Evaluación de Algoritmos Genéticos Aplicados a la Resolución del Problema de Generación de Horarios

Autor:
Daniel Alberto Alegría Sánchez

Asesor Interno:
M.C. Néstor Antonio Morales Navarro

Asesor Externo:
Dr. Noel Enrique Rodríguez Maya

Reporte Final de Residencia Profesional

Instituto Tecnológico de Tuxtla Gutiérrez

Ingeniería en Sistemas Computacionales

3 de Febrero de 2017

Contenido

Lista de Imágenes **I**

Lista de Tablas **II**

1 Justificación **1**

2 Objetivos **2**
2.1 Objetivo General . . . 2
2.2 Objetivos Específicos . . . 2

3 Problemática a Resolver **3**

4 Procedimiento y descripción de las actividades realizadas **5**
4.1 Obtención de datos . . . 5
4.2 Ordenación de los datos . . . 5
4.3 Desarrollo del algoritmo genético . . . 6
4.3.1 Codificación de los individuos . . . 6
4.3.2 Generación inicial de individuos . . . 6
4.3.3 Parámetros a evaluar . . . 7
4.3.4 Restricciones . . . 8
4.3.5 Función Objetivo . . . 8
4.3.6 Cruza . . . 9
4.3.7 Mutación . . . 9
4.4 Obtención de resultados . . . 10
4.4.1 Obtención de tamaño de población, numero de generaciones y método de selección . . . 10
4.4.2 Obtención del porcentaje de mutación . . . 10

5 Resultados **11**

6 Anexo **14**
6.1 Individuos . . . 14
6.1.1 Inicialización . . . 14
6.1.2 Codificación . . . 14
6.1.3 Representación de las horas . . . 15
6.2 Métodos de selección . . . 16
6.2.1 Torneo . . . 16
6.2.2 Selección por rangos . . . 17
6.2.3 Método de la ruleta . . . 18
6.2.4 Muestreo estocástico universal . . . 19

7 Conclusiones y Recomendaciones **20**

8 Competencias desarrolladas y/o aplicadas **21**
8.1 Investigación 21
8.2 Desarrollo de software 21
8.3 Pensamiento crítico 21
8.4 Comunicación verbal y escrita 21

Lista de Imágenes

1 Comparativa de métodos de selección 11
2 Comparativa entre diferentes porcentajes de mutación 12
3 Ejemplo de codificación 15

Lista de Tablas

1 Ejemplo de horario 1 9
2 Ejemplo de horario 2 9
3 Resultados de métodos de selección 11
4 Resultados de porcentajes de mutación 13
5 Representación de un horario 15
6 Ejemplo de representación de tiempos 15
7 Mejor configuración encontrada 20

1 Justificación

La calendarización de horarios esta presente en varios sectores de la sociedad, desde los transportes y trabajo hasta en las escuelas. Esta actividad es conocida por representar una alta complejidad al momento de realizarse, ya que se busca que los recursos que se necesiten calendarizar estén distribuidos de una manera optima y eficiente. Es por esto que este problema se aborda como un problema de optimización.

En el caso de horarios para escuelas, los recursos manejados son maestros, aulas y alumnos. El objetivo es distribuir a estos en un horario de manera que ninguno quede fuera de este, pero también que su inclusión no sea problemática para los demás recursos, por ejemplo, no se podrá asignar un maestro para dos grupos diferentes el mismo día y a la misma hora.

Como es de suponerse, la realización manual de estos horarios implica un gran numero de dificultades, ya que se trata de manejar muchas restricciones a la par de un gran numero de variables. Esto conlleva a que el proceso manual sea muy lento e impreciso para el horario y muy tedioso para el encargado de dicha labor.

Debido a esto han sido muchas las aportaciones que se han hecho a la generación automática de horarios [1, 2]. Esta tarea ha sido abordada desde varias perspectivas, principalmente algoritmos heurísticos. Dentro de estos algoritmos sobresalen los algoritmos evolutivos, que se basan en el proceso de reproducción y selección natural para resolver problemas de optimización, como es el caso del problema de horarios.

El Instituto Tecnológico de Tuxtla Gutiérrez es una institución educativa que actualmente ofrece nueve licenciaturas, dos maestrías y dos doctorados. Esta institución brinda clases a alrededor de 5000 alumnos, estos están distribuidos en dos turnos: matutino y vespertino. También tiene en su currículo a mas de 200 docentes, cada uno especializado en una cierta área de estudios. Para cubrir las necesidades especificas de ciertas carreras, el ITTG cuenta con laboratorios especializados, como son laboratorios de Física, Química, Mecánica, Eléctrica, entre otros.

Cada semestre se tiene que elaborar un horario para cada grupo de alumnos de esta institución. Este horario debe de tener en cuenta, además de las características ya descritas, las preferencias de horarios de los maestros, las restricciones de horarios de los alumnos y las restricciones de cada salón de clases.

Como se menciono anteriormente, el proceso de generar horarios es una tarea que esta expuesta a errores, sobretodo si se hace de manera manual.

Es por esta razón que se cree que es conveniente que esta institución cuente con una herramienta que permita hacer esta tarea, minimizando los errores y reduciendo el tiempo en que es llevada a cabo.

2 Objetivos

2.1 Objetivo General

Desarrollar una comparativa entre el rendimiento de diferentes configuraciones de algoritmos genéticos orientados al momento de resolver el problema de generación de horarios de manera que se aprecie las características de cada horario generado con el fin de observar cual configuración es mas optima.

2.2 Objetivos Específicos

- Analizar el estado del arte referente a resolución del problema de horarios utilizando algoritmos genéticos
- Describir detalladamente el problema de generación de horarios
- Identificar métodos de selección y mutación mas utilizados en algoritmos genéticos
- Identificar porcentajes de cruza y mutación mas utilizados en algoritmos genéticos
- Obtener datos escolares para generar horarios
- Construir algoritmos de acuerdo a los datos recolectados
- Recolectar información de cada algoritmo al momento de generar el horario
- Graficar datos
- Crear cuadro comparativo en donde se puedan observar los datos obtenidos

3 Problemática a Resolver

La idea de resolver el problema de generación de horarios surge en los años 60's con la publicación de artículos que proponían el uso de la computadora como herramienta de apoyo en la resolución de este problema.

Los primeros trabajos presentaban la solución a través de un proceso muy similar al realizado manualmente. Estos también se caracterizaban por no poder transmitir de manera apropiada el enfoque heurístico que utilizaban, además de solo generar una solución [3].

El primer intento no heurístico fue propuesto por C. Gotlieb en 1963, este trabajo describía la generación de horarios por medio de un proceso de reducción de disponibilidades.

El algoritmo desarrollado detectaba asignaciones estrictas y disponibilidades a través de problemas de menor complejidad. Enfocándose en un solo participante o en una sola hora se puede obtener una sección bidimensional que representa un problema de asignación, en ocasiones revela asignaciones estrictas o disponibilidades e incluso si el problema tiene o no solución. Estas asignaciones o disponibilidades causan disponibilidades o asignaciones adicionales, estableciendo así un proceso iterativo de reducción. Si durante este proceso no se puede demostrar que el problema no tiene solución, el proceso se volverá estacionario, siempre y cuando todas las situaciones sean evaluadas. El resultado es llamado Arreglo de Disponibilidades Reducidas [4].

En 1993, [5] compara el proceso de programación de horarios con dos versiones de un algoritmo genético (con y sin búsqueda binaria), el método de recocido simulado y el de búsqueda tabú contra horarios programados manualmente. Además presenta un modelo y una estructura jerárquica en la función objetivo y los operadores genéticos que se aplicaran a las matrices que representan el horario.

Los autores enlistan las dificultades que surgen al resolver problemas de optimización combinatoria con algoritmos genéticos. Las mas relevantes surgen al momento de aplicar los operadores genéticos, ya que esto puede dar lugar a soluciones erróneas. Para evitar esto proponen las siguientes soluciones:

- Cambiar la representación de la solución de manera que el operador de cruza pueda aplicarse consistentemente
- Crear operadores de mutación que puedan generar únicamente soluciones factibles
- Aplicar operadores de mutación y cruza para después de realizar un proceso de reparación genética que convierta soluciones infactibles en soluciones factibles a través del uso de un algoritmo de filtrado

Además enlistan la manera en que las restricciones son manejadas, esto con el fin de generar mejores soluciones en cada generación. El manejo de restricciones se hace de la siguiente forma:

- Por los operadores genéticos, para que el conjunto de horas asignadas a un maestro no sea cambiada por los operadores de cruza y muta
- Por el algoritmo de filtrado, así los errores generados por los operadores genéticos son eliminados, total o parcialmente

- Por la función objetivo, así se limita la selección de individuos con soluciones factibles, de manera que entre menos apto sea el individuo, mayor sea su penalización

Como se puede apreciar, la calendarización de horarios es una tarea compleja. y el realizar este proceso de manera manual, se expone a que el resultado no sea optimo, sin mencionar de que es una tarea tediosa para el o los encargados de realizarla, por lo que lo recomendable es utilizar herramientas que faciliten este proceso, y optimicen el resultado final.

La generación de horarios utilizando este tipo de herramientas varia en torno a la representación que se le da al problema, por ejemplo, un horario para una universidad es diferente al de una escuela normal, incluso, la forma en que los horarios se representan varían de universidad a universidad. Ciertas universidades crean módulos de cuatro horas por materia, mientras que otras crean módulos de una o dos horas.

Además de estas variaciones, se tiene que tomar en cuenta los parámetros de estas herramientas, en el caso de algoritmos genéticos se tiene que contar al menos seis parámetros diferentes. Estos son: numero de población, máximo de generaciones, porcentaje de cruza, porcentaje de muta, método de selección, entre otros.

Este trabajo pretende analizar cada uno de los problemas implicados en la generación de horarios, además de evaluar cada uno de los parámetros utilizados por los algoritmos genéticos, esto con la finalidad de encontrar una configuración optima que se adecue a las necesidades del Instituto Tecnológico de Tuxtla Gutiérrez.

4 Procedimiento y descripción de las actividades realizadas

El desarrollo del proyecto se dividió en cuatro partes:

1. Obtención de datos
2. Ordenación de datos
3. Desarrollo del algoritmo genético
4. Obtención de resultados

Hay que indicar que la parte de obtención de datos ya se había hecho antes de iniciar la residencia profesional. Este semestre se trato de obtener datos actualizados, pero no fue posible.

4.1 Obtención de datos

Par obtener los datos primero se acudió con el jefe de la División de Estudios Superiores, el Ing. Juan José Arreola. Esta división es la que, entre otras actividades, se encarga de de realizar los horarios cada semestre. Para poder entregar los datos requeridos, esta división pidió que se proporcionara un escrito de nuestra parte, en donde se indicase el nombre del proyecto, el nombre de las personas encargadas de desarrollarlo, el objetivo del proyecto y los datos requeridos.

Los datos requeridos eran:

- Lista de maestros (nombres, horas de entrada, horas de salida, materias impartidas)
- Lista de salones (capacidad de alumnos, carrera(s) a la que pertenecen)
- Lista de grupos de alumnos (tamaño, carrera, etc)

Despues de entregar este escrito, la División de Estudios Superiores nos facilito en un archivo los datos que se requerían.

4.2 Ordenación de los datos

El archivo proporcionado por la Division era un horario de un semestre anterior. Este horario era el horario general del ITTG. Lo que se hizo en esta etapa fue extraer únicamente los datos que se necesitaban.

Aunque se pudieron extraer los nombres de los maestros, no hubo forma de obtener sus horarios de entrada y de salida, por lo que lo que se hizo fue obtener la primer y la ultima hora en que se le asignaban clases. Esto nos dio el rango de trabajo de cada maestro.

Aunque los datos ya ordenados contenían información de todo el ITTG, para el desarrollo del proyecto se utilizo solamente datos de la carrera de Ingeniería en Sistemas Computacionales.

4.3 Desarrollo del algoritmo genético

El algoritmo fue desarrollado en Java, utilizando MySQL para el manejo de base de datos. Al principio se pensó en utilizar la librería ECJ, que es una librería especializada en computo evolutivo, pero debido a que los métodos de selección, la mutación y la cruza ya estaban escritos en Java, esta idea se descarto.

4.3.1 Codificación de los individuos

Para la codificación de los individuos se tuvieron en cuenta varias representaciones. La primera fue de la forma

$$I = \{C, L, T_1 ... T_n, Cl_1 .. Cl_n\}$$

en donde C representa el curso, L el maestro asignado, T el conjunto de horas asignadas y Cl el salón de clases, teniendo en cuenta que para ambos casos, T y Cl, n representa el numero de horas correspondiente al curso.

El principal problema encontrado con esta codificación es que no podría saber con certeza cuando se presentarían choques entre salones, maestros y cursos, por lo que fue descartada.

La segunda codificación que se probo fue una de la forma

$$I = \begin{array}{c|cccc} & T_1 & T_2 & \dots & T_{70} \\ Cl_1 & C,L & C,L & \dots & C,L \\ Cl_2 & C,L & C,L & \dots & C,L \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ Cl_n & C,L & C,L & \dots & C,L \end{array}$$

donde T representa a las horas, Cl representa los salones de clases y el conjunto C, L representa a las relaciones Curso-Maestro.

Una explicación detallada aclarando la codificación de los individuos se puede encontrar en 6.1.

4.3.2 Generación inicial de individuos

Después de decidirse la manera en que los individuos se codificaran, se busco la forma en que estos se generaran.

Al principio se opto por una generación totalmente aleatoria. Después de hacer varias pruebas, esta forma de generación se descarto, ya que se perdía mucho tiempo al momento de evaluar los individuos. Además, no se podía deliberar si el individuo violaba restricciones flexibles o inflexibles. También surgía un problema al momento de realizar la cruza, ya que la mayoría de las veces los nuevos individuos tenían valores de aptitud peores que los de generaciones anteriores, lo que provocaba que el algoritmo nunca alcanzara valores óptimos.

Debido a esto, se tomo la decisión de que los individuos serian horarios validos, es decir, que no violaran ninguna de las restricciones inflexibles, y que el algoritmo genético se encargara de tomar estas soluciones validas y convertirlas a soluciones optimas, tratando de minimizar únicamente las restricciones flexibles, ya que, en un problema con tantas restricciones, se corre el riesgo de crear un algoritmo genético que pase la mayor parte del tiempo evaluando individuos no validos [6].

Un proceso similar se lleva a cabo al momento de generar nuevos individuos a través de la recombinación y la mutación.

4.3.3 Parámetros a evaluar

Aunque existen muchos métodos de selección utilizados en los algoritmos genéticos, se decidió evaluar solamente cinco. Estos métodos fueron elegidos ya que se observo que son los mas utilizados para realizar esta tarea. Los métodos son:

- Torneo
- Torneo binario
- Rango lineal
- Ruleta
- Muestreo estocástico universal

Cada algoritmo utiliza una cierta probabilidad de mutación. Los parámetros utilizados para este valor son los siguientes

- 0.1%
- 0.2%
- 0.3%
- 0.4%
- 0.5%
- 0.6%
- 0.7%
- 0.8%
- 0.9%
- 1.0%

Para el numero de generaciones y el tamaño de población se utilizaron 5 valores entre el 100 y el 500.

4.3.4 Restricciones

Restricciones Inflexibles:

- Un maestro, grupo o salón de clases no puede tener superposiciones de cualquier tipo
- Las clases para el turno matutino comienzan a las 7am y terminan, como máximo, a las 4pm
- Las clases para el turno vespertino comienzan, como máximo, a las 12pm y terminan a las 9pm
- Los salones de clases debe de cumplir con los requerimientos de la clase asignada

Restricciones Flexibles:

- Las horas asignadas para cada maestro deben de estar dentro de su horario de trabajo
- Los grupos y maestros no deberán de contener tiempos muertos
- Las clases deberán de estar dispuestas en módulos de dos horas
- El tamaño de los salones debe de coincidir con el tamaño del grupo asignado
- La distribución de las clases entre los salones debe de ser uniforme
- Todos los maestros deben de tener asignados materias

4.3.5 Función Objetivo

Evaluar los individuos que generamos es una tarea critica, ya que de esto depende el resultado final de nuestro algoritmo genético. Debido a que los individuos que generamos no violan las restricciones inflexibles, solo los evaluamos de acuerdo a sus restricciones flexibles.

La función Objetivo se puede representar de la siguiente manera:

$$F = S + O + T + G$$

en donde S se representa como la cantidad de salones cuyo tamaño no coincide con el de la clase asignada.

O es la cantidad de salones cuyo porcentaje de disponibilidad esta muy por debajo o por encima de la disponibilidad media. La disponibilidad de un salón se obtiene de la siguiente manera:

$$D = \frac{70 - clasesAsignadas}{70} * 100$$

De esta manera, si un salon no tiene ninguna clase asignada su disponibilidad será 100%, por ejemplo.

T requiere de dos valores para calcularlos:

$$T = T_1 + T_2$$

donde T_1 es la cantidad de maestros a los que no se asignaron materias y T_2 es la cantidad de maestros a los que no se les respeto sus horas de trabajo.

G se puede representar como como la desviación estándar de las horas libres. La formula utilizada es:

$$\sigma = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \overline{x})^2}$$

donde N es el numero de clases que tiene por días y x es la diferencia de horas que hay entre estas clases. Si el resultado es mayor a 0.5, se aplica la penalización. Por ejemplo:

El grupo S2A tiene 5 clases el día lunes. Estas clases son las siguientes:

Table 1: Ejemplo 1

Materia	Hora
POO	1
POO	2
FISICA	6
FISICA	7

Table 2: Ejemplo 2

Materia	Hora
POO	1
POO	2
FISICA	4
FISICA	5

Las diferencias entre las horas para el primer ejemplo son las siguientes: 1, 4 y 1, por lo tanto, $\sigma = 1.41$. Este caso se penalizaría. Las diferencias entre las horas del segundo ejemplo son 1,2,1, por lo tanto $\sigma = 0.47$. Este caso no se penalizaría ya que, aunque las horas no son consecutivas, solo hay una hora libre.

4.3.6 Cruza

El método de cruza utilizado es el de un punto.

En este método se selecciona un punto aleatorio dentro de los genes del individuo. Después se intercambian los genes de los padres a partir de este punto generando así dos nuevos individuos [7].

Este operador también se encarga de que los nuevos individuos sean validos, al no violar ninguna restricción inflexible.

4.3.7 Mutación

El método de mutación utilizado es mutación uniforme [8].

Cuando la mutación ocurre, el gen seleccionado es dado un valor nuevo. En este caso se entiende como gen a una relación maestro-curso dentro del horario generado que cuando es mutado recibe una nueva posición.

4.4 Obtención de resultados

Esta etapa del trabajo se dividió en dos partes. Primero obtendríamos el valor adecuado para el tamaño de población, el numero de generaciones y el método de selección, después se obtendría el valor para el porcentaje de mutación.

4.4.1 Obtención de tamaño de población, numero de generaciones y método de selección

Al principio se tenia planeado hacer una comparativa con 6 tamaños de población y 5 números de generaciones, pero al ver que el algoritmo alcanzaba resultados óptimos en el primer experimento, solo se probó con un numero de generaciones.

En total en esta etapa se trabajo con 30 variaciones del algoritmo, cada variación se ejecuto 5 veces para poder sacar un promedio en el valor de aptitud final.

4.4.2 Obtención del porcentaje de mutación

Una vez que se obtuvieron los valores anteriores solo quedaba descubrir el mejor porcentaje de mutación. Para esto se crearon diez variaciones del algoritmo y estas fueron ejecutadas 5 veces para poder sacar un promedio.

5 Resultados

Esta etapa del proyecto se dividió en dos partes. La primera consiste en elegir que método de selección y tamaño de población dan mejores resultados, y la segunda es ver que porcentaje de mutación es el mas optimo.

Originalmente se tenia pensado evaluar el numero de generaciones, los valores que se iban a evaluar eran 100, 200, 300, 400 y 500. Esto ya no fue necesario, ya que después de hacer el experimento con un numero de generaciones de 100, los resultados obtenidos eran muy óptimos.

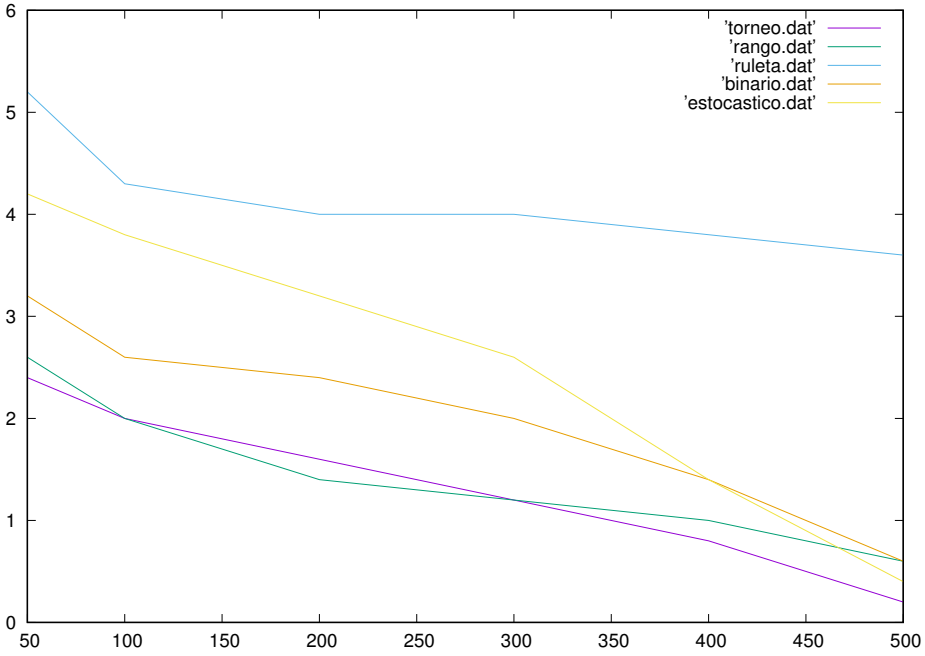

Figure 1: Comparación del rendimiento de cinco diferentes métodos de selección con varios tamaños de población. Los tamaños de población utilizados fueron 50, 100, 200, 300, 400 y 500. El eje Y representa la aptitud media del mejor individuo.

Tamaño de poblacion	Torneo	Rango	Ruleta	Torneo binario	Muestreo Estocástico
50	2.4	2.6	5.2	3.2	4.2
100	2	2	4.3	2.6	3.8
200	1.6	1.4	4	2.4	3.2
300	1.2	1.2	4	2	2.6
400	0.8	1	3.8	1.4	1.4
500	0.2	0.6	3.6	0.6	0.4

Table 3: Tabla con los valores de cada método de selección

Como se puede observar, el método de selección que peor rendimiento tiene es el de ruleta, mientras que el que mejor tiene es el de torneo. Cabe destacar que el método de muestreo estocástico universal comienza con un valor similar al método de la ruleta, pero conforme el numero de población aumenta, el rendimiento también aumenta hasta el punto de que la aptitud media de su mejor individuo es similar al del método de selección por torneo.

El siguiente paso fue de evaluar al mejor método de selección con varios porcentajes de mutación. Los resultados fueron los siguientes:

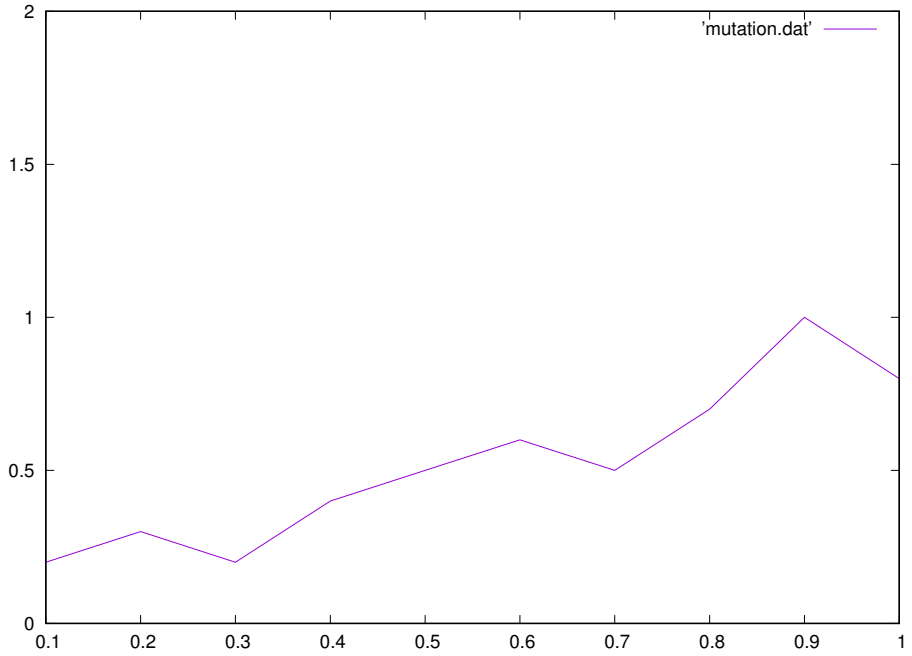

Figure 2: Comparación del rendimiento del método de selección por torneo con diferentes valores de mutación. El eje Y representa el valor de aptitud promedio mientras que el eje X representa los porcentajes de mutación.

Porcentaje de mutación	Aptitud media
1%	0.2
2%	0.3
3%	0.2
4%	0.4
5%	0.5
6%	0.6
7%	0.5
8%	0.7
9%	1.0
10%	0.8

Table 4: Tabla con la aptitud media del mejor individuo generado con cada porcentaje de mutacion

Como se observa, a menor porcentaje de mutación, mejor valor de aptitud.

6 Anexo

6.1 Individuos

En 4.3.2 se menciono de manera breve la forma en que los individuos se generan. En esta sección se tratara de ampliar esta información.

6.1.1 Inicialización

Antes de asignar horas y salones a las clases, se asignan los maestros. Primero se toman los maestros que pueden dar esa materia, después se elige uno de forma aleatoria:

```
private void initCourses() {
    for (Course course : courses) {
        Gene gene = new Gene.Builder().setCourse(course).
            setTeacher(getRandomTeacher(course)).build();

        for (int i = 0; i < course.getHours(); i++) {
            genes.add(gene);
            gene.getTeacher().addCourse(course);
        }
    }
}
```

Después de asignar un maestro, se crean varias copias de este conjunto. Este numero de copias es dado por la cantidad de horas que la materia tiene asignada:

```
private void arrangeCourses() {
    for (Course course : courses) {
        ArrayList<Gene> courseGenes =
            genes.stream().filter(gene ->
            gene.getCourse().equals(course))
             .collect(Collectors.toCollection(ArrayList<Gene>::new));
        getCoursesPosition(courseGenes);
    }
}
```

6.1.2 Codificación

Cada individuo es representado por un arreglo de dos dimensiones, la dimensión de ese arreglo es de n x 70, en donde n es dado por el numero de salones que ocuparemos. El numero 70 es dado por el numero de horas en las que es posible dar clases. En el ITTG la primer hora de clases es a las 7am, mientras que la ultima hora es a las 9pm. Eso da un total de 14 horas, que multiplicadas por 5, ya que las clases son de lunes a viernes, nos da ese resultado.

Cada posición de este arreglo se puede definir como una clase en el que no se ha definido ni el curso ni el maestro. Después de inicializar el algoritmo, a cada clase se le asigna una

posición. Este valor representa una posición dentro de este arreglo. Al asignar esta clase a esa posición, ese espacio dentro del arreglo deja de estar disponible.

A continuación se hace una representación mas grafica de esto.

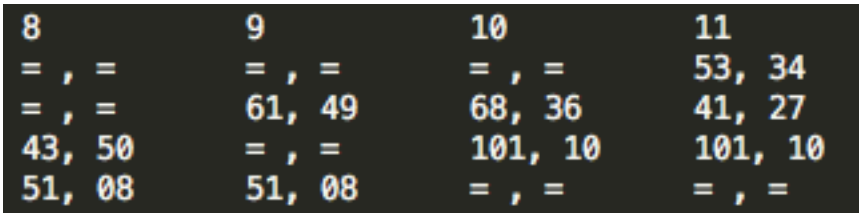

Figure 3: En esta imagen se aprecia un extracto de un individuo. La primer fila de numeros representa la hora y cada fila es un salón. Se representa el rango de horas de 2pm a 6pm y los salones 1, 2, 3 y 4.

	L14-15	L15-16	L16-17	L17-18
Salon1	-	-	-	TBDS5B, T34
Salon2	-	LAUTS6B, T49	GRS8B, T36	TPROGS4B, T27
Salon3	BDDS4B, T50	-	INTARTS9B, T10	INTARTS9B, T10
Salon4	TELCS5B, T08	TELCS5B, T08	-	-

Table 5: Esta tabla es la representación del horario de la imagen anterior. Cada celda esta formada por el nombre de la clase correspondiente y el identificador del profesor.

6.1.3 Representación de las horas

Para representar las horas se utiliza la idea del tiempo.

La idea de un tiempo es sencilla, en lugar de representar la hora en que una clase es impartida como un rango de horas, se representa como una cantidad de horas.

Por ejemplo, en vez de decir que la clase c toma lugar los lunes de 7am a 8am, de 8am a 9am, los martes de 8am a 9am y los jueves de 8am a 9am, se dice que esta clase toma lugar en los tiempos 1, 2, 16 y 44.

Un tiempo consta de dos valores, uno absoluto y otro relativo, el valor absoluto es un numero entre el 1 y el 70 mientras que el relativo es un numero entre 1 y 14. El valor absoluto representa al tiempo en comparación a todo el horario, mientras que el relativo lo representa únicamente para un día determinado.

Por ejemplo, el tiempo 1 representa tanto absolutamente como relativamente, el día lunes de 7am a 8am. En comparación, si quisiéramos representar las mismas horas pero para un día diferente, digamos jueves, el valor absoluto seria 43, mientras que el relativo seguiría siendo 1.

Hora	Lunes 7am-8am	...	Miercoles 4pm-5pm	...	Viernes 8pm-9pm
Tiempo Relativo	1	...	10	...	14
Tiempo Absoluto	1	...	38	...	70

Table 6: Ejemplo de como se representa un tiempo

6.2 Métodos de selección

6.2.1 Torneo

La idea de la selección por torneo es simple. Tomar n individuos de nuestra población, seleccionar al mejor de este grupo y repetir hasta que sea necesario [9].

El numero de individuos puede variar. Para el desarrollo de este trabajo se utilizaron dos tipos de torneo, uno con dos individuos, torneo binario, y otro con cinco individuos.

A continuación se muestran el código de ambos métodos de selección.

```
/*
* Torneo binario
*/
private void binaryTournamentSelection() {
        matingPool.clear();
        for (int i = 0; i < poolSize; i++) {
            Individual individualA =
               population.getIndividual(Methods.getRandomNumber(0,
               population.getSize() - 1));
            Individual individualB =
               population.getIndividual(Methods.getRandomNumber(0,
               population.getSize() - 1));

            if (individualA.getFitness() <
               individualB.getFitness()) {
                matingPool.add(individualA);
            } else {
                matingPool.add(individualB);
            }
        }
    }

/*
* Torneo de n individuos
*/
private void tournamentSelection() {
        matingPool.clear();

        for (int i = 0; i < poolSize; i++) {
            ArrayList<Individual> competitors = new ArrayList<>();

            for (int j = 0; j < tournamentSize; j++) {
                competitors.add(population.getIndividual(Methods.getRandomNumber(0,
                   population.getSize() - 1)));
            }
```

```
            Individual best = competitors.get(0);

            for (int j = 0; j < tournamentSize; j++) {
                if (competitors.get(j).getFitness() <
                    best.getFitness()) {
                    best = competitors.get(j);
                }
            }

            matingPool.add(best);
        }
    }
```

6.2.2 Selección por rangos

La selección por rangos consiste en ordenar la población de acuerdo a su valor de aptitud, y asignar a cada individuo un valor de acuerdo a su nueva posición con respecto a la población. Por ejemplo, si se encuentra en la primer posición su valor será de 1, en la segunda ser 2, etc. El código utilizado es el siguiente:

```
private void rankSelection() {
        matingPool.clear();
        population.sortByFitness();
        population.reverse();

        int size = population.getSize();

        for (int i = 1; i <= size; i++) {
            population.getIndividual(i - 1).setRank(i);
        }

        population.shuffle();

        int pool = 0;

        while (pool < poolSize) {
            for (Individual individual :
                population.getIndividuals()) {
                int rand = Methods.getRandomNumber(0,
                    population.getSize());

                if (rand <= individual.getRank() && pool <
                    poolSize) {
                    matingPool.add(individual);
                    pool++;
```

```
                }
            }
        }
    }
```

6.2.3 Método de la ruleta

Este método de selección da a cada individuo dentro de la población una probabilidad de ser elegido de acuerdo a su valor de aptitud.

Su nombre se debe al hecho de que asignar a los individuos una probabilidad de selección de acuerdo a su aptitud es como asignarles un espacio dentro de una ruleta, entre mas aptos sean, mas espacio tendrán [10].

```
    private void rouletteWheelSelection() {
        matingPool.clear();
        population.sortByFitness();

        population.getIndividuals().stream().forEach(individual ->
                individual.setExpectedFitness((individual.getFitness()
                    == 0) ? 2 : 1 / individual.getFitness()));

        population.reverse();

        double sumFitness = 0.0;
        for (Individual individual : population.getIndividuals())
            {
            sumFitness += individual.getExpectedFitness();
            individual.setExpectedFitness(sumFitness);
        }

        int pool = 0;
        while (pool < poolSize) {
            double rand = Methods.getRandomDouble(0, sumFitness);

            for (int i = 0; i < population.getSize(); i++) {
                if (rand <=
                    population.getIndividual(i).getExpectedFitness())
                    {
                    matingPool.add(population.getIndividual(i));
                    pool++;
                    break;
                }
            }
        }
    }
```

6.2.4 Muestreo estocástico universal

Este método es una variación del método de la ruleta. Busca ser imparcial, además de tener una propagación mínima [11].

```
    private void stochasticUniversalSelection() {
        matingPool.clear();
        population.sortByFitness();

        population.getIndividuals().stream().forEach(individual ->
                individual.setExpectedFitness((individual.getFitness()
                   == 0) ? 2 : 1 / individual.getFitness()));

        population.reverse();

        double sumFitness = 0.0;
        for (Individual individual : population.getIndividuals())
           {
            sumFitness += individual.getExpectedFitness();
            individual.setExpectedFitness(sumFitness);
        }

        double distance = sumFitness / poolSize;
        double start = Methods.getRandomDouble(0, distance);

        for (double i = start; i < sumFitness; i += distance) {
            for (Individual individual :
               population.getIndividuals()) {
                if (i <= individual.getExpectedFitness()) {
                    matingPool.add(individual);
                    break;
                }
            }
        }
    }
```

7 Conclusiones y Recomendaciones

De acuerdo a los resultados obtenidos podemos concluir que la mejor configuración utilizando nuestro algoritmo seria la siguiente:

Método de Selección	Porcentaje de Mutación	Tamaño de Población	Numero de Generaciones
Torneo de 5 individuos	1%	500	100

Table 7: Mejor configuración.

Sin embargo, no se descarta de que mas adelante se hagan mas pruebas. Esto debido a los resultados arrojados por el método de selección por ruleta.

Aunque cada método de selección fue probado para verificar su funcionamiento, el método de la ruleta arrojo unos resultados que varían mucho en comparación a los demás métodos. Esto puede indicar que puede estar mal codificado, por lo que en un futuro se planea corregirlo.

Otro comportamiento inesperado es el de los porcentajes de mutación, ya que entre mas alto sea el porcentaje, peores individuos genera. Tal vez el problema esta en la manera en que asigna las nuevas posiciones, ya que en algunos casos la nueva posición es igual a la antigua. En un futuro se tratara de arreglar este comportamiento.

También se pretende explorar nuevas formas de generar individuos, ya que la que se usa actualmente esta propensa a errores al momento de generar nueva población o mutar individuos, como ya se menciono antes.

Además se planea experimentar con nuevos métodos de cruza, como los propuestos por [12] y [13]

8 Competencias desarrolladas y/o aplicadas

Las competencias desarrolladas durante este proyecto fueron cuatro:

- Investigación
- Desarrollo de software
- Pensamiento crítico
- Comunicación verbal y escrita

8.1 Investigación

Al principio del proyecto la idea que se tenia acerca de un algoritmo genético era muy vaga, fue a través de la investigación y de la ayuda de los asesores que esta idea fue tomando forma.

Tampoco se tenia muy claro cual era el objetivo final del proyecto. Fue a través de la investigación que se fueron descubriendo trabajos similares a este. Esto fue de gran ayuda para crear una idea general de que y como iba a desarrollarse este proyecto.

8.2 Desarrollo de software

Aunque ya se tenían conocimientos acerca de desarrollo de software, estos eran muy básicos. Fue a través del desarrollo de este trabajo que estos conocimientos se fueron ampliando.

A través del desarrollo del algoritmo fue que los conocimientos acerca del lenguaje de programación Java fueron creciendo. Esto a través del descubrimiento de buenas practicas de programación, convenciones para la escritura de código y patrones de desarrollo.

8.3 Pensamiento crítico

Muchas veces durante el avance de un trabajo no podemos ver la cantidad de errores que estamos cometiendo debido a que no se es lo suficientemente autocrítico. El desarrollo de esta competencia permitió poder identificar los errores que se estaban cometiendo y arreglarlos, aunque esto significase empezar desde cero.

8.4 Comunicación verbal y escrita

La tarea de recolectar datos, entrevistarse con gente responsable de la generación de horarios, redacción de documentos y redacción de reportes requiere de excelentes habilidades de comunicación. Al principio del proyecto el no poder expresar con claridad lo que se necesitaba y cual era el objetivo de este trabajo trajo una serie de conflictos y malentendidos.

Con el desarrollo de este proyecto estas habilidades fueron mejorando gradualmente y, aunque no están del todo desarrolladas, gracias a este proyecto el desarrollo de trabajos futuros será mas sencillo.

References

[1] A. Schaerf, "A survey of automated timetabling," *Artificial Intelligence Review*, vol. 13, no. 2, pp. 87–127, 1999.

[2] H. Babaei, J. Karimpour, and A. Hadidi, "A survey of approaches for university course timetabling problem," *Computers and Industrial Engineering*, vol. 86, pp. 43–59, 2015.

[3] J. Appleby, D. Blake, and E. Newman, "Techniques for producing school timetables on a computer and their application to other scheduling problems," *The Computer Journal*, vol. 3, no. 4, pp. 237–245, 1961.

[4] C. Gotlieb, "The construction of class-teacher timetables," 1963.

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

[6] W. Erben and J. Keppler, "A genetic algorithm solving a weekly course-timetabling problem," in *International Conference on the Practice and Theory of Automated Timetabling*. Springer, 1995, pp. 198–211.

[7] F. Alabsi and R. Naoum, "Comparison of selection methods and crossover operations using steady state genetic based intrusion detection system," *Journal of Emerging Trends in Computing and Information Sciences*, vol. 3, no. 7, pp. 1053–1058, 2012.

[8] N. Soni and T. Kumar, "Study of various mutation operators in genetic algorithms," *International Journal of Computer Science and Information Technologies*, vol. 5, no. 3, pp. 4519–4521, 2014.

[9] D. E. Goldberg and K. Deb, "A comparative analysis of selection schemes used in genetic algorithms," *Foundations of genetic algorithms*, vol. 1, pp. 69–93, 1991.

[10] K. Jebari and M. Madiafi, "Selection methods for genetic algorithms," *International Journal of Emerging Sciences*, vol. 3, no. 4, pp. 333–344, 2013.

[11] J. E. Baker, "Reducing bias and inefficiency in the selection algorithm," in *Proceedings of the second international conference on genetic algorithms*, 1987, pp. 14–21.

[12] E. Yu and K.-S. Sung, "A genetic algorithm for a university weekly courses timetabling problem," *International transactions in operational research*, vol. 9, no. 6, pp. 703–717, 2002.

[13] D. Mittal, H. Doshi, and M. Sunasra, "Automatic timetable generation using genetic algorithm," 2015.