IMPLEMENTACIÓN DE ALGORITMOS GENÉTICOS PARA LA RESOLUCIÓN DEL PROBLEMA SCHOOL TIMETABLING PARA LAS INSTITUCIONES EDUCATIVAS

MAURICIO ANDRES GUERRA CUBILLOS

ERWIN HAMID PARDO QUIROGA

UNIVERSIDAD DISTRITAL “FRANCISCO JOSÉ DE CALDAS”

BOGOTÁ D.C.

COLOMBIA

2014

IMPLEMENTACIÓN DE ALGORITMOS GENÉTICOS PARA LA RESOLUCIÓN DEL PROBLEMA SCHOOL TIMETABLING PARA LAS INSTITUCIONES EDUCATIVAS

MAURICIO ANDRES GUERRA CIBILLOS

20091078058

ERWIN HAMID PARDO QUIROGA

20091078078

PROYECTO DE GRADO PRESENTADO COMO REQUISITO PARA OPTAR AL TÍTULO DE TECNÓLOGO EN SISTEMATIZACIÓN DE DATOS

TUTOR-DIRECTOR:

ROBERO EMILIO SALAS RUIZ

UNIVERSIDAD DISTRITAL “FRANCISCO JOSÉ DE CALDAS”

BOGOTÁ D.C.

COLOMBIA

2014

NOTA DE ACEPTACIÓN

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Tutor

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Jurado

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Jurado

TABLA DE CONTENIDO

INTRODUCCIÓN………………………………………………………………………..5

* 1. Fase de definición, planeación y organización……………………………….….6

1.1.1 Titulo del trabajo………………………………….………………….……….6

1.1.2 Tema………………………………….…………………………………….….6

1.1.3 Planteamiento del Problema…………………………………………..…….6

1.1.3.1 Descripción del Problema…………………………………….……….6

1.1.3.2 Formulación del Problema…………………………………….………7

1.1.4 Alcances y Delimitaciones………………………………………………..….7

1.1.4.1 Alcances…………………………………………………….……….….7

1.1.4.2 Delimitaciones…………………………………………………………..7

1.1.5 Objetivos………………………………….………………………….…..…….8

1.1.5.1 Objetivo General………………………………………………………..8

1.1.5.2 Objetivos Específicos…………………………………………….…….8

1.1.6 Justificación………………………………….…………..…………………….8

1.1.7 Marco de Referencia………………………………………………………….9

1.1.7.1 Marco Histórico………………………………………………………….9

1.1.7.2 Marco Teórico………………………………………………………….15

1.1.7.3 Marco Conceptual………………………………………………….….19

1.1.8 Factibilidad………………………………….……………..………………….23

1.1.8.1 Técnica………………………………………………………………….23

1.1.8.2 Operativa……………………………………………………….……….24

1.1.8.3 Legal………………………………………………………………….….24

1.1.9 Cronograma de Actividades……………………………………………...….25

1.2 Fase de Análisis………………………………….………………………………….26

1.2.1 Análisis de requerimientos del sistema y del software…………...………26

1.2.1.1 Requerimientos Humanos…………………………………………….26

1.2.1.2 Requerimientos de Hardware…………………………………..…….26

1.2.1.3 Requerimientos de Software………………………………………….26

1.2.2 Diagrama de Funciones……………………………………………………...27

1.3 Fase de Diseño……………………………………………………….………….….28

1.3.1 Diseño Entidad Relación…………………………………………………….28

1.3.2 Diagramas de Caso de Uso………………………………………..……….29

1.3.3 Documentación de Casos de Uso………………………………………….32

1.3.4 Diagramas de Secuencia…………………………………………...……….37

1.3.5 Diagramas de Actividad………………………………………………….….42

1.3.6 Diseño de Interfaz de Usuario……………………………………………...56

1.4 Fase de Implementación……………………………………….…………………..63

1.5 Fase de Pruebas……………………………………………………………..……..64

1.6 Fase de Implantación

BIBLIOGRAFIA

**RESUMEN**

Este documento presenta un trabajo de investigación para la realización de un sistema capaz de realizar la generación automática de horarios de clase para los colegios, dando así solución al problema School Timetabling, implementando algoritmos genéticos para resolver dicho problema. El argumento teórico muestra los diferentes métodos con los cuales es posible llegar a dar solución al problema del School Timetabling, estudiando con un poco más de profundidad el tema central de los Algoritmos Genéticos, su historia y su desarrollo. Así mismo, se entrega el concepto del Timetabling, sus clasificaciones, características y complejidad computacional.

Finalmente, se muestran los resultados del desarrollo del algoritmo genético para la resolución del problema School Timetabling utilizando el lenguaje de programación Java, aplicando el producto final al colegio María Mercedes Carranza, que en este caso funciona como prueba piloto.

**ABSTRACT**

**INTRODUCCIÓN**

Gracias a la gran dificultad de resolución y al tamaño del problema que representa el Timetabling, este está clasificado dentro de los problemas NP-Completos, que son los más complejos de resolver, sin embargo, existen una variedad de métodos que han sido implementados exitosamente dando soluciones optimas a dicho problema dentro de los cuales están el Recocido Simulado, las Redes Neuronales, Búsqueda Tabú y los Algoritmos Genéticos, entre otros. Timetabling hace referencia a la calendarización, específicamente a la organización de horarios de diferentes organizaciones y/o eventos tales como hospitales, colegios, universidades, centros de transporte, fixtures deportivos, etc. Para el caso específico de este proyecto, se dará resolución al problema del School Timetabling, que es la rama que estudia la organización de los horarios de clase para los colegios, para lo cual se deben tener en cuenta una gran variedad de parámetros tales como la asignación de profesores, cursos, aulas y periodos de tiempo, para obtener la mejor solución.

Como bien se mencionaba anteriormente, los Algoritmos Genéticos son uno de los métodos mediante los cuales se ha dado resolución al problema del Timetabling, y es este método precisamente el estudiado e implementado para la realización de este proyecto.

Los algoritmos genéticos, ofrecen cierta variedad de características como simplicidad, efectividad y eficiencia a la hora de brindar una solución, y como una de las características que lo diferencian de los demás métodos, es que maneja N soluciones dependiendo de la naturaleza del problema, es decir, los Algoritmos Genéticos pueden brindarle al usuario 1 o varias soluciones distintas al problema del Timetabling, cada una de ellas igualmente valida.

En resumen, este proyecto da solución al problema del School Timetabling con la implementación de Algoritmos Genéticos, usando como prueba piloto para la visualización de los resultados el Colegio María Mercedes Carranza.

**1.1 Fase de definición, planeación y organización**

**1.1.1 Título del trabajo:** IMPLEMENTACIÓN DE ALGORITMOS GENÉTICOS PARA LA RESOLUCIÓN DEL PROBLEMA SCHOOL TIMETABLING PARA LAS INSTITUCIONES EDUCATIVAS

**1.1.2 Tema**. Durante el desarrollo del proyecto se van a tratar dos (2) temas principales como lo son School Timetabling y los Algoritmos Genéticos.

**1.1.3 Planteamiento del problema**

**1.1.3.1 Descripción del Problema:** Ya sea al finalizar un periodo académico, o culminando la etapa de vacaciones, diversas instituciones educativas coinciden en la organización y realización del horario académico para el nuevo ciclo académico. Para la elaboración de un horario de un colegio, generalmente se cuentan con recursos que deben ser asignados a periodos de tiempo establecidos, recursos como: asignaturas, profesores, cursos y aulas. Estos deben o en algunos casos pueden, cumplir una serie de restricciones que se ajusten a cada institución; restricciones obligatorias o duras como por ejemplo:

* La disponibilidad de los profesores en periodos de tiempo.
* Cruce entre 2 o más cursos, en un mismo salón en el mismo periodo de tiempo.

Restricciones como las anteriores, deben cumplirse para garantizar que un horario sea válido y viable para su implementación. Por otra parte, se encuentran las restricciones deseadas o blandas, la violación de alguna de estas seguirá ocasionando un horario factible, pero no de la calidad deseada, por ejemplo:

* La asignatura Matemáticas no se debería dictar en la última hora de cada día.

La asignación de recursos como docentes, materias y salones en periodos de tiempo dados por cada institución o colegio, que cumplan con restricciones, es un problema de programación horaria conocido como *“School Timetabling”*. El espacio de búsqueda compuesto por los recursos, variables y restricciones que se maneja en este problema, lo hace bastante complejo y de un cuidadoso manejo, por esto se dice que este problema está situado dentro de los problemas NP-Completos de la complejidad computacional.

En muchos colegios en el país, se realiza el horario de clases de forma manual, lo cual se torna bastante tedioso y costoso en cuanto al tiempo necesario para esta labor, tanto en primaria como en educación media; teniendo en cuenta que en muchos casos los recursos y las restricciones varían de un periodo a otro, por ej.: en el año 2011 un colegio cuenta con 3 cursos en el grado sexto y 2 en séptimo, para el año 2012 lo más probable es, que para el grado séptimo se cuente con 3 cursos y para el grado octavo 2. De esta manera se dice que el espacio varía, al igual que la asignación de los recursos en el nuevo horario de clases cambia.

Cuando esta asignación se realiza manualmente generalmente, no se optimiza ni se explora totalmente el espacio de búsqueda, tampoco, se encuentra la mejor solución (entiéndase como la solución que mejor satisfaga a todos los recursos y cumpla con todas las restricciones), simplemente se examina que las restricciones obligatorias se cumplan, para que el horario académico sea factible y en algunos casos se dejan de lado restricciones deseadas, como por ejemplo: preferencias de las asignaturas y períodos de dictado por parte de los docentes.

Sin embargo, considerando la asignación de recursos para la realización de horarios de clases, como un problema de optimización, surge la idea de automatizar el proceso de realización de horarios académicos, con la implementación de técnicas de Inteligencia Artificial, para dar solución al problema, con la obtención de horarios correctos. Una de estas técnicas son los *Algoritmos Genéticos,* ya que utilizan una población de soluciones, siendo menos sensibles a quedar atrapadas en óptimos locales (posibles soluciones, pero no la mejor) que las técnicas que utilizan una solución única.

De esta manera, se busca desarrollar una herramienta computacional, que permita explorar el espacio de búsqueda creado en cada colegio, teniendo en cuenta las restricciones para al final dar una solución, en este caso un horario de clases factible.

Para observar y posteriormente analizar la solución obtenida se realizará una prueba piloto del software en el Colegio María Mercedes Carranza IED, dicho software debe tener la característica de permitir cargar información, correspondientes a los recursos que trata este problema y la implementación de la técnica Algoritmos Genéticos.

**1.1.3.2 Formulación del Problema:** ¿Cómo implementar algoritmos genéticos para la resolución del problema School Timetabling en la generación de horarios de las instituciones educativas?

**1.1.4 Alcances y Delimitaciones:**

**1.1.4.1 Alcances:** El proyecto Implementación de Algoritmos Genéticos para la Resolución del Problema School Timetabling Para las Instituciones Educativas busca resolver el problema de la generación de horarios, haciendo esta tarea con un sistema capaz de organizar los horarios de clase en los colegios de manera automática, rápida y eficaz.

**1.1.4.2 Delimitaciones:** El proyecto Implementación de Algoritmos Genéticos para la Resolución del Problema School Timetabling Para las Instituciones Educativas está limitado inicialmente a la organización de horarios de clase en los colegios de Bogotá, para efectos de pruebas piloto se va a implementar la solución en el Colegio María Mercedes Carranza.

El estudio realizado abarca lo que corresponde a la programación de algoritmos genéticos y la resolución del problema School Timetabling.

Técnicamente, el proyecto Implementación de Algoritmos Genéticos para la Resolución del Problema School Timetabling Para las Instituciones Educativas va a ser trabajado en java usando el IDE Netbeans y PostgreSQL para el desarrollo de la base de datos.

**1.1.5 Objetivos**

**1.1.5.1 Objetivo General:** Desarrollar una herramienta computacional para resolver el problema del School Timetabling en la generación automatizada de horarios académicos para centros educativos mediante algoritmos genéticos.

**1.1.5.2 Objetivos Específicos:**

* Investigar acerca a la técnica algoritmos genéticos y el problema de optimización School Timetabling.
* Diseñar una interfaz de uso práctico y funcional para dar solución el problema de School Timetabling
* Realizar módulos para la captura, gestión de los datos y representación de restricciones duras y blandas para resolver el problema de School Timetabling.
* Implementar una prueba piloto de la herramienta en el colegio María Mercedes Carranza IED.
* Implementar una base de datos que permita relacionar las entidades y enlazarlas con el aplicativo.
* Diseñar un módulo que proporcione la generación y visualización de reportes personalizables actualizados.

**1.1.6 Justificación:** Al desarrollar una herramienta tecnológica que permita la generación automática de horarios de clase en los colegios, se van a reducir costos procedimentales y de personal, se va a reducir el tiempo de realización de las tareas y se busca optimizar los resultados finales para lograr satisfacer las necesidades tanto de docentes como de estudiantes, así como también aprovechar los espacios físicos con que se cuenta.

Otro punto a favor para la realización del School Timetabling es el salto tecnológico que se da, hoy en día existe la necesidad de avanzar junto con la tecnología, y aprovechar todas las herramientas que están a disposición de los usuarios para de esta manera convertir un trabajo tedioso y costoso, en algo simple, rápido y más eficaz.

Además, se está desarrollando un sistema que adrede pueda ser implantado en cualquier colegio de Bogotá, ya que la gran mayoría tiene el mismo sistema de organización horaria.

Finalmente, se está tratando el tema de algoritmos genéticos, tema que concierne a la investigación de la inteligencia artificial y que no se trabaja en gran medida durante el ciclo tecnológico de la Universidad Distrital Francisco José de Caldas en cuanto a proyectos de grado se refiere.

**1.1.7 Marco de Referencia**

**1.1.7.1 Marco Histórico**

**TIMETABLING Y METODOS DE RESOLUCION IMPLEMENTADOS**

A pesar de la gran dificultad de resolución del problema Timetabling clasificado como un problema NP-Completo dentro de la complejidad computacional, existen una variedad de métodos que han sido utilizados exitosamente obteniendo resultados correctos. Estos métodos o técnicas utilizadas se encuentran divididos en 2 grandes grupos (Figura 1):



**Fuente**: *Mejía & Arboleda [38*]

Figura 1: Técnicas de Búsqueda

**TÉCNICAS TRADICIONALES**

Son métodos denominados completos, que recorren todo el espacio de búsqueda y encuentran todas las posibles soluciones a determinado problema, sin embargo, el éxito de estos métodos depende directamente del número de variables que influyen en el problema. Dentro de este grupo se encuentran entre otros, la programación entera, programación lineal y backtracking.

**TÉCNICAS NO TRADICIONALES**

Estas técnicas son denominadas meta heurísticas, ya que en contraste con las técnicas tradicionales, no encuentran todas las soluciones posibles a un problema ya que acotan o reducen el espacio de búsqueda, diciendo así también que son métodos incompletos [15].

Dentro de este grupo están: Recocido Simulado (Simulated Annealing), Algoritmos Evolutivos (Evolutionary Algorithms), búsqueda tabú (Tabu Search), algoritmos voraces (GRASP), redes neuronales (Neuronal Networks), entre otras.

* **Templado o recocido Simulado (SimulatedAnnealing)**

El recocido simulado o” Simulated Annealing” en inglés, fue propuesto y estudiado en primera instancia por Metrópolis en el año de 1953 [23]

Es un método heurístico que tiene más relación con la termodinámica (similar al proceso de enfriamiento del metal), que con la optimización. Como es una variante de la búsqueda local, puede quedar atrapado prematuramente en un óptimo local [16]. En cada iteración una vecindad es generada (Un horario factible se modifica ligeramente de forma aleatoria para crear uno nuevo también factible). Este vecino es aceptado como el actual horario si se considera que tiene baja penalidad. Por el contrario, si este nuevo vecino presenta alta penalidad, se considera para ser aceptada como la actual solución, es decir, como un calendario (horario) acorde a una probabilidad relacionada con un parámetro de control denominado temperatura.

“Las soluciones obtenidas por estas estrategias descendentes, dependen fuertemente de las soluciones iniciales consideradas” [18]

A mayor temperatura, mayor probabilidad de aceptación de soluciones peores, de tal manera que el algoritmo acepta soluciones mucho peores al principio de la ejecución (exploración) pero no al final (explotación).

"Finalmente, cuando la temperatura es tan baja que ningún cambio se acepta, el algoritmo se detiene con el objeto inicial profundamente alterado, de hecho con el objeto que probablemente pueda encontrarse para la propiedad de interés. Esta situación inicial es el cero absoluto del objeto respecto a la propiedad" [17]

Gómez en [18] comenta que las principales desventajas se presentan por el tiempo computacional y la dificultad para ajustar adecuadamente los parámetros que controlan el algoritmo.

Aplicaciones en Timetabling se pueden ver en [24] y [25].

* **Búsqueda Tabú (Tabu Search)**

La meta heurística búsqueda Tabú es introducida y desarrollada por Fred Glover en 1986 [19]. Este método está diseñado para salir del óptimo local, "La filosofía de esta técnica es la creencia de que la elección de una mala estrategia sistemática de búsqueda es mejor que una buena elegida al azar" [4].

Su funcionamiento se ve expuesto en [26] de la siguiente manera: “Una búsqueda con lista tabú o Taboo Search consiste en partir de un candidato al azar (o generado con alguna otra heurística) y modificarlo progresivamente (mediante un segundo algoritmo) hasta que no sea posible obtener mejoras haciendo esa modificación”.

La principal característica de la búsqueda tabú, utiliza una memoria flexible por medio de estructuras simples, de tal manera que dirige la búsqueda de acuerdo a la historia que lleva, es así como el escape de óptimos locales se realiza de manera sistemática y no aleatoria.

"…desde el punto de vista de la Búsqueda Tabú, la memoria flexible envuelve el proceso dual de crear y explotar estructuras para tomar ventaja mediante la combinación de actividades de adquisición, evaluación y mejoramiento de la información de manera histórica…" Glover y Laguna [14]

Restrepo y Velásquez [20] expresan que la memoria es representada mediante una lista tabú, la cual contiene para la mejores soluciones o en su defecto, los movimientos realizados para obtener dicha solución, de esa forma no serán tenidos en cuenta en futuras iteraciones, lo que beneficia a tener un reducido número de soluciones elegibles.

Para el problema específico del Timetabling la búsqueda tabú existen implementaciones como en [27] y [28].

* **Colonia de Hormigas (Ant Colony)**

Una de las meta-heurísticas más empleadas recientemente para enfrentar problemas de optimización, desde su inicio por Dorigo, Maniezzo y Colorni en la primera mitad de la década de los 90’ [21].

Cada hormiga en la colonia realiza inicialmente trayectorias aleatorias en búsqueda de su alimento, al hallarlo estudia la cantidad y la calidad según Gómez [18] y regresa a su colonia depositando una feromona, que permitirá a otras hormigas seguir el rastro reforzando la intensidad de la feromona y evitando su evaporación, de manera análoga con los problemas de optimización el concepto de evaporación de la feromona es utilizado para evitar que el algoritmo converja a un óptimo local. [20] En contraste si no existiese la evaporación de la feromona, cualquier trayectoria sería igual de atractiva para las hormigas lo que se traduciría en una exploración muy amplia de soluciones.

En general, lo que se pretendió y se pretende aun con el método de la colonia de hormigas se expone en [18] de la siguiente manera:

“La idea primordial al poner en práctica la Metodología de Colonia de Hormigas es intentar obtener una alta organización y distribución entre las hormigas artificiales para utilizarlas en la administración de la población de agentes artificiales, obteniendo como resultado las mejora en la solución de problemas de optimización combinatoria.”

Algunas aplicaciones en Timetabling [18] y [29].

* **GRASP (Greedy Randomize Adaptive Search Procedure)**

Esta surgió en 1989 gracias a Feo y Resende, y según su autor fue desarrollada para resolver problemas difíciles en el campo de la optimización combinatoria. [30]

Esta metodología se desarrolla mediante un proceso iterativo, dividido en dos fases principales, la fase de construcción y la fase de mejoramiento, explicado en [31] de la siguiente manera:

En la fase de construcción, entra la función greedy o miope, que determina el añadido de un elemento a una solución parcial, es si, la función miope consiste en elegir el mejor camino o la mejor opción para un elemento, y luego de que el elemento es añadido a la solución parcial, se re calculan los valores de la función, lo cual hace que este procedimiento sea adaptativo.

Pero en realidad no se garantiza una solución óptima y es acá donde entra a trabajar la fase de mejoramiento, es por ello que en la fase anterior se habla de una solución parcial y no final.

En esta segunda fase lo que se realiza, es un procedimiento de búsqueda, que a partir de la solución parcial dada, busca una solución mejor.

Podemos encontrar algunas implementaciones para el problema de Timetabling aplicando esta metodología en [27] y [32].

* **Algoritmos Genéticos (Genetics Algorithms)**

Cabezas [4] hace referencia a Díaz [22] para presentar este método meta-heurístico:

Definición (Algoritmo Genético, según Díaz [22]). Un algoritmo Genético es una estructura de control que organiza o dirige un conjunto de transformaciones y operaciones diseñadas para simular los procesos de evolución.

Estos algoritmos están inspirados en la teoría de evolución de Darwin en1859 (evolución por selección natural), donde los individuos con más aptitudes para sobrevivir y dejar un mayor número de descendientes, son los más favorecidos (aptos) y transmiten a sus hijos los caracteres favorables de manera hereditaria.

El funcionamiento de un AG (sigla que se utilizara para referirse a Algoritmo Genético), parte de crear un cromosoma o cadena de información, conocida como genotipo, la cual establece la relación entre un conjunto de soluciones de un problema (fenotipo) y el conjunto de individuos de una población inicial.

Varios individuos se agrupan formado una población, aquellos que mejor se adapten son los que tienen mayor probabilidad sobrevivir y reproducirse

Los nuevos cromosomas se formaran seleccionando algunos individuos, utilizando operadores genéticos de cruzamiento y mutación y serán evaluados en cada nueva iteración (generación) mediante una medida de aptitud, originándose así, una nueva descendencia.

**CUADRO COMPARATIVO DE TÉCNICAS METAHEURISTICAS**

Del trabajo investigativo realizado con cada una de las metaheurísticas, se obtuvo el siguiente cuadro, en el cual se describen las características de cada método:

**Tabla 1: Características de las técnicas meta-heurísticas estudiadas.**

***Fuente:*** *Trabajo investigativo, elaboración propia*

|  |  |
| --- | --- |
| META-HEURÍSTICA | CARACTERÍSTICAS |
| GRASP | * 1989 * Adaptativa de acuerdo a las condiciones del problema * Metodología de las más recientes * Requiere alto tiempo para hallar solución * Búsqueda aleatoria * Dificultad de adecuar los parámetros * **Soluciones que maneja: 1** |
| BÚSQUEDA TABÚ | * 1986 * Requiere solución inicial * Más complejo de implementar * Buenos resultados en poco tiempo * No garantiza optimalidad * **Soluciones que maneja: 1** |
| RECOCIDO SIMULADO | * 1983 * Facilidad de implementación * Complejo para problemas muy grandes * No garantiza optimalidad * Facilidad para combinar con otras técnicas heurísticas, para obtener sistemas híbridos * Dependiendo de los parámetros elegidos, las soluciones que se van encontrando pueden ser poco estables. * **Soluciones que maneja: 1** |
| COLONIA DE HORMIGAS | * 1996 * La más novedosa * Tiempo para encontrar solución de calidad es alto * Ofrece buenas soluciones * No garantiza optimalidad * **Soluciones que maneja: N, determinadas por la naturaleza del problema.** |
| ALGORITMOS GENÉTICOS | * 1975 * Método más antiguo de los estudiados en este articulo * Implementación relativamente simple * No necesitan conocimientos específicos sobre el problema a resolver. * Altamente estudiados * Utilizan operadores probabilísticos (otras técnicas utilizan determinanticos), sin embargo recorren el espacio de soluciones en forma "más inteligentes" que la búsqueda aleatoria. * Mucha información y soporte * Útiles en casos donde no es necesario obtener una solución óptima al problema, sino que una buena solución aproximada sería suficiente. * Utilizan una población de soluciones, siendo menos sensibles a quedar atrapadas en óptimos locales que las técnicas que utilizan una solución única. * **Soluciones que maneja: N, determinadas por la naturaleza del problema.** |

En [33] se realizó un estudio comparativo entre las técnicas meta-heurísticas de Recocido Simulado, Búsqueda Tabú, Algoritmos Genéticos y Algoritmos Meméticos, teniendo en cuenta una serie de características como simplicidad, efectividad, adaptabilidad, autonomía, ente otras, a las cuales se les dio un puntaje entre 0 y 1, siendo 0 el puntaje más bajo y 1 el puntaje más alto. De esta comparación se dedujo que el uso de Algoritmos Genéticos es el más óptimo en comparación con los otros ubicados en este estudio.

En la tabla 2 se muestra de manera resumida el estudio elaborado en [33].

**Tabla 2: Calificación de metaheurísticas**

***Fuente:*** *Naupari y Rosales* **[33]**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| CARACT | ALG. MEMÉT | ALG GEN | BÚSQ TABÚ | REC SIMUL |
| Simplicidad | 0.5 | 1 | 0.75 | 0.75 |
| Independencia | 1 | 1 | 1 | 1 |
| Coherencia | 0.75 | 1 | 0.75 | 1 |
| Efectividad | 1 | 1 | 1 | 0.75 |
| Eficacia | 0.75 | 0.75 | 0.75 | 0.5 |
| Eficiencia | 1 | 1 | 0.75 | 0.5 |
| Generalidad | 0.5 | 0.75 | 1 | 1 |
| Adaptabilidad | 1 | 1 | 1 | 0.75 |
| Robustez | 0.75 | 0.75 | 0.75 | 0.75 |
| Interactividad | 0.75 | 0.75 | 0.75 | 0.75 |
| Diversidad | 1 | 1 | 0.5 | 0.5 |
| Autonomía | 1 | 1 | 1 | 1 |
| Puntaje Final | 10 | 11 | 10 | 9.25 |

**CASO DE ESTUDIO**

**BREVE HISTORIA DE LOS ALGORITMOS GENÉTICOS**

La formulación y creación de los algoritmos genéticos es atribuida principalmente a John Holland quien trabajo en ello durante las décadas de 1960 y 1970. Sin embargo existen autores que fueron parte importante durante el estudio y desarrollo de los algoritmos genéticos, entre los cuales podemos encontrar a:

Bagley (1967), que según [39], diseño algoritmos genéticos para buscar conjuntos de parámetros en funciones de evaluación de juegos, los comparo con los algoritmos de correlación.

Igno Rochenberg, en uno de los aportes más importantes en el tema, introdujo una técnica llamada estrategia evolutiva, dicha técnica no tenía población ni cruzamiento, simplemente un padre mutaba para producir u descendiente y se conservaba el mejor de ellos [40].

En 1996, L.J.Fogel, A.J. Owens y M.J. Walsh introdujeron en América una técnica que llamaron programación evolutiva, con este método las soluciones candidatas para los problemas con representadas como máquinas de estado finito sencillas y que al igual que la estrategia de Rochenberg, el algoritmo muta aleatoriamente alguna de las maquinas simuladas y se conserva el mejor de los 2 [40].

Sin embargo, como se mencionó anteriormente, fue Holland, quien con sus estudios fue el primero en proponer explícitamente el cruzamiento y otros operadores de recombinación [40]. En contraste con las estrategias evolutivas (Rochenberg) y la programación evolutiva (Fogel), el propósito de Holland era estudiar de manera formal el fenómeno de la adaptación tal cual ocurre en la naturaleza, y de esta manera aplicarlo a sistemas computacionales.

Todas las teorías de Holand fueron plasmadas en su libro “Adaptation in natural and artificial systems'' (1975). La mayor innovación, fue introducir un algoritmo basado en poblaciones con cruces, mutaciones e inversiones, simulando así el proceso de la evolución biológica para la resolución de problemas computacionales [40].

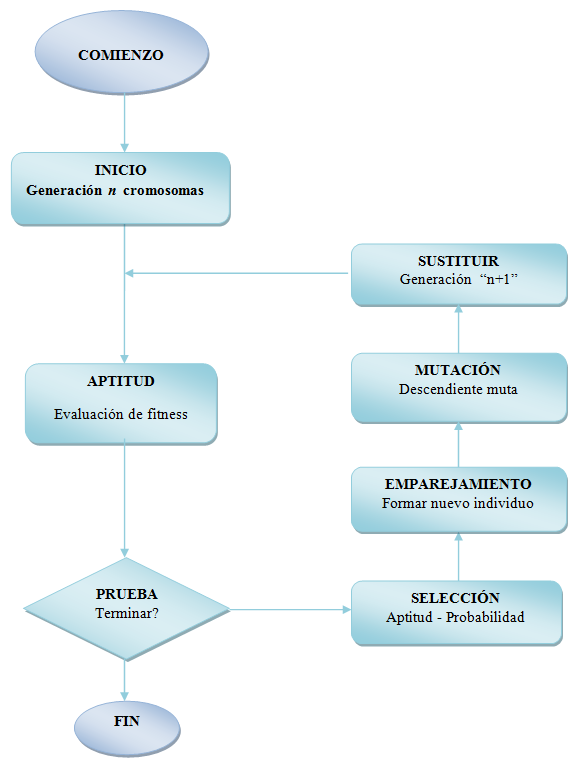
Posteriormente, y basados principalmente en las teorías de Holland, los estudios se fueron ampliando de manera teórico-práctica, y de allí se empezó a vislumbrar el enorme potencial de los algoritmos genéticos para la resolución de problemas de optimización [41].

En la actualidad, campos como la ingeniería, la investigación operativa y la programación automática, entre otros, desarrollan aplicaciones basadas en algoritmos genéticos y siguen surgiendo avances y estudios por parte de los grupos de investigación mundiales que se reúnen desde 1985, cuando se organizó el primer “Adaptation in natural and artificial systems'', en donde se presentaron los últimos avances teóricos y prácticos en el empleo de estas técnicas. Las actividades y estudios en el campo de los algoritmos genéticos creció en tal proporción que se fundó la “International Society for Genetic Algorithms (ISGA) '', entre otras asociaciones y congresos internacionales [41].

**1.1.7.2 Marco Teórico**

**ALGORITMOS GENÉTICOS**

En [33], describen el proceso básico de un AG, la codificación del cromosoma, operadores genéticos y parámetros de configuración. El funcionamiento del mismo se muestra en la figura 2:



Fuente: Naupari y Rosales [33]

Figura 2: Funcionamiento de un algoritmo genético

1. [Inicio] se genera la población aleatoria de n cromosomas (soluciones posibles para el problema).
2. [Aptitud] se evalúa la aptitud f (x) de cada cromosoma x de la población.
3. [Prueba] si la condición de término está satisfecha, se para el algoritmo, se devuelve la mejor solución de la población actual y se va al paso 7.

IV. [Nueva población] se crea una nueva población repitiendo los siguientes pasos, hasta que se cumpla la condición de parada.

a. [Selección] se selecciona dos cromosomas padres, de una población, según su aptitud (cuanto mejor es la aptitud, mayor es la probabilidad de ser seleccionado).

b. [Emparejamiento] con una probabilidad de emparejamiento, los padres se emparejan para formar a un nuevo descendiente (hijos). Si no se realiza emparejamiento alguno, el descendiente es la copia exacta de los padres.

c. [Mutación] con una probabilidad de mutación, el nuevo descendiente muta (en alguna posición de su cromosoma).

V. [Sustituir] la nueva población generada es aplicada para otra iteración del algoritmo.

VI. [Bucle] se va al paso 2.

VII. Fin del algoritmo.

**Codificación del genotipo**

Como se ha señalado los AG están compuestos de una población, que contiene diversos individuos (cromosomas), estos están conformados por un número determinado de genes.



**Fuente:** *Naupari y Rosales [33]*

Figura 3: Individuo Binario de un AG

En la Figura 3, se observa que un gen contiene una información valiosa para solución de problema, esta información debe manejar una codificación, generalmente es binaria, sin embargo dependiendo de la naturaleza del problema se determina cual se implementará.

Naupari y Rosales [33], dan a conocer dos (2) tipos de codificación utilizadas en los algoritmos genéticos:

* Codificación Indirecta: en este tipo de codificación se halla la binaria, representada por cadenas de “1” y “0”; es la más común, por ventajas de cómputo y de programación.
* Codificación Directa: en contraste con la anterior, aquí se sitúan las cadenas de números reales, principalmente genes compuestos por números enteros o decimales.

En la codificación directa, también se tienen a las cadenas de letras alfabéticas.

**Operadores genéticos**

Consisten en los métodos que se pueden ejecutar sobre una población, son 4:

* Selección: Proceso que escoge los miembros de la población que serán utilizados en la reproducción (padres). Se eligen los más aptos.

Existen varios métodos de selección la más conocida es la rueda de ruleta (Roulette Wheel); en [32, 36] dan a conocer algunos como: elitista, por Estado Estacionario, por torneo, escalada, entre otras.

* Reproducción, Emparejamiento o Cruce (Crossover): Consiste de alguna manera los cromosomas de dos padres, para formar dos descendientes (hijos).

Algunas variaciones son: cruce de n puntos, uniforme, segmentada, aritmético, etc. [32,36]

* Mutación: es encargada de modificar uno o más genes del descendiente, para buscar un factor de diversificación. Se realiza de manera aleatoria, siguiendo a la probabilidad de mutación establecida. Según [33] existen algunas técnicas, algunas como: mutación de bit, de gen, multibit, multigen, de intercambio.
* Reemplazo o Sustitución: es el método por el cual se insertan los hijos en la población; por ejemplo, mediante la eliminación del individuo más débil o al azar. [33]

**Parámetros de un AG**

Estos parámetros deben ser establecidos para cada ejecución que se haga, según [33, 37] definen:

**Tamaño de la Población**

Este parámetro es muy importante determinarlo, dependiendo del problema y las variables a manejar, estamos tratando de la cantidad de individuos que conformara la población en el trascurso de la ejecución. De tal manera, que para un número insuficiente de cromosomas, el AG tiene pocas posibilidades de diversidad, lo que afecta la reproducción y realizará una búsqueda escaza y poco óptima. Por otro lado, si la población es excesiva, el algoritmo genético será excesivamente lento. [37]

**Probabilidad o porcentaje de cruce (crossover**):

Determina con qué frecuencia se cruzan los individuos; si es 0% los hijos serán como los padres y solo se afectaran por la mutación. Si este es 100% todos los nuevos individuos son creados mediante reproducción de los padres de la generación previa.

Cuanto más se emparejen los individuos, se supone que los hijos serán mejores; sin embargo, se recomienda por la naturaleza de un AG, que algunos individuos pasen a la siguiente generación sin modificarse. [33]

**Probabilidad o porcentaje de mutación:**

Indica la probabilidad en qué deben ser mutados los individuos; si es 0% los descendientes son los mismos que había tras la reproducción. En caso de que haya mutaciones, parte del cromosoma descendiente se modifica; si es de 100%, la totalidad del cromosoma se cambia. [37]

La mutación trata de impedir que la búsqueda caiga en óptimos locales, por eso es conveniente que ocurra de vez en cuando; por el contrario, si ocurre continuamente, se convierte en una búsqueda aleatoria. [33]

**METODOLOGIA DE DESARROLLO RUP (Rational Unified Process)**

El Proceso Unificado de Rational es un proceso de ingeniería del software. Según describe Kruchten en [44] RUP proporciona un acercamiento disciplinado a la asignación de tareas y responsabilidades en una organización de desarrollo. Su propósito es asegurar la producción de software de alta calidad que se ajuste a las necesidades de sus usuarios finales con unos costos y calendario predecibles.

En [45] se describen las 4 fases mediante las cuales se realiza el proceso de la metodología RUP de la siguiente manera:

* Inicio: Esta fase debe establecer el ámbito del proyecto y sus límites, encontrar los casos de uso críticos del sistema, estimar el coste en recursos y tiempo de todo el proyecto, realizar los modelos de casos de uso y planificar los objetivos del proyecto.
* Elaboración: Se define y valida la arquitectura del software, se define en concreto cual es la visión general del proyecto.
* Construcción: Se inicia el desarrollo practico del proyecto, aquí se realiza el proceso de diseño, codificación y pruebas continuas.
* Transición: Se pone el producto en manos del usuario final, se completa la documentación, se entrena al usuario en el manejo del producto y en general se realizan tareas de ajuste, configuración, instalación y usabilidad.

De la misma forma, RUP define 9 flujos de trabajo que son realizados a lo largo del todo el proceso en caso de ser necesario:

* Modelado de Negocio
* Requisitos
* Análisis y Diseño
* Implementación
* Test
* Despliegue
* Administración de Proyecto
* Configuración de Control y Cambios
* Entorno



Fuente: Martinez, Raul & Martinez, Alejandro [45]

Figura 4: Fases Metodología RUP

**1.1.7.3 Marco Conceptual**

**TIMETABLING**

Conocido en la literatura de habla hispana, como un problema de Programación horaria o Calendarización; en él existen recursos que deben ser asignados, en instantes o bloques de tiempo determinados, teniendo en cuenta requisitos y condiciones (“restricciones”). Las siguientes son algunas de las definiciones más claras del término:

**Definición 1:** Zhipeng Lu y Jin-Kao Hao, definen Timetabling como: “Asignar un número de eventos, cada uno con ciertas características, a un número limitado de recursos sujeto a restricciones” [1]. Anterior a ellos Anthony Wren en 1996, determina el Timetabling, como un caso especial de Programación (scheduling), a esta la define como:

**Definición 2:** “la asignación, sujeta a restricciones, de un grupo de recursos a objetos ubicados en tiempo y espacio, de tal manera que se satisfagan un conjunto de objetivos deseados” [2]

**Definición 3:** Lance D. Chambers [42] toma la definición de una forma más general: “El Timetabling (Calendarización) se puede definir como aquello que describe, donde y cuando las personas y los recursos deben estar en un instante dado”

El problema de calendarización se puede apreciar en diferentes escenarios, motivo por el cual gran cantidad y variedad de grupos de investigación toman este caso de estudio para optimizar los resultados y lograr soluciones de calidad. En general existen 4 tipos principales de Timetabling determinados por PATAT y su coorganizador (grupo de investigación ASAP [3]), estos son:

**Transport Timetabling:** aquí se encuentran las diferentes problemáticas en la asignación de rutas de los conductores de buses de transporte público o privado, trenes (por ej., [6, 12]) y/o aviones (conocido como Airline Schedule en [7]); además con la programación y previsibilidad de horarios de salida y la optimización de operaciones de llegada principalmente en aeropuertos, donde los controladores de las pistas de aterrizaje de aeropuertos modernos, se realizan de manera manual y requieren de un alto grado de precisión en poco instantes de tiempo.

**Sports Timetabling:** (también conocido como Sports scheduling, en [3]). En esta interesante variante del problema mencionado, los deportes en especial el futbol, cuenta con una característica como son los diferentes tipos de enfrentamientos entre equipos, ya sea de uno contra uno, ida y vuelta, o torneos de todos contra todos; los cuales requieren un tipo de programación de encuentros distinta, otra inconveniente por ejemplo, en épocas de Navidad y Año nuevo se realizan encuentros muy seguidos, y los simpatizantes deben desplazarse por largas distancias, lo cual requiere otro tipo de optimización que beneficie al espectáculo, taquillas, transmisión por cable, etc. Un ejemplo se puede ver en [5].

**Employee Timetabling and Rostering:** primordialmente esta modalidad trata de los turnos particularmente en el sector de la salud (Nurse Rostering en [9]), puesto que, enfermeras y médicos deben cumplir diferentes turnos de trabajo, se deben equilibrar las cargas de trabajo, teniendo en cuenta restricciones duras y blandas, tales como, exigencia mínima de enfermeras, días de descanso, etc. [9]. También algunas empresas, imparten horarios de trabajo a sus empleados, por lo tanto, en turnos y/o horarios, se deben tener en cuenta distintas restricciones, en [10] encontramos algunas como: límite máximo de horas o turnos, existencia de interrupciones, existencia de trabajadores temporales y de tiempo completo, periodos de planificación, disponibilidad y preferencias del trabajador, entre otras. (Véase un ejemplo de un Call Center en [8]).

**Educational Timetabling:** en el ámbito educativo, los principales problemas son los de programación de horarios tanto en colegios (School Timetabling) como en universidades (University or Course Timetabling), este tipo de programación de materias es subdividida por Cabezas J. [4] en “EB-CTT”3 y “CB-CTT”4 ; todos los anteriores requieren una eficiente asignación de recursos respetando instantes de tiempo establecidos, esto implica una serie de restricciones y preferencias derivadas de personas, instituciones, reglamentos u otras. Este tipo de programación tiene una gran complejidad, por la cantidad de variables y limitaciones, de tal manera que existe un espacio para la investigación y el desarrollo [3]. Otro campo subsecuente de la educación tiene que ver con la carga de exámenes y su calendarización (Exam o Examination Timetabling).

Dentro del campo Educational Timetabling se encuentran 2 tipos de variaciones generales, como lo son la asignación de horarios escolares y la asignación de horarios universitarios los cuales difieren en ciertos aspectos que en [43] son definidos de la siguiente manera:

Asignación de Horarios Escolares (School Timetabling): “También conocido como Class-Teacher Problem, considera el horario semanal para las sesiones de las asignaturas de una escuela o colegio. Dadas las asignaturas, profesores, bloques y una matriz de requerimientos (que establece el número de sesiones que cada profesor dicta por asignatura), el problema consiste en asignar las sesiones a los períodos de tiempo, de tal manera que ningún profesor o asignatura tenga más de una sesión en el mismo período y que todas las sesiones de la asignatura estén presentes en el horario.”

Asignación de Horarios Universitarios (University Timetabling): “Este problema consiste en organizar un horario para las sesiones de un conjunto de asignaturas, considerando un número determinado de salas y bloques de tiempo.”

Diferencias: La gran diferencia entre un horario escolar y uno universitario es la forma en la que son considerados los estudiantes. En un horario escolar pueden considerarse una entidad, debido a que es un grupo de alumnos que toman las mismas asignaturas. En el caso universitario, los estudiantes toman distintas asignaturas, por lo que se generan asignaturas en común con otros estudiantes.

En segunda instancia están los profesores, ya que en las escuelas estos enseñan una asignatura y en la universidad generalmente imparten entre 1 y 3 asignaturas cada uno.

Existen algunos otros parámetros que difieren entre el School Timetabling y el University Timetabling que son resumidos en siguiente tabla: [43]

|  |  |  |
| --- | --- | --- |
| **Características** | **Escolar** | **Universitario** |
| **Programación** | Pocas elecciones  Mallas bien estructuradas | Muchas elecciones  Mallas débilmente estructuradas |
| **Disponibilidad Profesor** | Ajustado  (Poseen gran carga) | Flexible  (Posee carga liviana) |
| **Salas** | Pocas salas  Mismo tamaño Centralizadas | Muchas Salas  Variedad de tamaños Descentralizadas |
| **Carga Estudiantes** | Muy saturado  Una sola jornada | Medianamente holgado Utiliza mañanas y tardes |
| **Criterio de Optimización** | Satisfacción de restricciones | Minimización de restricciones Transgredidas |

**Fuente:** *Franco, Toro, Mirledy y Gallego [43]*

**Tabla 3: Comparación entre la asignación de horarios escolares y universitarios**

El problema de Educational Timetabling, por lo general maneja 2 grandes tipos de restricciones para su desarrollo, las Restricciones Duras y las Restricciones Blandas. Larrosa [11] las describe de la siguiente manera:

**Restricciones Duras (Obligatorias):** son condiciones de obligatorio cumplimiento, de tal manera que la violación a alguna origina un horario no valido. Son espaciales (p. ej.: la cantidad de estudiantes no debe superar la capacidad de un aula) o temporales (p. ej.: un docente no debe tener asignado dos o más cursos en un mismo bloque de tiempo), de esta manera se dice que toda restricción dura se debe satisfacer.

**Restricciones Blandas (Deseadas):** son restricciones que denotan preferencias del usuario, se busca que se cumplan en la medida de lo posible (p. ej.: no se desea que un profesor se traslade a diferentes aulas cuando se tiene una clase de dos periodos consecutivos). La violación de alguna de estas seguirá ocasionando un horario factible, pero no de la calidad deseada.

**COMPLEJIDAD COMPUTACIONAL**

En la búsqueda de soluciones a una gran cantidad de problemas en el área de la computación, se notó que hay algunos más difíciles de resolver que otros, teniendo en cuenta principalmente el tiempo de procesamiento y la cantidad de espacio en memoria que se requiere para resolver el problema, sabiendo esto, la complejidad del problema se puede clasificar en 3 tipos principales: P, NP y NP-COMPLETOS. [33]

**Problemas P:** Los problemas de tipo P son aquellos que son solucionables en tiempo polinomial, es decir problemas sencillos que se pueden resolver fácilmente de forma práctica tales como multiplicaciones, funciones lineales, cuadráticas, etc. Todo problema que se encuentra en P hace parte de los problemas situados en NP [33].

**Problemas NP:** Los problemas de tipo NP tienen un concepto similar al de los problemas P, ya que son resueltos en un tiempo polinomial, la diferencia es que son problemas NO DETERMINISTICOS, es decir, suelen ser resueltos mediante el uso de una máquina de turing no determinista, con esto estamos diciendo que no sabemos cuál es el resultado que se va a dar, y el tiempo de procesamiento depende de la cantidad de datos de entrada.

Este tipo de problema contiene los problemas que también son contenidos dentro de las otras clases (P, NP-c).

Se dice que contiene los problemas P, porque es posible la aplicación de un algoritmo polinomio que compruebe que la solución dada es válida o no, entonces, en P los problemas se resuelven en tiempo polinómico y en NP los problemas se comprueban en tiempo polinómico.

Principalmente esta clase abarca problemas de búsqueda y optimización como la utilización de grafos [33].

**Problemas NP-Completos:** Los problemas NP-COMPLETOS, son también problemas NP, es decir, los problemas NP pueden ser reducidos a problemas NP-COMPLETOS, y el tiempo computacional requerido aumenta exponencialmente con el tamaño que tenga el problema [34].

Como se dijo antes NP abarca el conjunto completo de problemas (figura 5), entonces se puede decir que los problemas NP-COMPLETOS son los más difíciles de resolver dentro del conjunto NP, y no están presentes dentro de los problemas P.

Los problemas NP-COMPLETOS podrían parecer tan complejos que algunos dirían que son intratables, pero en realidad no se ha podido comprobar esto.

También se puede decir que este tipo de problemas son equivalentes entre sí. Si existe una solución para un problema NP-COMPLETO, entonces existe para cualquier problema de este tipo, y si por el contrario se comprobara que un problema NP-COMPLETO no tiene solución, entonces ninguno la tendría. [33]



**Fuente:** *Elaboración propia.*

Figura 5: Complejidad Computacional

Para el caso específico de nuestro problema de estudio (Timetabling), la gran mayoría de autores coinciden que está ubicado en la clase de problemas NP-COMPLETOS [32] [34] [35], por su gran dificultad de resolución.

**1.1.8 Factibilidad**

**1.1.8.1 Técnica:** Se cuenta con los siguientes equipos de cómputo para el desarrollo del proyecto:

|  |  |  |
| --- | --- | --- |
| **PARTE** | **PRINCIPAL** | **SECUNDARIO** |
| **MEMORIA RAM** | 2 GB – DDR3 | 3GB |
| **DISCO DURO** | 500 GB | 500 GB |
| **PROCESADOR** | INTEL CORE 2 DUO E8400 – 3.0 GHz | INTEL CORE i3 - 2.2 GHz |
| **MONITOR** | ACER V173 – 17” | HP |
| **TECLADO** | GENIUS | HP |
| **ESTABILIZADOR** | GAMMAX R: 1000 VA |  |
| **MOUSE** | GENIUS |  |
| **TARJETA DE VIDEO** | NVIDIA GeForce 9500 GT 512 MB – DDR3 | INTEL HD Graphics 3000 - 1.5 GB |

**Tabla 4: Factibilidad Técnica**

Teniendo en cuenta los datos anteriores, se puede concluir que se cuenta con los equipos necesarios para el desarrollo del proyecto.

**1.1.8.2 Operativa:** El proyecto es conformado por 2 estudiantes de Tecnología en Sistematización de Datos de la Universidad Distrital FJC, con los conocimientos necesarios para el manejo de herramientas computacionales de programación y capacitados para la investigación e interpretación de las temáticas del proyecto (School Timetabling, Algoritmos Genéticos).

Además, el proyecto cuenta con el apoyo de un tutor con bastantes conocimientos acerca de algoritmos genéticos y que forma parte del grupo de investigación de la UDFJC.

**1.1.8.3 Legal:** Las herramientas principales a utilizar para el desarrollo del proyecto como lo son el IDE Netbeans y PostgreSQL son totalmente gratis, no asi el Windows OS el cual es facilitado por el grupo de investigación de Inteligencia Artificial de la Universidad Distrital.

Herramientas secundarias como lo StarUML para el modelado de los diagramas o para la creación del modelo relacional de la base de datos también son totalmente gratis.

**1.1.9 Cronograma de actividades:**

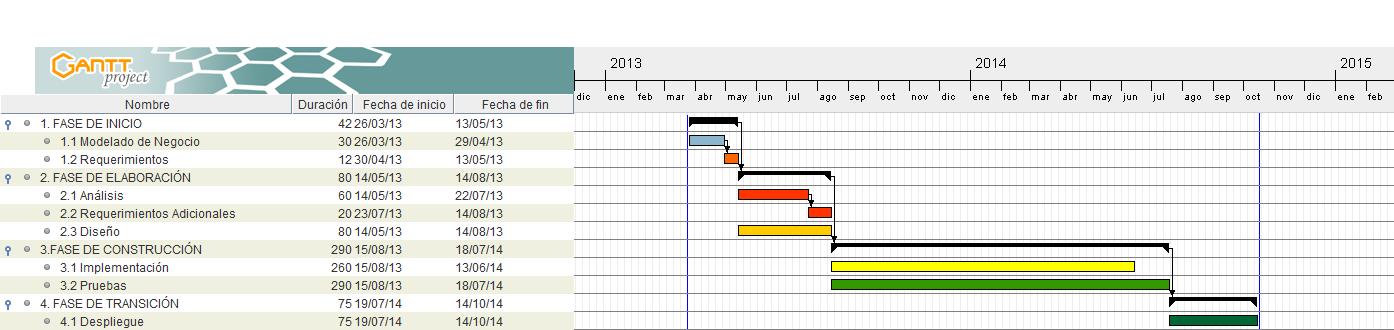


Figura 6: Cronograma

**1.2 Fase de análisis**

**1.2.1 Análisis de Requerimientos del sistema y del software.**

**1.2.1.1 Requerimientos Humanos**

La realización del proyecto de grado requiere personas con conocimientos de programación en el lenguaje java y la gestión de bases de datos SQL, así como también un tutor el cual guiará a los desarrolladores y ayudará conceptualmente sobre los temas tratados en el proyecto, School Timetabling y Algoritmos Genéticos en este caso.

**Planeación, creación y desarrollo del proyecto**

Mauricio Andes Guerra Cubillos

Erwin Hamid Pardo Quiroga

**Tutor**

Roberto Emilio Salas Ruiz

**1.2.1.2 Requerimientos de Hardware**

**1.2.1.3 Requerimientos de Software**

|  |  |
| --- | --- |
| **HERRAMIENTA** | **VERSION** |
| Netbeans | 7.2.1 o superior |
| PostgreSQL | 9.2 o superior |
| Windows OS | XP, Vista, 7, 8. |
| Microsoft Office | 2010 o superior |
| StarUML | 5.0 o superior |

**1.2.2 Diagrama de funciones:**

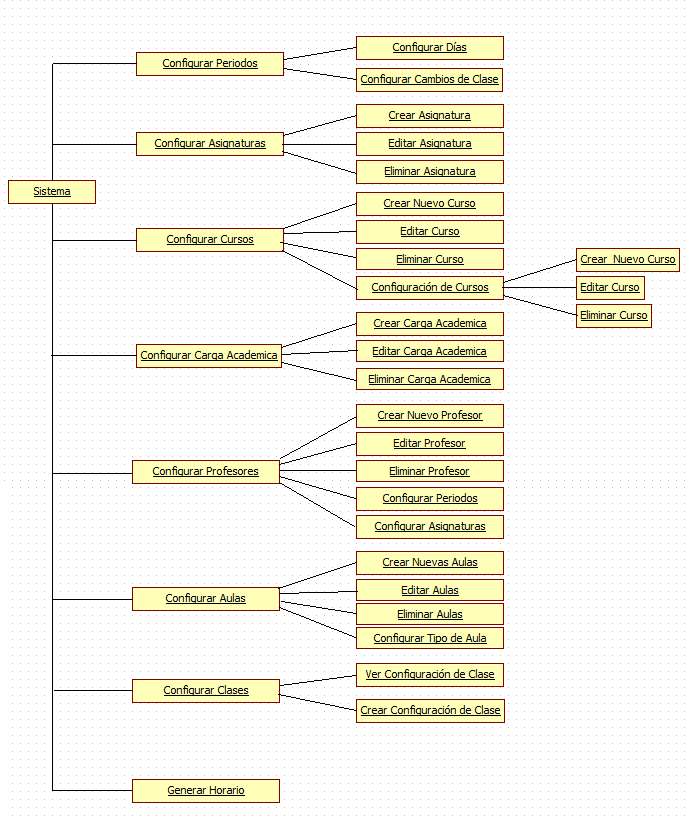


Figura 7: Diagrama de Funciones

**1.3 Fase de diseño**

**1.3.1 Diseño Entidad Relación**



Figura 8: Diagrama Entidad-Relación

**1.3.2 Diagramas de caso de uso**

**Caso de Uso No 1: Recolectar Información de Docentes**

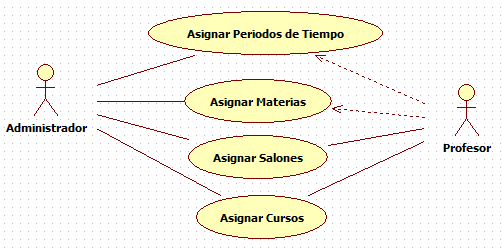


Figura 9: Recolectar Información

**Caso de Uso No. 2: Configurar Días**

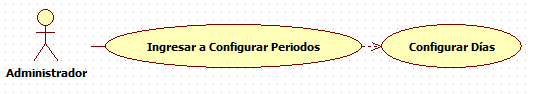


Figura 10: Caso de Uso – Configurar Días

**Caso de Uso No. 3: Configurar Clases**

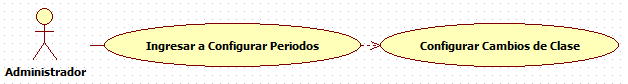


Figura 11: Caso de uso – Configurar Clases

**Caso de Uso No. 4: Crear Nueva Asignatura**

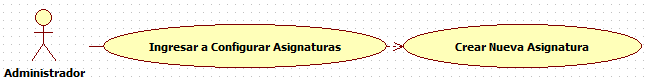
****

Figura 12: Caso de uso – Crear Asignatura

**Caso de Uso No. 5: Editar Asignatura**

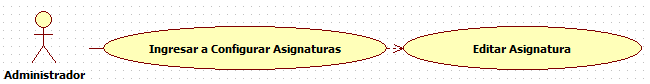
****

Figura 13: Caso de uso – Editar Asignatura

**Caso de Uso No. 6: Eliminar Asignatura**

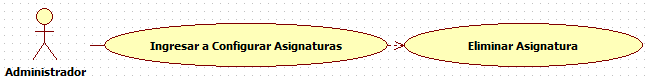


Figura 14: Caso de uso – Eliminar Asignatura

**Caso de Uso No. 7: Crear Nuevo Grado**

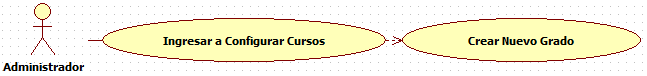
****

Figura 15: Caso de uso – Crear Grado

**Caso de Uso No. 8: Eliminar Grado**

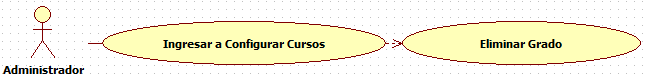
****

Figura 16: Caso de uso – Eliminar Grado

**Caso de Uso No. 9: Crear Nuevo Curso**

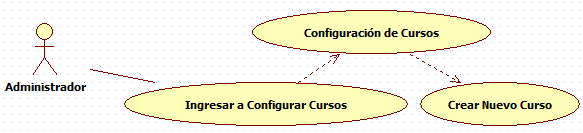
****

Figura 17: Caso de uso – Crear Curso

**Caso de Uso No. 10: Editar Curso**

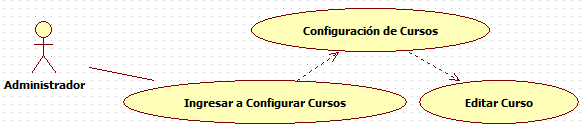
****

Figura 18: Caso de uso – Editar Curso

**Caso de Uso No. 11: Eliminar Curso**

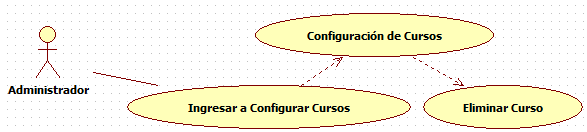
****

Figura 19: Caso de uso– Eliminar Curso

**Caso de Uso No. 12: Crear Nueva Carga Académica**

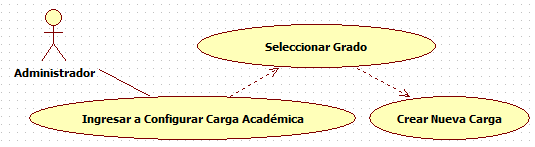
****

Figura 20: Caso de uso– Crear Carga Académica

**Caso de Uso No. 13: Editar Carga Académica**

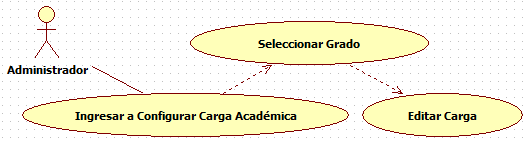
****

Figura 21: Caso de uso– Editar Carga Académica

**Caso de Uso No. 14: Eliminar Carga Académica**

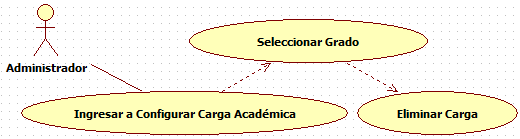
****

Figura 22: Caso de uso– Eliminar Carga Académica

**Caso de Uso No. 15: Crear Nuevo Profesor**

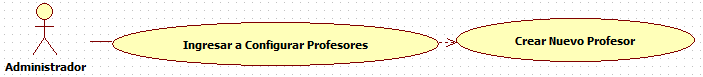
****

Figura 23: Caso de uso– Crear Profesor

**Caso de Uso No. 16: Editar Profesor**

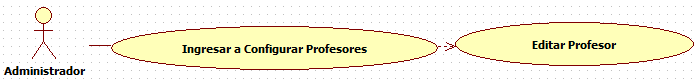
****

Figura 24: Caso de uso– Editar Profesor

**Caso de Uso No. 17: Eliminar Profesor**

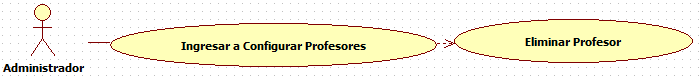
****

Figura 25: Caso de uso– Eliminar Profesor

**Caso de Uso No. 18: Configurar Periodos Profesor**

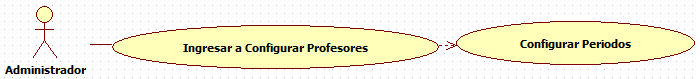
****

Figura 26: Caso de uso– Configurar Periodos

**Caso de Uso No. 19: Configurar Asignaturas Profesor**

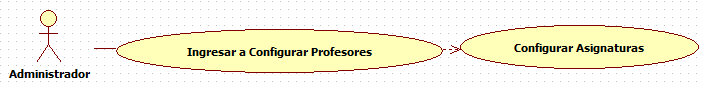
****

Figura 27: Caso de uso– Configurar Asignaturas

**Caso de Uso No. 20: Crear Nueva Aula**

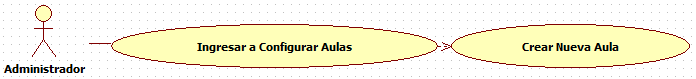
****

Figura 28: Caso de uso– Crear Aula

**Caso de Uso No. 21: Editar Aula**

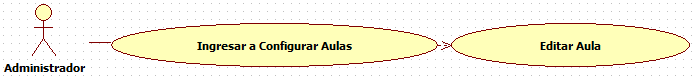
****

Figura 29: Caso de uso– Editar Aula

**Caso de Uso No. 22: Eliminar Aula**

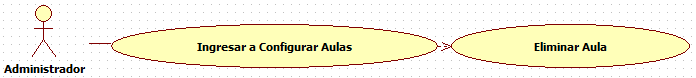
****

Figura 30: Caso de uso– Eliminar Aula

**Caso de Uso No. 23: Crear Nueva Configuración de Clases**

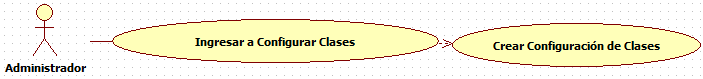
****

Figura 31: Caso de uso– Crear Configuración de Clases

**Caso de Uso No. 24: Editar Configuración de Clases**

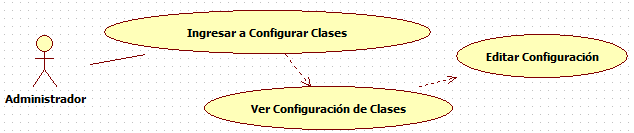
****

Figura 32: Caso de uso– Editar Configuración de Clases

**Caso de Uso No. 25: Agregar Configuración de Clases**

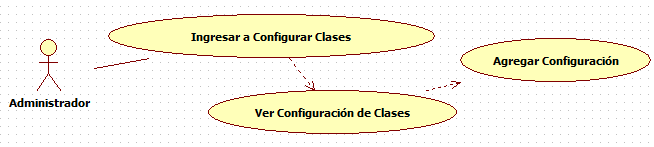
****

Figura 33: Caso de uso– Agregar Configuración de Clases

**Caso de Uso No. 26: Eliminar Configuración de Clases**

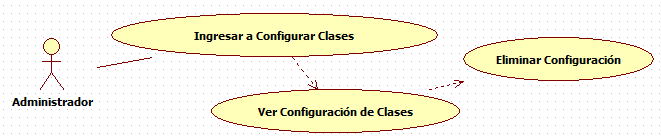
****

Figura 34: Caso de uso– Eliminar Configuración de Clases

**Caso de Uso No. 27: Configurar Parámetros del Algoritmo**

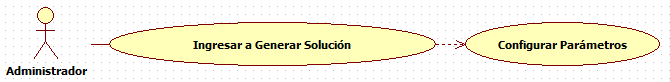
****

Figura 35: Caso de uso– Configurar Parámetros

**Caso de Uso No. 28: Generar Horario**

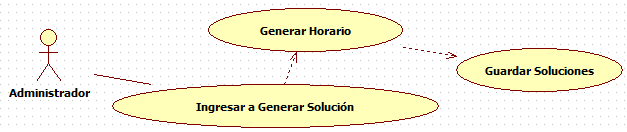
****

Figura 36: Caso de uso– Generar Horario

**Caso de Uso No. 29: Mostrar Todas las Soluciones**

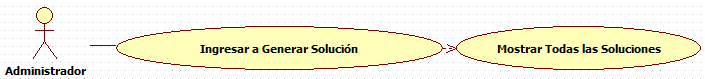
****

Figura 37: Caso de uso– Mostrar Todas las Soluciones

**Caso de Uso No. 30: Ver Horario**

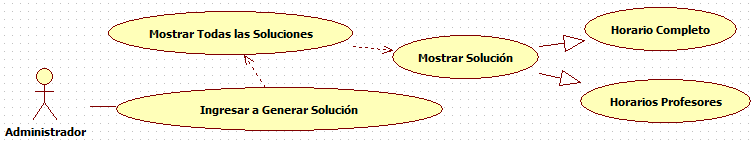
****

Figura 38: Caso de uso– Ver Horario

**Caso de Uso No. 31: Ver Reportes**

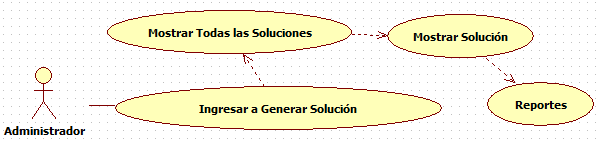
****

Figura 39: Caso de uso– Ver Reportes

**1.3.3 Documentación de Casos de Uso**

|  |  |
| --- | --- |
| ID | No. 1 |
| CASO DE USO | Asignar parámetros de profesores |
| DESCRIPCIÓN | Asignar periodos de tiempo, salones, materias a dictar y cursos a los profesores del colegio para la generación del horario de clases. |
| ACTORES | Administrador, Profesor |
| PRECONDICIONES |  |
| POSTCONDICIONES |  |
| FLUJO NORMAL | 1. El Administrador asigna periodos de tiempo y materias dependiendo de la disponibilidad y especialidad del profesor. 2. El Administrador asigna los salones según al profesor su disponibilidad. 3. El Administrador asigna los cursos al profesor. 4. El profesor pacta y autoriza con el administrador la información. |
| FLUJO ALTERNO |  |

|  |  |
| --- | --- |
| ID | No. 2 |
| CASO DE USO | Configurar Días |
| DESCRIPCIÓN | Realizar la configuración de la cantidad de días a la semana |
| ACTORES | Administrador |
| PRECONDICIONES |  |
| POSTCONDICIONES |  |
| FLUJO NORMAL | 1. El Administrador abre la aplicación. 2. El Administrador da clic en el botón Configurar Periodos. 3. Se abre una nueva ventana en el sistema. 4. El Administrador da clic en el botón Configurar Días. 5. Se abre una nueva ventana en el sistema. 6. El Administrador configura los días y guarda la configuración. 7. El sistema guarda la configuración en la BD |
| FLUJO ALTERNO |  |

|  |  |
| --- | --- |
| ID | No. 3 |
| CASO DE USO | Configurar Clases |
| DESCRIPCIÓN | Realizar la configuración de los bloques y cambios de clase |
| ACTORES | Administrador |
| PRECONDICIONES |  |
| POSTCONDICIONES |  |
| FLUJO NORMAL | 1. El Administrador abre la aplicación. 2. El Administrador da clic en el botón Configurar Periodos. 3. Se abre una nueva ventana en el sistema. 4. El Administrador da clic en el botón Configurar Clases. 5. Se abre una nueva ventana en el sistema. 6. El Administrador configura los días y guarda la configuración. 7. El sistema guarda la configuración en la BD |
| FLUJO ALTERNO |  |

|  |  |
| --- | --- |
| ID | No. 4 |
| CASO DE USO | Crear Nueva Asignatura |
| DESCRIPCIÓN | Crear nuevas asignaturas |
| ACTORES | Administrador |
| PRECONDICIONES |  |
| POSTCONDICIONES |  |
| FLUJO NORMAL | 1. El Administrador abre la aplicación. 2. El Administrador da clic en el botón Configurar Asignaturas. 3. Se abre una nueva ventana en el sistema. 4. El Administrador da clic en el botón Nueva Asignatura 5. El sistema abre una nueva ventana 6. El administrador crea la asignatura. 7. El sistema guarda la nueva asignatura en la BD. 8. El sistema regresa la configuración de asignaturas. |
| FLUJO ALTERNO 1 | 1. El Administrador abre la aplicación. 2. El Administrador da clic en el botón Configurar Asignaturas. 3. Se abre una nueva ventana en el sistema. 4. El Administrador da clic en el botón Nueva Asignatura 5. El sistema abre una nueva ventana 6. El administrador crea la asignatura. 7. El sistema informa que la asignatura ya existe. 8. El sistema regresa a la configuración de asignaturas. |

|  |  |
| --- | --- |
| ID | No. 5 |
| CASO DE USO | Editar Asignatura |
| DESCRIPCIÓN | Editar asignaturas existentes |
| ACTORES | Administrador |
| PRECONDICIONES | La asignatura existe. |
| POSTCONDICIONES |  |
| FLUJO NORMAL | 1. El Administrador abre la aplicación. 2. El Administrador da clic en el botón Configurar Asignaturas. 3. Se abre una nueva ventana en el sistema. 4. El Administrador selecciona una asignatura existente. 5. El Administrador da clic en el botón Editar Datos Básicos 6. El sistema carga los datos de la asignatura 7. El administrador modifica los datos de la asignatura. 8. El sistema actualiza la BD. 9. El sistema regresa la configuración de asignaturas. |
| FLUJO ALTERNO |  |

|  |  |
| --- | --- |
| ID | No. 6 |
| CASO DE USO | Eliminar Asignatura |
| DESCRIPCIÓN | Eliminar asignaturas existentes |
| ACTORES | Administrador |
| PRECONDICIONES | La asignatura existe. |
| POSTCONDICIONES |  |
| FLUJO NORMAL | 1. El Administrador abre la aplicación. 2. El Administrador da clic en el botón Configurar Asignaturas. 3. Se abre una nueva ventana en el sistema. 4. El Administrador selecciona una asignatura existente. 5. El Administrador da clic en el Eliminar 6. El sistema actualiza la BD. 7. El sistema regresa la configuración de asignaturas. |
| FLUJO ALTERNO |  |

|  |  |
| --- | --- |
| ID | No. 7 |
| CASO DE USO | Crear Nuevo Grado |
| DESCRIPCIÓN | Crear nuevos grados |
| ACTORES | Administrador |
| PRECONDICIONES |  |
| POSTCONDICIONES |  |
| FLUJO NORMAL | 1. El Administrador abre la aplicación. 2. El Administrador da clic en el botón Configurar Cursos. 3. Se abre una nueva ventana en el sistema. 4. El administrador hace clic en Nuevo Grado. 5. El sistema abre una nueva ventana. 6. El Administrador crea el grado. 7. El sistema guarda la información en la BD. 8. El sistema regresa a la configuración de Cursos |
| FLUJO ALTERNO | 1. El Administrador abre la aplicación. 2. El Administrador da clic en el botón Configurar Curso. 3. Se abre una nueva ventana en el sistema. 4. El Administrador da clic en el botón Nuevo Grado 5. El sistema abre una nueva ventana 6. El administrador crea l grado. 7. El sistema informa que la asignatura ya existe. 8. El sistema regresa a la configuración de asignaturas. |

|  |  |
| --- | --- |
| ID | No. 8 |
| CASO DE USO | Eliminar Grado |
| DESCRIPCIÓN | Eliminar grados existentes |
| ACTORES | Administrador |
| PRECONDICIONES | El grado existe. |
| POSTCONDICIONES |  |
| FLUJO NORMAL | 1. El Administrador abre la aplicación. 2. El Administrador da clic en el botón Configurar Cursos. 3. Se abre una nueva ventana en el sistema. 4. El Administrador selecciona un grado existente. 5. El administrador hace clic en Eliminar. 6. El sistema actualiza la información en la BD. 7. El sistema regresa a la configuración de Cursos. |
| FLUJO ALTERNO |  |

|  |  |
| --- | --- |
| ID | No. 9 |
| CASO DE USO | Crear Nuevo Curso |
| DESCRIPCIÓN | Crear nuevos cursos |
| ACTORES | Administrador |
| PRECONDICIONES | Existe un grado |
| POSTCONDICIONES |  |
| FLUJO NORMAL | 1. El Administrador abre la aplicación. 2. El Administrador da clic en el botón Configurar Cursos. 3. Se abre una nueva ventana en el sistema. 4. El Administrador selecciona un grado existente. 5. El administrador hace clic en Configuración de Cursos. 6. El sistema abre una nueva ventana. 7. El administrados hace clic en Nuevo Curso 8. El Administrador crea el curso. 9. El sistema guarda la información en la BD. 10. El sistema regresa a la configuración de Cursos. |
| FLUJO ALTERNO | 1. El Administrador abre la aplicación. 2. El Administrador da clic en el botón Configurar Cursos. 3. Se abre una nueva ventana en el sistema. 4. El Administrador selecciona un grado existente. 5. El administrador hace clic en Configuración de Cursos. 6. El sistema abre una nueva ventana. 7. El administrados hace clic en Nuevo Curso 8. El Administrador crea el curso. 9. El sistema informa que el curso ya existe. 10. El sistema regresa a la configuración de cursos. |

|  |  |
| --- | --- |
| ID | No. 10 |
| CASO DE USO | Editar Curso |
| DESCRIPCIÓN | Editar la información de un curso existente |
| ACTORES | Administrador |
| PRECONDICIONES | El grado existe, el curso existe. |
| POSTCONDICIONES |  |
| FLUJO NORMAL | 1. El Administrador abre la aplicación. 2. El Administrador da clic en el botón Configurar Cursos. 3. Se abre una nueva ventana en el sistema. 4. El Administrador selecciona un curso existente. 5. El administrador hace clic en Editar Curso. 6. El sistema carga la información del curso seleccionado. 7. El Administrador edita la información del curso 8. El sistema actualiza la información en la BD. 9. El sistema regresa a la configuración de Cursos. |
| FLUJO ALTERNO |  |

|  |  |
| --- | --- |
| ID | No. 11 |
| CASO DE USO | Eliminar Curso |
| DESCRIPCIÓN | Eliminar cursos existentes |
| ACTORES | Administrador |
| PRECONDICIONES | El grado existe, el curso existe. |
| POSTCONDICIONES |  |
| FLUJO NORMAL | 1. El Administrador abre la aplicación. 2. El Administrador da clic en el botón Configurar Cursos. 3. Se abre una nueva ventana en el sistema. 4. El Administrador selecciona un grado existente. 5. El administrador hace clic en Eliminar. 6. El sistema actualiza la información en la BD. 7. El sistema regresa a la configuración de Cursos. |
| FLUJO ALTERNO |  |

|  |  |
| --- | --- |
| ID | No. 12 |
| CASO DE USO | Crear Nueva Carga Académica |
| DESCRIPCIÓN |  |
| ACTORES | Administrador |
| PRECONDICIONES | Existen grados, existen asignaturas. |
| POSTCONDICIONES |  |
| FLUJO NORMAL | El Administrador abre la aplicación.  El Administrador da clic en el botón Configurar Carga Académica.  Se abre una nueva ventana en el sistema.  El Administrador selecciona un grado existente.  El administrador hace clic en Nueva Carga Académica.  El sistema abre una nueva ventana.  El administrador crea la nueva carga académica.  El sistema guarda la información en la BD.  El sistema regresa a la configuración de Carga Académica. |
| FLUJO ALTERNO |  |

**1.3.4 Diagramas de secuencia**

**Diagrama de secuencia No. 1: Configurar Periodos**

****

**Diagrama de secuencia No. 2: Configurar Asignaturas**

****

**Diagrama de secuencia No. 3: Configurar Cursos – Grados**

****

**Diagrama de secuencia No. 4: Configurar Cursos – Cursos**

****

**Diagrama de secuencia No. 5: Configurar Carga Académica**

****

**Diagrama de Secuencia No. 6: Configurar Profesores**

****

**Diagrama de Secuencia No. 7: Configurar Aulas**

****

**Diagrama de Secuencia No. 8 Configurar Clases**

****

**Diagrama de Secuencia No. 9: Generar Horario**

****

**Diagrama de Secuencia No. 10: Reportes**

****

1**.3.5 Diagramas de Actividad**

**Diagrama No.1: Configurar Días**

****

**Diagrama de Actividad No. 2: Configurar Cambios de Clase**

****

**Diagrama de Actividad No. 3: Registrar Asignatura**

****

**Diagrama de Actividad No. 4: Editar Asignatura**

****

**Diagrama de Actividad No. 5: Eliminar Asignatura**

****

**Diagrama de Actividad No. 6: Registrar Grado**

****

**Diagrama de Actividad No. 7: Editar Grado**

****

**Diagrama de Actividad No. 8: Eliminar Grado**

****

**Diagrama de Actividad No. 9: Registrar Nuevo Curso**

****

**Diagrama de Actividad No. 10: Editar Curso**

****

**Diagrama de Actividad No. 11: Eliminar Curso**

****

**Diagrama de Actividad No. 12: Registrar Nueva Carga Académica**

****

**Diagrama de Actividad No. 13: Editar Carga Académica**

****

**Diagrama de Actividad No. 14: Eliminar Carga Académica**

****

**Diagrama de Actividad No. 15: Registrar Nuevo Profesor**

****

**Diagrama de Actividad No. 16: Eliminar Profesor**

****

**Diagrama de Actividad No. 17: Editar Profesor**

****

**Diagrama de Actividad No. 18: Configurar Preferencia de Periodos de Profesores**

****

**Diagrama de Actividad No. 19: Configurar Preferencia de Asignaturas de Profesores**

****

**Diagrama de Actividad No. 20: Crear Aulas**

****

**Diagrama de Actividad No. 21: Eliminar Aula**

****

**Diagrama de Actividad No. 22: Editar Aula**

****

**Diagrama de Actividad No. 23: Crear Configuración de Clases**

****

**Diagrama de Actividad No. 24: Agregar Clase a Configuración Existente**

****

**Diagrama de Actividad No. 25: Editar Clase de Configuración Existente**

****

**Diagrama de Actividad No. 26: Eliminar Clase de Configuración Existente**

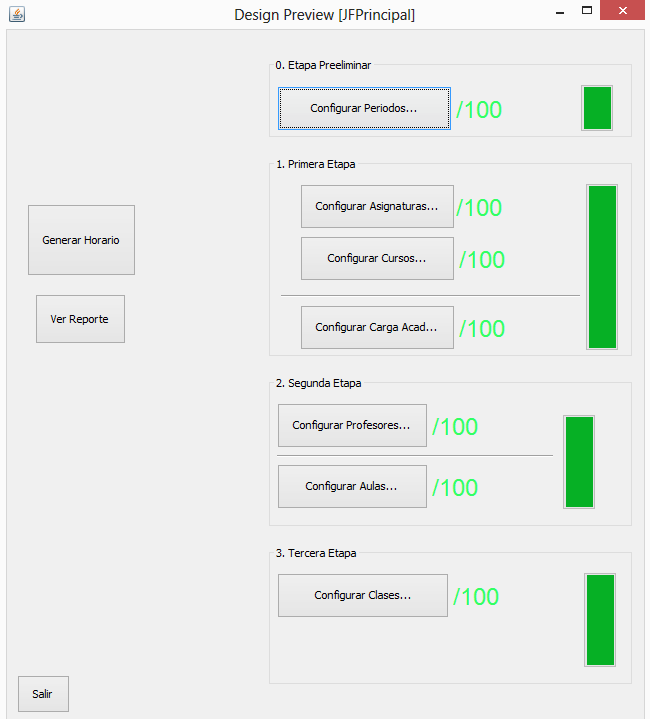
****

**Diagrama de Actividad No. 27: Generar Horario**

****

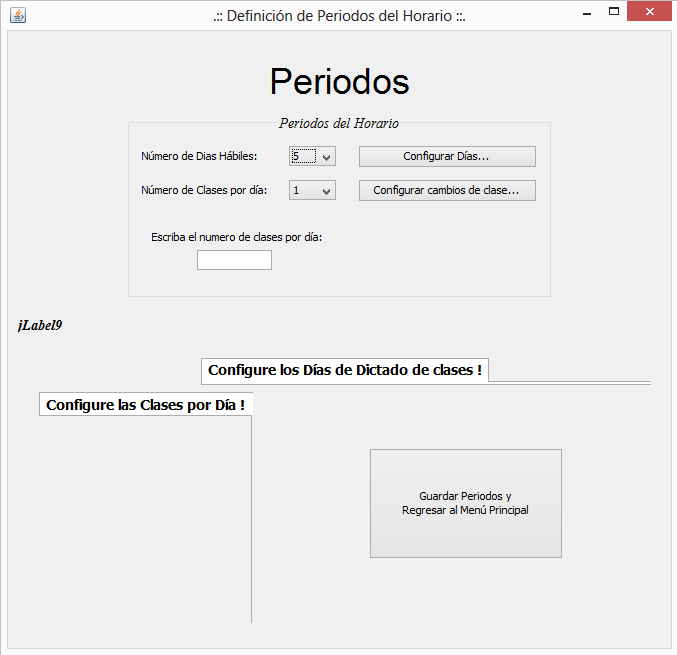
**1.3.6 Diseño de interfaz de Usuario**

La implementación de la solución se realizó en varias fases, por lo cual, se le brindara al usuario una interfaz que indique el orden de cada una de las fases y señalando el avance de cada una de ellas. Cada módulo de configuración le indica al usuario que parámetros se deben configurar y una interfaz predecible para su fácil manejo.

****

**Ventana de Configuración de Periodos**

Por medio de esta el usuario hará la configuración de los periodos de tiempo que se manejan regularmente en el horario del colegio.



**Ventana de Configuración de Asignaturas**

Por medio de esta vista el usuario podrá crear, editar o eliminar las asignaturas en el sistema.

****

**Ventana de Configuración de Cursos**

En esta vista el usuario podrá realizar la creación de los grados (6°, 7°, 8°, etc), al igual que la creación y la configuración de los cursos requeridos.

****

**Ventana de Configuración de Carga Académica**

El usuario podrá realizar la configuración de la carga académica correspondiente a las asignaturas.

****

**Ventana de Configuración de Profesores**

Principalmente, el usuario puede crear, editar y eliminar profesores, así como también hacer la configuración de la disponibilidad horaria de cada profesor y las asignaturas que va a impartir.

****

**Ventana de Configuración de Aulas**

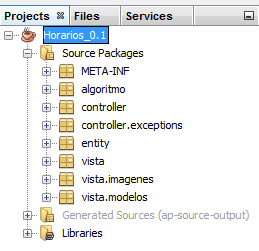
El usuario puede configurar la asignación de aulas para los profesores y/o cursos según corresponda.

****

**1.4 Fase de Implementación:**

La implementación del proyecto se realizó sobre la plataforma de desarrollo Netbeans, utilizando el lenguaje de programación Java.

A continuación se muestra la estructura de paquetes de la aplicación:



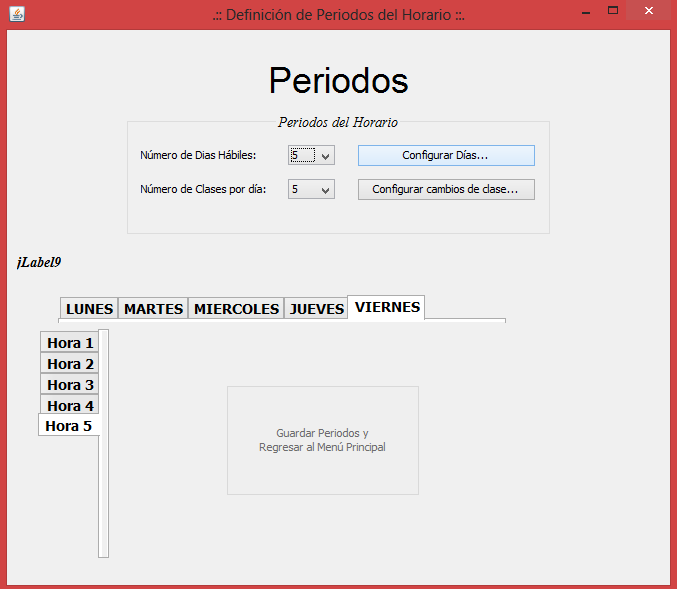
Para el control y conexión con la Base de Datos se manejó el uso de controladores para cada una de las clases creadas, y relacionándolos con las vistas de la aplicación, básicamente se tomó el modelo vista-controlador y se aplicó a nuestro proyecto.

**1.5 Fase de Pruebas del sistema:**

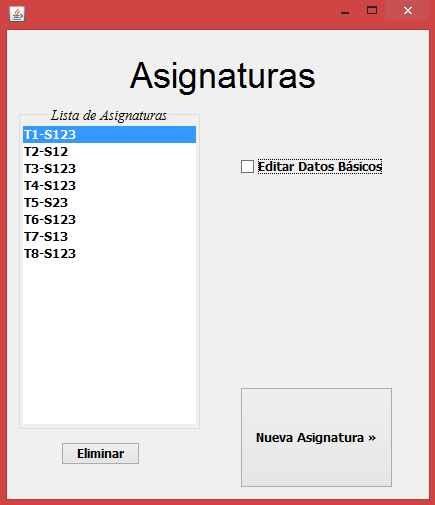
**Caso de Ejemplo**

A continuación se muestra paso a paso la implementación del caso de ejemplo con el cual se realizaron las pruebas del sistema:

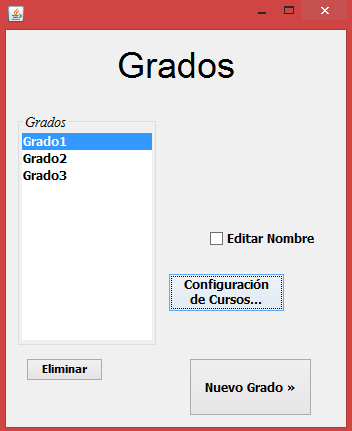
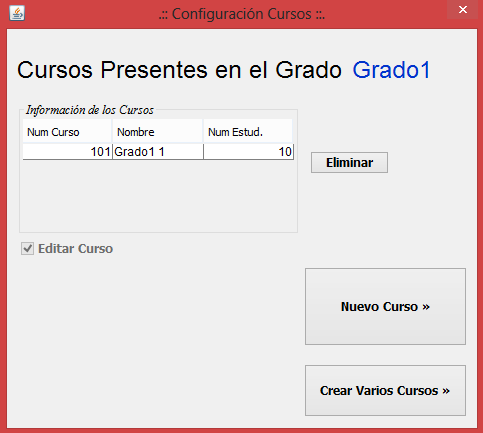
* **Configuración de periodos:** 5 días a la semana con 5 horas diferentes de clase por día.



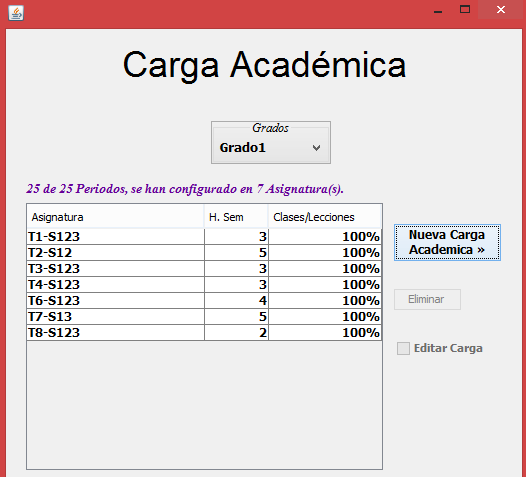
* **Configuración de Asignaturas:** En total 8 asignaturas.



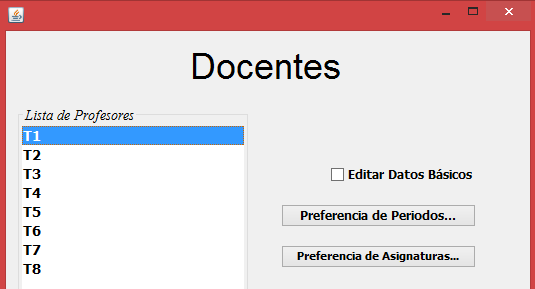
* **Configuración de cursos:** 3 cursos, 1 en cada grado (Grado1 – Curso 101, Grado 2 – Curso 201, Grado 3 – Curso 301). Cada curso cuenta con un total de 10 estudiantes.

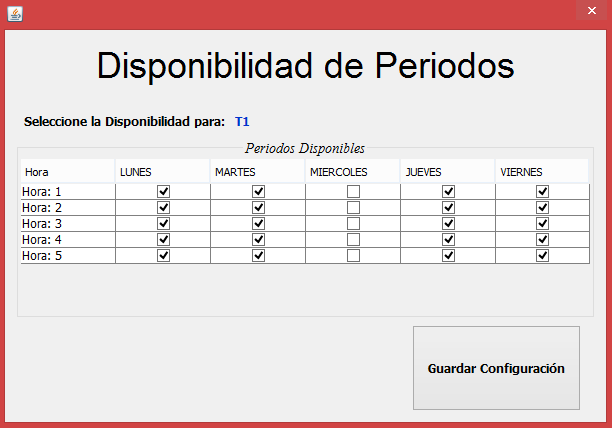
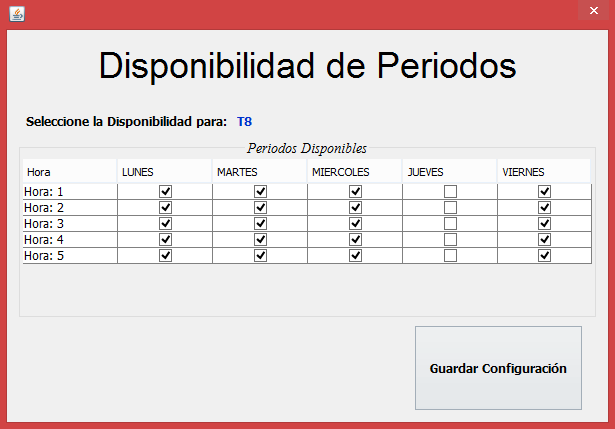
 

* **Configuración de carga académica:** Se asignaron cierta cantidad de horas a la semana en las cuales debe impartirse cada materia, estas varían entre 2, 3, 4 y 5 horas a la semana.

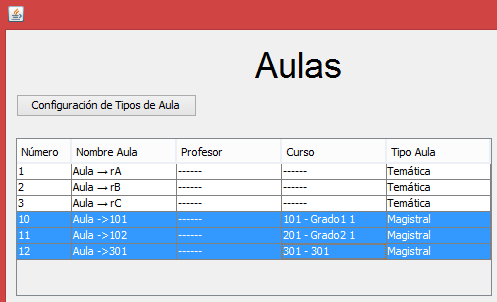


* **Configuración de Profesores:** En total son 8 profesores. A cada profesor se le asignó una configuración de preferencia de periodos y de asignaturas, de tal manera cada uno tiene 1 día libre a la semana, y así, al generar el horario final podamos ver que la solución se ajusta a los parámetros asignados.

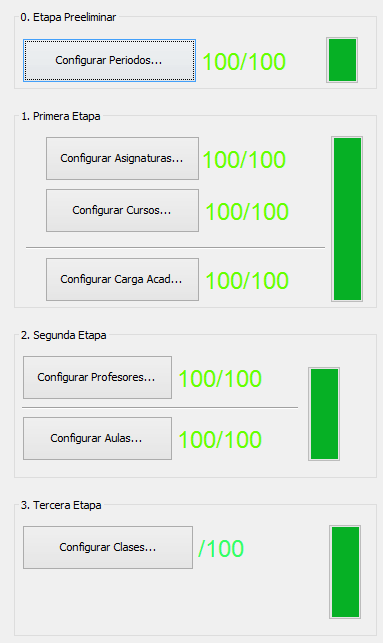


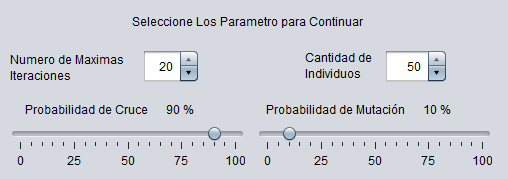
* **Configuración de Aulas:** A cada curso se le asignó 1 aula.



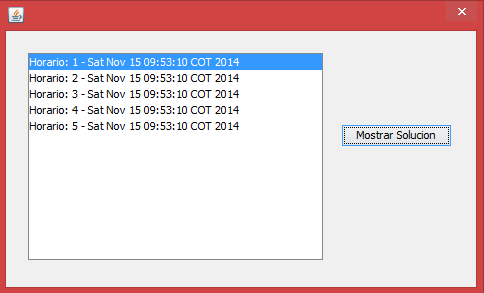
* Finalmente, obtuvimos que la configuración estaba completa, y se procede a la generación de la solución.



* **Parámetros para la ejecución del algoritmo:** Se variaron los parámetros de probabilidad de cruce y mutación, de tal manera que se puedan observar los cambios en cuanto a la cantidad y la calidad de soluciones que se generan para el ejemplo.



* **Generación de la solución:** Una vez configurados todos los parámetros requeridos para generación del horario para el ejemplo. Ya con el horario generado, se pueden visualizar las diferentes soluciones obtenidas y los diferentes reportes de cada uno de ellos.

****

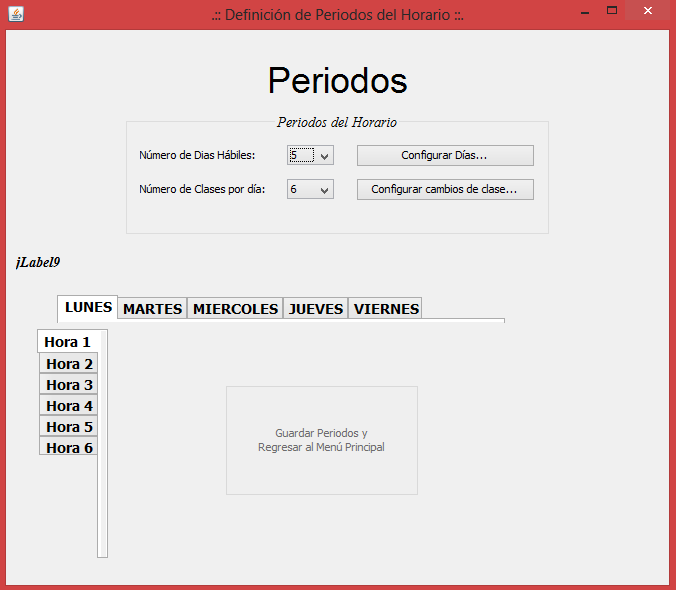
****

**1.6 Fase de Implantación**

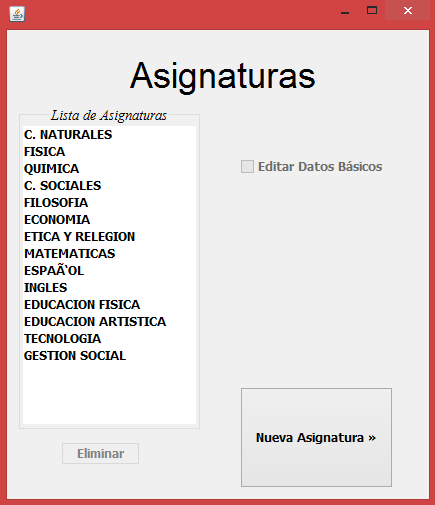
**Caso de Prueba – Colegio María Mercedes Carranza**

Uno de los objetivos de este proyecto fue realizar un caso de prueba real en el colegio María Mercedes Carranza. Allí nos proporcionaron toda la información de los horarios necesaria para evidencia del funcionamiento de la aplicación y el algoritmo implementado.

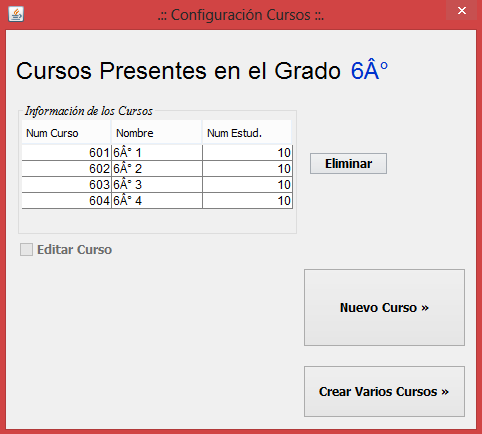
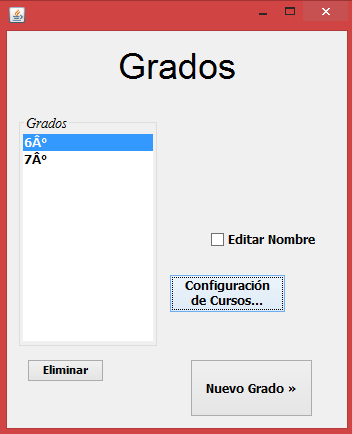
* **Configuración de Periodos:** Actualmente el colegio realiza sus labores escolares de lunes a viernes dictando 6 horas de clase diarias.



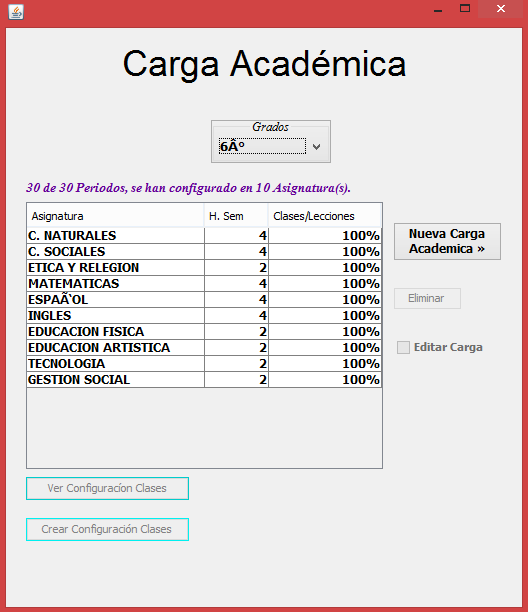
* **Configuración de Asignaturas:** En total se dictan 14 asignaturas.



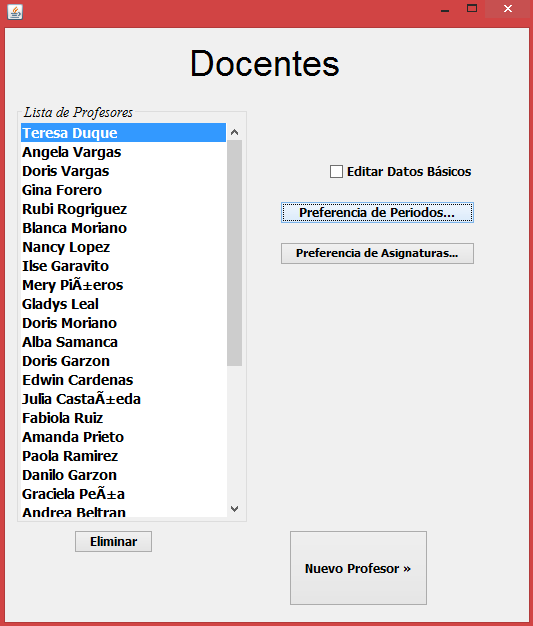
* **Configuración de Cursos:** Tenemos 2 grados, cada uno con 4 cursos.

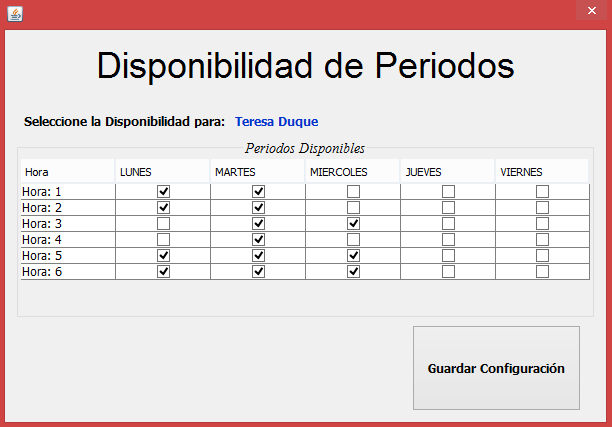
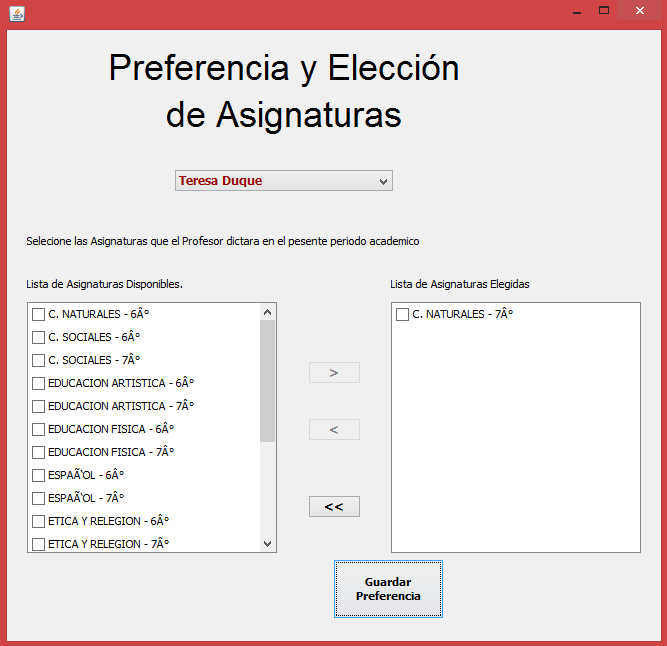
****

* **Configuración de Carga Académica:** Cada materia debe dictarse 2 o 4 horas a la semana según corresponda.

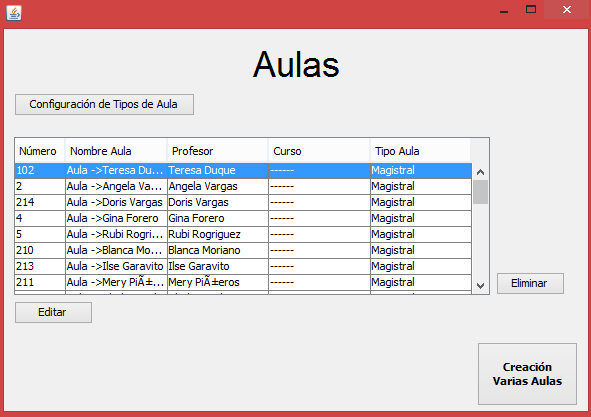
****

* **Configuración de Profesores:** Se cuenta con un total de 33 profesores en la institución. A cada profesor se le asignan los periodos de disponibilidad y asignaturas correspondientes

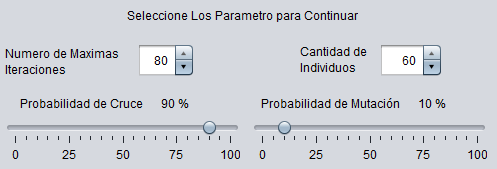
****

****

* **Configuración de Aulas:** Para el caso del colegio Maria Mercedes Carranza, se asigna un aula a cada profesor.



Finalmente con todos los datos guardados y la configuración guardada correctamente, se ejecutó el algoritmo con los siguientes parámetros:



Los parámetros se variaron uno por uno para ver qué efecto tienen en el algoritmo y sobre las soluciones que se van a encontrar manteniendo siempre los datos iniciales de cada uno si no se está estudiando.

**Parámetros Iniciales**

**Número de Máximas Iteraciones:** 50

**Cantidad de Individuos**: 80

**Probabilidad de Cruce:** 90%

**Probabilidad de Mutación:** 18%

* **VARIACIONES DE RESULTADOS MODIFICANDO EL NÚMERO MÁXIMO DE ITERACIONES**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Iteraciones | 5 | 25 | 50 | 75 | 95 |
| Nro. De Prueba |
| 1 |  |  |  |  |  |
| 2 |  |  |  |  |  |
| 3 |  |  |  |  |  |
| 4 |  |  |  |  |  |
| 5 |  |  |  |  |  |
| 6 |  |  |  |  |  |
| 7 |  |  |  |  |  |
| 8 |  |  |  |  |  |
| 9 |  |  |  |  |  |
| 10 |  |  |  |  |  |

**BIBLIOGRAFIA**

[1]: Lü, Z. &Hao, J.-K.: Adaptive tabu search for course timetabling. In: European Journal of Operational Research 200 (2010), Nr. 1, S. 235—244

[2]: Wren, A.: Scheduling, timetabling and rostering --- A special relationship?. In: Burke, E. & Ross, P. (Hrsg.): Springer Berlin Heidelberg. 1153: Practice and Theory of Automated Timetabling., 1996, S. 46-75

[3] Automated Scheduling,Optimisation and Planning(ASAP) Research Group. <http://www.asap.cs.nott.ac.uk/sites/default/files/ASAP_Brochure20092011.pdf>

[4]: CABEZAS GARCIA, J. X.: DISENO E IMPLEMENTACION DE UNA HEURISTICA PARA RESOLVER EL PROBLEMA DE CALENDARIZACION DE HORARIOS PARA UNIVERSIDADES, Dissertation (), ESCUELA SUPERIOR POLITECNICA DEL LITORAL, 2009

[5]: Schönberger, J.; Mattfeld, D. &Kopfer, H.: Memetic Algorithm timetabling for non-commercial sport leagues. In: European Journal of Operational Research 153 (2004), Nr. 1, S. 102 – 116

[6]: Leone, R.; Festa, P. &Marchitto, E.: A Bus Driver Scheduling Problem: aÂ new mathematical model and a GRASP approximate solution. In: Journal of Heuristics 17 (2011), S. 441-466

[7]: Barnhart, C.: Airline Schedule Optimization: John Wiley & Sons, Ltd. :The Global Airline Industry., 2009, S. 183—211

[8]: Rangel-Valdez, N. & Torres-Jimenez, J.: Solving Employee Timetabling in a Call Center of a Telecommunications Company in Mexico with Simulated Annealing. In:: . : Artificial Intelligence, 2009. MICAI 2009. Eighth Mexican International Conference on., 2009, S. 170 -175

[9]: Bai, R.; Burke, E.; Kendall, G.; Li, J. & McCollum, B.: A Hybrid Evolutionary Approach to the Nurse Rostering Problem. In: Evolutionary Computation, IEEE Transactions on 14 (2010), Nr. 4, S. 580 -590

[10]: Adamuthe, A. &Bichkar, R.: Tabu search for solving personnel scheduling problem. In:: . : Communication, Information Computing Technology (ICCICT), 2012 International Conference on., 2012, S. 1 -6

[11]: Larrosa J, Meseguer P. Restricciones Blandas: Modelos y Algoritmos. Inteligencia Artificial. Revista Iberoamericana de Inteligencia Artificial2003;7 Disponible en:http://redalyc.uaemex.mx/redalyc/src/inicio/ArtPdfRed.jsp?iCve=92572006. Consultado el 1 de febrero de 2013.

[12]: Hansen, I.: State-of-the-art of railway operations research. In: Timetable Planning and Information Quality (2010), S. 35

[13]: de Werra, D.: An introduction to timetabling. In: European Journal of Operational Research 19 (1985), Nr. 2, S. 151—162. Disponible en:

<http://www.sciencedirect.com/science/article/pii/0377221785901675>

[14]: Glover, F. & Laguna, M.: Tabu Search, Norwell, MA, USA: Kluwer Academic Publishers., 1997 Disponible en: <http://dl.acm.org/citation.cfm?id=549765>

[15]: Burke, E. K.; Kendall, G.; Mısır, M.; Özcan, E.; Burke, E.; Kendall, G.; Özcan, E. &Mısır, M.: Applications to timetabling. In: :Handbook of Graph Theory, chapter 5.6., 2004 Disponible en: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.188.1458>

[16]: Diaz Fernández, A. &Dowsland, K.: Diseño de heuristicas y fundamentos del recocido simulado. In: Inteligencia artificial: Revista Iberoamericana de Inteligencia Artificial 7 (2003), Nr. 19, S. 93—102. Disponible en: <http://sci2s.ugr.es/docencia/metaheuristicas/Enfriamiento-simulado.pdf>

[17]: Vázquez Espi, M.: Recocido simulado: un nuevo algoritmo para la optimización de estructuras. In: (1994) Disponible en: <http://oa.upm.es/968/>

[18]: Gómez Toro, J. A.; Vanegas Castellanos, J. D. & Zuluaga Gómez, N.: Diseño e implementación de un algoritmo para dar solución al problema de asignación de salones (Timetabling) usando el método de colonia de hormigas. (2009) Disponible en: <http://repositorio.utp.edu.co/dspace/handle/11059/1320>

[19]: Glover, F.: Tabu Search - Part I. In: ORSA Journal on Computing 1 (Summer 1989), Nr. 3, S. 190-206 Disponible en: <http://joc.journal.informs.org/content/1/3/190.short>

[20]: Restrepo, G. & Moreno, L.: Modelo para la Asignación de Recursos Académicos en Instituciones Educativas Utilizando Técnicas Metaheurísticas. In: Avances en Sistemas e Informática 8 (2012), Nr. 3. Disponible en: <http://digital.unal.edu.co/index.php/avances/article/view/22350>

[21]: Dorigo, M.; Birattari, M. &Stutzle, T.: Ant colony optimization. In: ComputationalIntelligence Magazine, IEEE 1 (Nov.), Nr. 4, S. 28-39. Disponible en: <http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4129846>

[22]: Diaz, A. & Fernández, J. L. G.: Optimización heuristica y redes neuronales: Paraninfo., 1996

[23]: Metropolis, N., Rosenbluth, A.W., Rosenbluth, M.N., Teller, A.H., Teller, E.: Equation of state calculation by fast computing machines. Journal of Chemistry Physics, 21: 1087-1091, 1953.

[24]: ABRAMSON, David; KRISHNAMOORTHY, Mohan; DANG, Henry. Simulated annealing cooling schedules for the school timetabling problem. Asia Pacific Journal of Operational Research, 1999, vol. 16, p. 1-22.

Disponible en: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.35.994

[25]: ABRAMSON, David. Constructing school timetables using simulated annealing: sequential and parallel algorithms. Management Science, 1991, vol. 37, no 1, p. 98-113.Disponible en: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.17.302

[26]: Abel Franco Flores.Estudio comparativo de Algoritmos Genéticos y Algoritmos de Búsqueda Tabú para la resolución del Flow Shop Problem.

[27]: SUÁREZ, Joseph Gallart; MANCHEGO, Fernando Alva; NICHO, Anthony Alama Nole3 Gissella Bejarano. Generación Inteligente de Horarios empleando heurísticas GRASP con Búsqueda Tabú para la Pontifica Universidad Católica del Perú. Disponible en: http://www.revistas.pucp.edu.pe/rii/sites/revistas.pucp.edu.pe.rii/files/Joseph\_Gallart.pdf

[28]: CARDEMIL, Andrés. Optimización de fixtures deportivos: Estado del arte y un algoritmo tabusearch para el travelingtournamentproblem. MasterÕsthesis, Universidad de Buenos Aires, Departamento de Computación, Buenos Aires, 2002. Disponible en: http://old.dii.uchile.cl/~gduran/docs/tesis/tesis\_andres.pdf

[29]: Toro Ocampo, E. M., Tabares Espinosa, P., & Granada Echeverry, M. (2004). Método de colonia de hormigas aplicado a la solución del problema de asignación generalizada. Revista Tecnura, 8(15), 66-76. Disponible en: http://tecnura.udistrital.edu.co/ojs/index.php/revista/article/view/151

[30]: Feo, T.A.; Resende, M.G.C. (1989). A probabilistic heuristic for a computationally

difficult set covering problem. Operations Research Letters, Vol. 8, No. 2, pp. 67-71.

[31]: GONZÁLEZ, Fernando Pérez. UNA METODOLOGÍA DE SOLUCIÓN BASADA EN LA METAHEURÍSTICA GRASP PARA EL PROBLEMA DE DISEÑO DE RED CON INCERTIDUMBRE. Disponible en: http://pisis.fime.uanl.mx/ftp/pubs/thesis/msc/2006-fernando\_perez/tesis-fer-2006.pdf

[32]: Pino, R., Martínez, C., Villanueva, V., Priore, P., &Fernández, I. Application of GRASP methodology to Vehicle Routing Problem (VRP).Disponible en: http://elrond.informatik.tu-freiberg.de/papers/WorldComp2012/ICA6018.pdf

[33]: Raúl Esteban Naupari Quiroz, GisselaKatheryn Rosales Gerónimo. Aplicación de algoritmos genéticos para el diseño de un sistema de apoyo a la generación de horarios de clases para la Facultad de Ingeniería de Sistemas e Informática de la UNMSM. Universidad Nacional Mayor de San Marcos. Facultad e Ingeniería de Sistemas e Informática. 2010 Lima – Perú.

[34]: Elvira Mayordomo. NP-completos. Universidad de Zaragoza. Zaragoza – España. Disponible en; <http://webdiis.unizar.es/~elvira/mac/npcompletos.pdf>

[35]: BEJARANO NICHO, Gissella María. Planificación de horarios del personal de cirugía de un hospital del Estado aplicando algoritmos genéticos (Time TablingProblem). 2011. Disponible en: <http://tesis.pucp.edu.pe/repositorio/bitstream/handle/123456789/551/BEJARANO_NICHO_GISSELLA_MAR%C3%8DA_PLANIFICACI%C3%93N_HORARIOS_PERSONAL_CIRUG%C3%8DA.pdf?sequence=1>

[36]: Francisco J. Martínez Ruiz, Eduardo García Sánchez, Jaime Muñoz Arteaga, Carlos H. Castañeda Ramírez. Timetabling Académico Usando Algoritmos Genéticos y Programación Celular. Universidad Autónoma de Zacatecas. Departamento de Ingeniería en Computación. México. Disponible en: <http://ingsw.ccbas.uaa.mx/sitio/images/pdfpublicaciones/artiCoNaCiCo05-20.pdf>

[37]: Arranz de la Peña, J. Parra Truyol, A. Algoritmos Genéticos. Disponible en: <http://www.it.uc3m.es/jvillena/irc/practicas/06-07/05.pdf>

[38] Mejía Caballero, J. M., & Paternina Arboleda, C. D. (2009). Asignación de horarios de clases universitarias mediante algoritmos evolutivos (Doctoral dissertation).

[39]Tolmos Rodriguez-Piñero, Piedad. Introducción a los Algoritmos Genétivos y sus Aplicaciones.

[40]Pitol Reyes, Fermín Jesús. Uso de Algoritmos Evolutivos para resolver el Problema de Asignación de Horarios Escolares de la Facultad de Psicologia de la Universidad Veracruzana.

[41]Manrique Gamos, Daniel. Computación Evolutiva: Algoritmos Genéticos.

[42]Lance D. Chambers. Practical Handbook of GENETIC ALGORITHMS, Volumen III. CRC Press LLC, 1999

[43]Franco Baquero, John Fredy, Toro Ocampo, Eliana Mirledy, & Gallego Rendón, Ramón Alfonso. (2008). Problema de asignación óptima de salones resuelto con Búsqueda Tabú. Ingeniería y Desarrollo, (24), 149-175. Retrieved October 03, 2014, from <http://www.scielo.org.co/scielo.php?script=sci_arttext&pid=S0122-34612008000200011&lng=en&tlng=es>.

[44] Philippe Kruchten, The Rational Unified Process An Introduction, Addison Wesley, 2001.

[45] Martinez, Raul & Martinez, Alejandro, Guía a Rational Unified Process, Escuela Politécnica Superior de Albacete – Universidad de Castilla la Mancha.