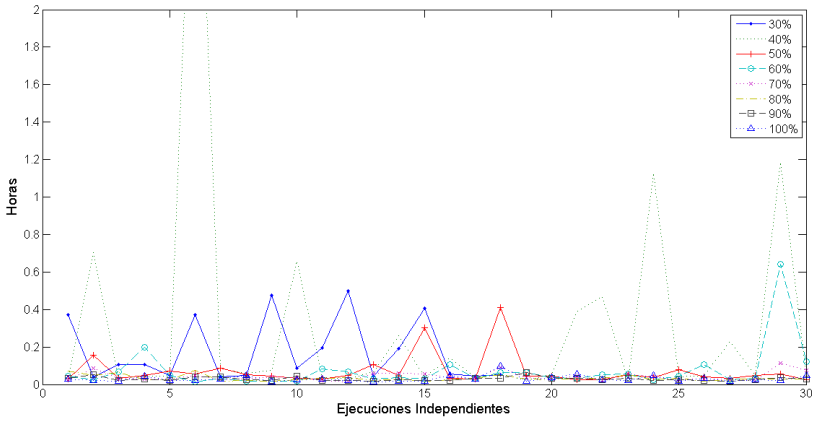


Figura B.15: Tiempo para el óptimo en el Tipo 4 con los diferentes porcentajes.

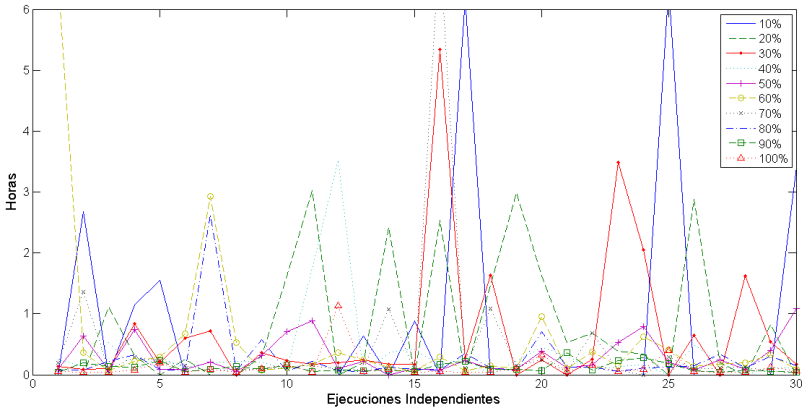


Figura B.16: Tiempo para el óptimo en el Tipo 5 con los diferentes porcentajes.

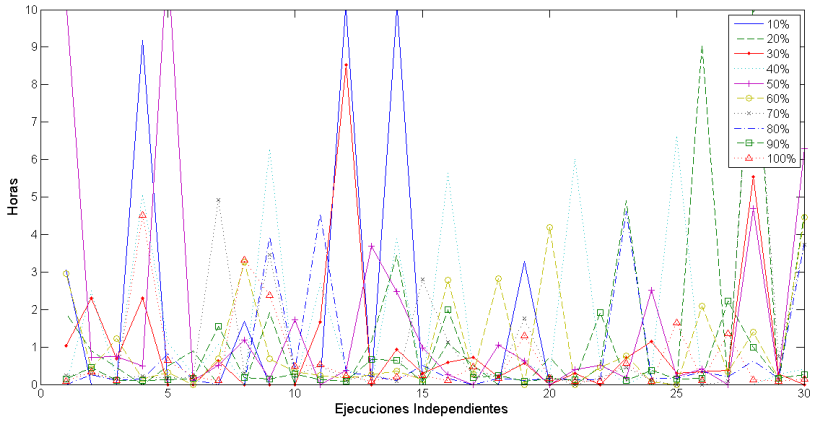


Figura B.17: Tiempo para el óptimo en el Tipo 6 con los diferentes porcentajes.

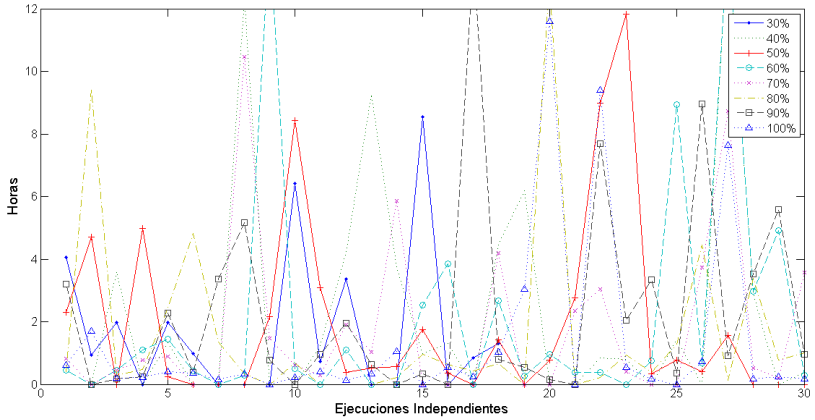


Figura B.18: Tiempo para el óptimo en el Tipo 7 con los diferentes porcentajes.

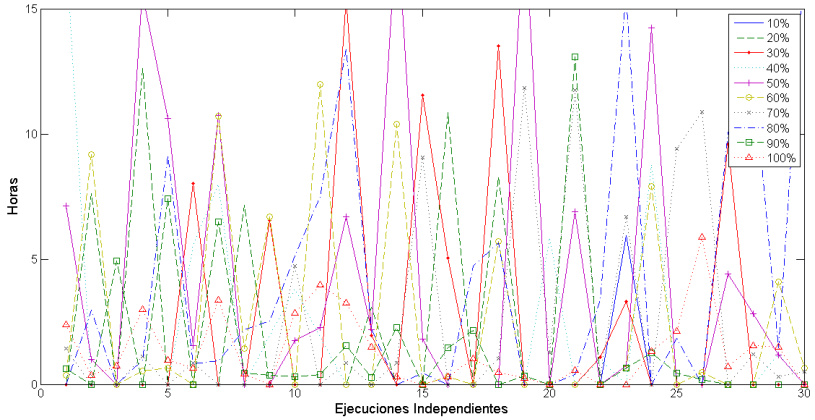


Figura B.19: Tiempo para el óptimo en el Tipo 8 con los diferentes porcentajes.

Referencias

[1] V. A. Bardadym, *Practice and Theory of Automated Timetabling: First International Conference Edinburgh, U.K., August 29–September 1, 1995 Selected Papers*, ch. Computer-aided school and university timetabling: The new wave, pp. 22–45. Berlin, Heidelberg: Springer Berlin Heidelberg, 1996.

[2] A. Garcia-Najera and J. A. Bullinaria, "An improved multi-objective evolutionary algorithm for the vehicle routing problem with time windows," *Computers & Operations Research*, vol. 38, no. 1, pp. 287–300, 2011.

[3] J. C. Rincón O., "Aplicación de algoritmos genéticos en la optimización del sistema de abastecimiento de agua de barquisimeto-cabudare," *Avances en Recursos Hidráulicos*, 2006.

[4] L. V. Santana Quintero and C. A. Coello Coello, "Una introducción a la computación evolutiva y algunas de sus aplicaciones en economía y finanzas = an introduction to evolutionary computation and some of its applications in economics and finance," *Revista de Métodos Cuantitativos para la Economía y la Empresa = Journal of Quantitative Methods for Economics and Business Administration*, vol. 2, no. 1, pp. 3–26, 2006.

[5] J. Zankis, S.H., "Heuristics 'optimization': why, when, and how to use it.," *Interfaces*, vol. 11, no. 5, 1981.

[6] R. Raghavjee and N. Pillay, "An informed genetic algorithm for the high school timetabling problem," in *Proceedings of the 2010 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists*, SAICSIT '10, (New York, NY, USA), pp. 408–412, ACM, 2010.

[7] R. Raghavjee and N. Pillay, "A study of genetic algorithms to solve the school timetabling problem," in *Advances in Soft Computing and Its Applications* (F. Castro, A. Gelbukh, and M. González, eds.), vol. 8266 of *Lecture Notes in Computer Science*, pp. 64–80, Springer Berlin Heidelberg, 2013.

[8] J.E.Beasley, "Or-library timetabling [en linea]," 2016.

[9] R. Raghavjee and N. Pillay, "An application of genetic algorithms to the school timetabling problem," in *Proceedings of the 2008 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists on IT Research in Developing Countries: Riding the Wave of Technology*, SAICSIT '08, (New York, NY, USA), pp. 193–199, ACM, 2008.

[10] D. E. Goldberg, "Real-coded genetic algorithms, virtual alphabets, and blocking," *Complex Systems*, vol. 5, pp. 139–167, 1990.

[11] R. et al., *Inteligencia Artificial Un Enfoque Moderno*. Prentice Hall, 2004.

[12] C. Darwin, *On the origin of species*. New York :D. Appleton and Co.,, 1804. http://www.biodiversitylibrary.org/bibliography/28875.

[13] L. Cervigón, Carlos; Araujo, *Algoritmos evolutivos: un enfoque práctico*. Alfaomega, Ra-Ma, 2009.

[14] C. Blum, "Ant colony optimization: Introduction and recent trends," 2005.

[15] Z. M. Raymond Chiong, Thomas Weise, *Variants of Evolutionary Algorithms for Real-World Applications*. Springer-Verlag Berlin Heidelberg, 2012.

[16] J. H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. Cambridge, MA, USA: MIT Press, 1992.

[17] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs (3rd Ed.)*. London, UK, UK: Springer-Verlag, 1996.

[18] F. Herrera, M. Lozano, and J. L. Verdegay, "Tackling real-coded genetic algorithms: Operators and tools for behavioural analysis," *Artificial Intelligence Review*, vol. 12, pp. 265–319, 1998.

[19] T. Bäck, F. Hoffmeister, and H.-P. Schwefel, "A survey of evolution strategies," in *Proceedings of the Fourth International Conference on Genetic Algorithms*, pp. 2–9, Morgan Kaufmann, 1991.

[20] J. R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. Cambridge, MA, USA: MIT Press, 1992.

[21] J. R. Koza, *Genetic Programming II: Automatic Discovery of Reusable Programs*. Cambridge, MA, USA: MIT Press, 1994.

[22] J. E. S. William E. Hart, Natalio Krasnogor, *Recent Advances in Memetic Algorithms*. Springer-Verlag Berlin Heidelberg, 2005.

[23] R. C. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," *Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, pp. 39–43, 1995.

[24] Y. Shi and R. Eberhart, "A modified particle swarm optimizer," in *Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on*, pp. 69–73, May 1998.

[25] M. Dorigo and T. Stützle, *Ant Colony Optimization*. Scituate, MA, USA: Bradford Company, 2004.

[26] E. Tsang, "A glimpse of constraint satisfaction," *Artificial Intelligence Review*, vol. 13, no. 3, pp. 215–227, 1999.

[27] F. Rossi, P. v. Beek, and T. Walsh, *Handbook of Constraint Programming (Foundations of Artificial Intelligence)*. New York, NY, USA: Elsevier Science Inc., 2006.

[28] V. Suarez, "Class schedule assignment based on students learning rhythms using a genetic algorithm," *Ingeniería y Ciencia*, vol. 9, pp. 77–95, 2013.

[29] A. Colorni, M. Dorigo, and V. Maniezzo, "A genetic algorithm to solve the timetable problem," 1993.

[30] C. D. von Lücken Martínez, "Algoritmos evolutivos para optimización multiobjetivo: Un estudio comparativo en un ambiente paralelo asíncrono," Master's thesis, Universidad Nacional de Asunción, 2003.

[31] J. F. F. Mauricio Granada E., Eliana M. Toro, "Programación óptima de horarios de clase usando un algoritmo memético," *Scientia Et Technica*, vol. XII, pp. 255–260, 2006.

[32] A. Öner, S. Özcan, and D. Dengi, "Optimization of university course scheduling problem with a hybrid artificial bee colony algorithm.," in *IEEE Congress on Evolutionary Computation*, pp. 339–346, IEEE, 2011.

[33] L. Lalescu, "Free timetabling software [en linea]," 2016.

[34] aSc Applied Software Consultants, "Asc horarios [en linea]," 2016.

[35] M. S. Ltd, "Mimosa scheduling software [en linea]," 2016.

[36] S. Ltd, "Scientia timetable scheduling [en linea]," 2016.

[37] W. T. Ltd, "Timetabling software [en linea]," 2016.

[38] J. Larrosa and P. Meseguer, "Restricciones blanda: Modelos y algoritmos.," *Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial*, vol. 7, no. 20, pp. 69–82, 2003.

[39] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1st ed., 1989.