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

MAURICIO ANDRÉS GUERRA CUBILLOS

ERWIN HAMID PARDO QUIROGA

UNIVERSIDAD DISTRITAL FRANCISCO JOSÉ DE CALDAS

FACULTAD TECNOLÓGICA

TECNOLOGÍA EN SISTEMATIZACIÓN DE DATOS

BOGOTÁ D.C.

COLOMBIA

2014

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

MAURICIO ANDRÉS GUERRA CUBILLOS

20091078058

ERWIN HAMID PARDO QUIROGA

20091078078

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

TUTOR-DIRECTOR:

ROBERTO EMILIO SALAS RUIZ

UNIVERSIDAD DISTRITAL FRANCISCO JOSÉ DE CALDAS

FACULTAD TECNOLÓGICA

TECNOLOGÍA EN SISTEMATIZACIÓN DE DATOS

BOGOTÁ D.C.

COLOMBIA

2014

Nota de aceptación:

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

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

Tutor

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

Jurado

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

Jurado

Bogotá D.c., 10 de Febrero de 2014

CONTENIDO

pág.

[RESUMEN 10](#_Toc411339995)

[ABSTRACT 10](#_Toc411339996)

[INTRODUCCIÓN 11](#_Toc411339997)

[1 Fase de definición, planeación y organización 12](#_Toc411339998)

[1.1 Titulo del trabajo 12](#_Toc411339999)

[1.2 Tema 12](#_Toc411340000)

[1.3 Planteamiento del problema 12](#_Toc411340001)

[1.3.1.1 Descripción del problema. 12](#_Toc411340002)

[1.3.2 Realidad problemática. 13](#_Toc411340003)

[1.3.3 Formulación del problema 14](#_Toc411340004)

[1.4 Alcance y delimitaciones 15](#_Toc411340005)

[1.4.1 Alcances 15](#_Toc411340006)

[1.4.2 Limitaciones 15](#_Toc411340007)

[1.5 Objetivos 16](#_Toc411340008)

[1.5.1 Objetivo general 16](#_Toc411340009)

[1.5.2 Objetivos específicos 16](#_Toc411340010)

[1.6 Justificación 17](#_Toc411340011)

[1.7 Marco de referencia 18](#_Toc411340012)

[1.7.1 Marco histórico 18](#_Toc411340013)

[1.7.1.1 Timetabling y métodos de resolución implementados 18](#_Toc411340014)

[1.7.1.2 Comparación de métodos meta-heurísticos 22](#_Toc411340015)

[1.7.1.3 Breve historia de los algoritmos genéticos 23](#_Toc411340016)

[1.7.2 Marco teórico 24](#_Toc411340017)

[1.7.2.1 Introducción a los Algoritmos Genéticos y su analogía con la naturaleza 24](#_Toc411340018)

[1.7.2.2 Componentes de un Algoritmo Genético 25](#_Toc411340019)

[1.7.2.3 Funcionamiento de un Algoritmo Genético 26](#_Toc411340020)

[1.7.2.4 Codificación del genotipo 27](#_Toc411340021)

[1.7.2.5 Operadores genéticos 28](#_Toc411340022)

[1.7.2.6 Parámetros de un AG 28](#_Toc411340023)

[1.7.2.7 Metodología RUP 29](#_Toc411340024)

[1.7.3 Marco conceptual 30](#_Toc411340025)

[1.7.3.1 Timetabling 30](#_Toc411340026)

[1.7.3.2 Educational Timetabling 31](#_Toc411340027)

[1.7.3.3 Clasificación de restricciones 32](#_Toc411340028)

[1.7.3.4 Complejidad computacional 33](#_Toc411340029)

[1.8 Factibilidad 35](#_Toc411340030)

[1.8.1 Factibilidad técnica 35](#_Toc411340031)

[1.8.1.1 Hardware 35](#_Toc411340032)

[1.8.1.2 Software 36](#_Toc411340033)

[1.8.2 Factibilidad operativa 36](#_Toc411340034)

[1.8.3 Factibilidad económica 37](#_Toc411340035)

[1.8.3.1 Análisis Costos – Beneficios. 37](#_Toc411340036)

[1.8.3.2 Relación Costo – Beneficio. 40](#_Toc411340037)

[1.8.4 Factibilidad legal 41](#_Toc411340038)

[1.9 Cronograma de actividades 42](#_Toc411340039)

[2 Fase de inicio 43](#_Toc411340040)

[2.1 Modelado del negocio 43](#_Toc411340041)

[2.1.1 Modelo del dominio 44](#_Toc411340042)

[2.2 Requerimientos 45](#_Toc411340043)

[2.2.1 Requisitos de la interfaz externa 45](#_Toc411340044)

[2.2.1.1 Interfaz con el usuario 45](#_Toc411340045)

[2.2.1.2 Interfaz con el hardware 46](#_Toc411340046)

[2.2.1.3 Interfaz con el software 46](#_Toc411340047)

[2.2.2 Requisitos funcionales 46](#_Toc411340048)

[2.2.2.1 Diagramas de caso de uso del negocio 46](#_Toc411340049)

[2.2.3 Requerimientos de desarrollo y restricciones de diseño. 48](#_Toc411340050)

[2.2.4 Atributos del sistema. 48](#_Toc411340051)

[2.2.5 Otros requisitos. 49](#_Toc411340052)

[2.2.6 Flujos de trabajo 50](#_Toc411340053)

[3 fase de elaboración 64](#_Toc411340054)

[3.1 análisis 64](#_Toc411340055)

[3.1.1 Modelos de Casos de Uso - Análisis 64](#_Toc411340056)

[3.1.2 Arquitectura de software 64](#_Toc411340057)

[3.2 diseño 65](#_Toc411340058)

[3.2.1 Diagramas de secuencia 65](#_Toc411340059)

[3.2.2 Modelo de datos 71](#_Toc411340060)

[3.2.3 Diccionario de datos 71](#_Toc411340061)

[3.2.4 Prototipos de interfaz de usuario 78](#_Toc411340062)

[4 Fase de construcción 86](#_Toc411340063)

[4.1 Adaptación del algoritmo genético 86](#_Toc411340064)

[4.1.1 Unidad de asignación 86](#_Toc411340065)

[4.1.2 Representación del Problema 87](#_Toc411340066)

[4.1.3 Codificación mediante alfabetos duros. 89](#_Toc411340067)

[4.1.4 Inicialización de la población 89](#_Toc411340068)

[4.1.5 Función de evaluación y manejo de las restricciones 93](#_Toc411340069)

[4.1.6 Restricciones del problema “school timetablig” a evaluar 94](#_Toc411340070)

[4.1.6.1 Restricciones duras (Obligatorias) 94](#_Toc411340071)

[4.1.6.2 Restricciones blandas (Deseadas) 95](#_Toc411340072)

[4.1.7 Criterio de parada 95](#_Toc411340073)

[4.1.8 Operadores genéticos 96](#_Toc411340074)

[4.1.8.1 Selección 96](#_Toc411340075)

[4.1.8.2 Cruce 97](#_Toc411340076)

[4.1.8.3 Mutación 98](#_Toc411340077)

[4.1.8.4 Reemplazo 99](#_Toc411340078)

[4.2 Pruebas 100](#_Toc411340079)

[4.2.1 Introducción 100](#_Toc411340080)

[4.2.2 Caso de análisis 100](#_Toc411340081)

[4.2.3 Esquema de pruebas 101](#_Toc411340082)

[4.2.3.1 Prueba sobre el tamaño del torneo 102](#_Toc411340083)

[4.2.3.2 Prueba de porcentajes de cruce 103](#_Toc411340084)

[4.2.3.3 Prueba de porcentaje de mutación 104](#_Toc411340085)

[5 Fase de transiciòn 106](#_Toc411340086)

[5.1 Despliegue 106](#_Toc411340087)

[5.2 Resulados 106](#_Toc411340088)

[6 Conclusiones 107](#_Toc411340089)

[BIBLIOGRAFIA 107](#_Toc411340090)

LISTA DE TABLAS

pág.

[Tabla 1. Características de las técnicas meta-heurísticas estudiadas 22](#_Toc411340091)

[Tabla 2. Comparación y calificación de meta-heurísticas 23](#_Toc411340092)

[Tabla 3. Comparación entre la asignación de horarios escolares y universitarios 32](#_Toc411340093)

[Tabla 4. Características técnicas del hardware existente 35](#_Toc411340094)

[Tabla 5. Ejemplo beneficios tangibles 39](#_Toc411340095)

[Tabla 6. Documentación caso de uso del negocio 1 47](#_Toc411340096)

[Tabla 7. Documentación caso de uso del negocio 2 47](#_Toc411340097)

[Tabla 8. Requerimientos no funcionales (atributos del sistema) 48](#_Toc411340098)

[Tabla 9. Tabla aula\_tipo 72](#_Toc411340099)

[Tabla 10. Tabla aula 72](#_Toc411340100)

[Tabla 11. Tabla curso 72](#_Toc411340101)

[Tabla 12. Tabla area 73](#_Toc411340102)

[Tabla 13.Tabla asignatura 73](#_Toc411340103)

[Tabla 14. Tabla grado 73](#_Toc411340104)

[Tabla 15. Tabla carga\_academica 73](#_Toc411340105)

[Tabla 16. Tabla pref\_profesor\_materia 74](#_Toc411340106)

[Tabla 17. pref\_profesor\_periodo 74](#_Toc411340107)

[Tabla 18. Tabla periodo 74](#_Toc411340108)

[Tabla 19. Tabla profesor 75](#_Toc411340109)

[Tabla 20. Tabla dia 75](#_Toc411340110)

[Tabla 21. Tabla hora 75](#_Toc411340111)

[Tabla 22. pref\_carga\_periodo 76](#_Toc411340112)

[Tabla 23. Tabla clase 76](#_Toc411340113)

[Tabla 24. Tabla restriccion\_tipo 77](#_Toc411340114)

[Tabla 25. Tabla restriccion 77](#_Toc411340115)

[Tabla 26. Tabla horario 77](#_Toc411340116)

[Tabla 27. Tabla solucion 78](#_Toc411340117)

[Tabla 28. Horas de dictado de Matemáticas para el grado sexto 87](#_Toc411340118)

[Tabla 29. Clases generadas para la asignatura Matemáticas del grado sexto 87](#_Toc411340119)

[Tabla 30. Resultados prueba tamaño del torneo 102](#_Toc411340120)

[Tabla 31. Porcentajes de Cruce 103](#_Toc411340121)

[Tabla 32. Resultados Prueba Porcentaje de Cruce 104](#_Toc411340122)

[Tabla 33. Porcentajes de Mutación 104](#_Toc411340123)

[Tabla 34. Resultados de la Prueba de Porcentaje de Mutación 105](#_Toc411340124)

LISTA DE FIGURAS

pág.

[Figura 1. Técnicas en la búsqueda de soluciones a problemas de optimización 19](#_Toc411346258)

[Figura 2. Diagrama de flujo de un AG 27](#_Toc411346259)

[Figura 3. Individuo binario de un AG 27](#_Toc411346260)

[Figura 4. Diagrama Metodología RUP 30](#_Toc411346261)

[Figura 5. Complejidad Computacional 34](#_Toc411346262)

[Figura 6. Diagrama Gantt de actividades 42](#_Toc411346263)

[Figura 7. Modelo del dominio 45](#_Toc411346264)

[Figura 8. Diagrama de Caso de Uso del Negocio 1 46](#_Toc411346265)

[Figura 9. Diagrama de Caso de Uso del Negocio 2 47](#_Toc411346266)

[Figura 10. Diagrama de actividad - Configurar Días 50](#_Toc411346267)

[Figura 11. Diagrama de actividad - Configurar cambios de Clase 51](#_Toc411346268)

[Figura 12. Diagrama de actividad - Registrar Asignatura 51](#_Toc411346269)

[Figura 13. Diagrama de actividad - Editar Asignatura 52](#_Toc411346270)

[Figura 14. Diagrama de actividad - Eliminar Asignatura 52](#_Toc411346271)

[Figura 15. Diagrama de actividad - Registrar Grado 53](#_Toc411346272)

[Figura 16. Diagrama de actividad - Editar Grado 53](#_Toc411346273)

[Figura 17. Diagrama de actividad - Eliminar Grado 54](#_Toc411346274)

[Figura 18. Diagrama de actividad - Registrar Nuevo Curso 54](#_Toc411346275)

[Figura 19. Diagrama de actividad - Editar Curso 55](#_Toc411346276)

[Figura 20. Diagrama de actividad - Eliminar Curso 55](#_Toc411346277)

[Figura 21. Diagrama de actividad - Registrar Nueva Carga Académica 56](#_Toc411346278)

[Figura 22. Diagrama de actividad - Editar Carga Académica 56](#_Toc411346279)

[Figura 23. Diagrama de actividad - Eliminar Carga Académica 57](#_Toc411346280)

[Figura 24. Diagrama de actividad - Registrar Nuevo Profesor 57](#_Toc411346281)

[Figura 25. Diagrama de actividad - Editar Profesor 58](#_Toc411346282)

[Figura 26. Diagrama de actividad - Eliminar Profesor 58](#_Toc411346283)

[Figura 27. Diagrama de actividad - Configurar Preferencia de Periodos de Profesor 59](#_Toc411346284)

[Figura 28. Diagrama de actividad - Configurar Preferencia de Asignaturas de Profesor 59](#_Toc411346285)

[Figura 29. Diagrama de actividad - Registrar Aula 60](#_Toc411346286)

[Figura 30. Diagrama de actividad - Editar Aula 60](#_Toc411346287)

[Figura 31. Diagrama de actividad - Eliminar Aula 61](#_Toc411346288)

[Figura 32. Diagrama de actividad - Crear Configuración de Clases 61](#_Toc411346289)

[Figura 33. Diagrama de actividad - Agregar Clase a Configuración Existente 62](#_Toc411346290)

[Figura 34. Diagrama de actividad - Editar Clase a Configuración Existente 62](#_Toc411346291)

[Figura 35. Diagrama de actividad - Eliminar Clase a Configuración Existente 63](#_Toc411346292)

[Figura 36. Diagrama de actividad - Generar Horario 63](#_Toc411346293)

[Figura 37. Arquitectura de software 64](#_Toc411346294)

[Figura 38. Diagrama de Secuencia - Configurar Periodos 66](#_Toc411346295)

[Figura 39. Diagrama de Secuencia - Configurar Asignaturas 66](#_Toc411346296)

[Figura 40. Diagrama de Secuencia - Configurar Grados 67](#_Toc411346297)

[Figura 41. Diagrama de Secuencia - Configurar Cursos 67](#_Toc411346298)

[Figura 42. Diagrama de Secuencia - Configurar Carga Académica 68](#_Toc411346299)

[Figura 43. Diagrama de Secuencia - Configurar Profesores 68](#_Toc411346300)

# 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.

Palabras Clave: Algoritmos genéticos, Timetabling

# ABSTRACT

This paper presents a research for carrying of a system capable of automatically generating class schedules for schools, giving solution to School Timetabling problem, implementing genetic algorithms to solve this problem. The theoretical argument shows the different methods by which it is possible to solve the School Timetabling problem, studying with a little more depth the theme of Genetic Algorithms, its history and development. Likewise, the concept of Timetabling, their classifications, characteristics and computational complexity is delivered.

Finally, the results of the development of genetic algorithm for solving School Timetabling problem using the Java programming language are shown, applying the final product at school Maria Mercedes Carranza, which in this case works as a pilot test.

Keywords:

# INTRODUCCIÓN

En cualquier institución educativa (universidad, colegio, escuela, instituto, entre otras) se realizan diferentes actividades como lo son: la creación de un manual de convivencia, la conformación de un consejo académico y/o directivo, la elección de un coordinador, un representante estudiantil, entre otras labores.

Una tarea imprescindible que se realiza cada periodo académico (semestre, año escolar, trimestre, etc.) es la creación del horario académico el cual en la mayoría de los casos se implementa para todo el ciclo académico de dicha institución, donde la principal dificultad es el costo que tiene realizar esta labor, debido que en diversos lugares se efectúa de forma manual.

En este documento se presenta un trabajo de investigación para la generación automatizada de horarios de clase en los colegios, explicamos cómo ha sido un problema que se ha tratado desde varios años atrás, conocido en la literatura (en inglés) como School Timetabling, o Calendarización en español y las diversas técnicas que se utilizan para dar solución a este problema.

Más adelante se propone la implementación de la técnica de inteligencia artificial conocida como Algoritmos Genéticos, como una alternativa en la solución del problema a tratar. Se expone las características, ventajas y desventajas de esta técnica, asimismo el procedimiento a seguir en la creación de este algoritmo.

Por último se crea un programa (software) que permita introducir los datos necesarios que conforman un horario escolar (profesores, cursos, aulas, asignaturas) y en conjunto implementar el algoritmo genético propuesto, de tal manera se ha decido tomar los datos de ejemplo de una institución educativa colombiana para la sección de pruebas y evaluación del software diseñado.

1. Fase de definición, planeación y organización

El primer capítulo tiene como fin identificar y describir el problema de la calendarización en las escuelas y colegios, más adelante se estipulan los objetivos y alcances de este proyecto de grado, al final se centra en el contexto del problema school timetabling desde los ámbitos: histórico, teórico y conceptual.

* 1. Titulo del trabajo

El siguiente titulo encapsula de manera global el propósito de este proyecto de grado. En las próximas secciones se descomponen los conceptos que se incluyen.

*Implementación de algoritmos genéticos para la resolución del problema school timetabling en las instituciones educativas.*

* 1. Tema

Durante el desarrollo de este proyecto se tratan dos temas principales, el *timetabling* como un problema de programación horaria de un número de recursos reflejado en las escuelas, colegios e institutos, y en segundo lugar la técnica de inteligencia artificial, conocida con el nombre de *Algoritmos Genéticos* como una alternativa para dar solución al problema mencionado.

* 1. Planteamiento del problema

En seguida se describe como se manifiesta el problema del timetabling en las escuelas, colegios y en especifico en las instituciones colombianas; se determinan los componentes de este problema, para definir en manera clara una solución al school timetabling.

* + 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 de un horario académico para el nuevo ciclo académico. En 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:

* Dos lecciones de clase no pueden ser programadas en la misma aula.
* Un docente no puede tener dos o más lecciones asignadas en un periodo.

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 debe dictar en la última hora de cada día.

El proceso de distribución horaria (Timetabling) es parte de los problemas de programación en general, que consiste en la asignación de una cantidad de recursos en bloques de tiempo establecidos, que cumplan una serie de restricciones. De esta manera la asignación de profesores, aulas, asignaturas, cursos, en periodos de tiempo semanales con el fin de satisfacer restricciones obligatorias y deseadas para una escuela o colegio, es un problema de programación horaria conocido como “*School Timetabling*”.

Sin embargo la definición de este problema es similar al “*Univeristy Timetabling*”, la principal diferencia entre una colegio y una universidad son los estudiantes ya que en una escuela un grupo de alumnos (curso) toman las mismas asignaturas, en contraste, en las universidades los estudiantes programan distintas asignaturas de varios semestres. En la sección 1.7.3 se detallan estos conceptos y se describen otras diferencias.

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. En el ámbito de las ciencias computaciones, los problemas de tipo timetabling están clasificados como NP-Completo que pertenecen a la clase NP; dentro de esta clase se encuentra el “*problema de satisfacibilidad*” (SAT). SAT es fundamental para la solución de varios problemas en razonamiento automático, diseño y manufactura asistida por computadoras, visión computacional, bases de datos, robótica, diseño de circuitos integrados, arquitectura de computadoras, redes de computadoras y asignación de tareas, entre otros. Los métodos para resolver el problema SAT juegan un papel crucial en el desarrollo de sistemas informáticos eficaces y las aplicaciones de dicho problema.[[1]](#footnote-1)El "*School Timetabling*" es un ejemplo de este tipo de problemas; existen técnicas para la solución a este problema como las técnicas metaheurísticas que combinan la simplicidad de sus ideas con la gran eficiencia para dar buenos resultados en problemas complejos.

* + 1. Realidad problemática.

En muchos colegios del 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 ciclo a otro, por ej.:

En el año 2011 un determinado colegio cuenta con tres cursos en el grado sexto y dos en séptimo, para el año 2012 lo más probable es, que para el grado séptimo se cuente con tres cursos y para el grado octavo dos. 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.

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 creación de horarios académicos, con la implementación de técnicas de Inteligencia Artificial, para dar solución al problema, mediante la obtención de horarios correctos. Una de las 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.

Por tanto, se busca desarrollar una herramienta computacional, que permita explorar el espacio de búsqueda creado de cada colegio, teniendo en cuenta las restricciones y arrojar como solución 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, correspondiente a los recursos que trata este problema y la implementación de la técnica Algoritmos Genéticos.

* + 1. Formulación del problema

¿Cómo implementar algoritmos genéticos para la resolución del problema School Timetabling en la generación de horarios de las instituciones educativas, con el fin de automatizar el proceso de generar un horario de clases reduciendo el tiempo y los recursos empleados en su elaboración?

* 1. Alcance y delimitaciones

A continuación se definen los aspectos que alcanzaremos en la investigación y limitaciones que tiene este proyecto.

* + 1. Alcances

Con la ejecución de este proyecto se busca que el problema del school timetabling se pueda corregir en una institución educativa colombiana, mediante la implementación de un algoritmo genético capaz de organizar y encontrar uno o varios horarios viables, con el fin de evitar que esta labor se realice de forma manual como se hace en la actualidad. También se conocerá y describirá el proceso que lleva a cabo un algoritmo genético, de tal manera que se pueda adecuar esta problemática, al final se diseñará un programa de escritorio que facilitará la configuración y ejecución del algoritmo para apreciar los resultados arrojados y el usuario pueda decidir cual horario implementar.

* + 1. Limitaciones

El proyecto *Implementación de Algoritmos Genéticos para la Resolución del Problema School Timetabling en las Instituciones Educativas* está limitado inicialmente a la organización de horarios de clase en los colegios de Bogotá, de esta manera se creará una estructura que se adapte al modelo del problema del Colegio María Mercedes Carranza y otras instituciones que se ajusten a esta estructura.

El estudio realizado abarca la programación de algoritmos genéticos y la resolución del problema School Timetabling, por tanto, queda por fuera de este proyecto, la comparación de los horarios generados con la implementación de otra técnica de inteligencia artificial, que también pueda dar solución al problema.

Técnicamente, el proyecto va a ser desarrollado en el lenguaje de programación java usando el IDE Netbeans y PostgreSQL como el motor de la base de datos, por consiguiente en este documento se limita únicamente la configuración necesaria para que el programa pueda ejecutarse y no se especifica el proceso de instalación del software necesario para el desarrollo de la aplicación.

* 1. Objetivos
     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. Objetivos específicos
* Investigar acerca de 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 obligatorias y deseadas 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. Justificación

Hoy en día aprovechar todas las herramientas que están a disposición de las personas facilita la realización de una tarea determinada. Desarrollar una herramienta tecnológicamente hablando, como es un programa trae diversos beneficios, si hablamos de una aplicación que permita la generación automática de horarios de clase en los colegios, definitivamente va a reducir el costo que se requiere para la realización de esta labor como en el recurso humano involucrado.

Trasladar a una herramienta computacional el manejo del los datos implicados en la creación de un horario de clases para una institución educativa, por ejemplo el Colegio María Mercedes Carranza, tiene como fin manipularlos de forma práctica, evitando formularios en papel. Asimismo, se cuenta con la ventaja de acceder a esta información fácilmente, con la funcionalidad de editar, eliminar y agregar datos sencillamente desde un ordenador.

Otro punto a favor es la utilización de una técnica que realice la búsqueda de horarios académicos, ya que se optimiza este proceso mediante el uso de un computador y no de forma manual como se efectúa en varias de las instituciones educativas colombianas. De esta manera se exploran más zonas del espacio de búsqueda para mejorar en la asignación de recursos y lograr satisfacer restricciones deseadas como la preferencia de horas de dictado por parte de los docentes.

Implementar algoritmos genéticos demuestra que términos relacionadas de la biología y genética, como la selección natural de Darwin, reproducción sexual, mutaciones, se pueden abstraer para ser simulados en un programa de forma simple, dando resolución a problemas de alta complejidad como el school timetabling.

Además se está diseñando una aplicación que pueda ser utilizada por otras instituciones en la creación del horario de clases para un determinado ciclo académico ya sea un año, trimestre, semestre, o uno personalizado.

* 1. Marco de referencia

Para definir un marco de este proyecto, recordemos que el school timetabling se desprende del timetabling, de esta manera, ahora se explica cómo éste ha sido tratado, desde las técnicas no tradicionales que se han usado para resolverlo, centrando la investigación en los algoritmos genéticos como una metaheurística que simula procesos de evolución. Al final se expone un marco conceptual para aclarar términos de manera formal que se emplean en este documento.

* + 1. Marco histórico

En esta sección se presentan algunas de las técnicas que se han utilizado para dar solución al problema timetabling, estas se han seleccionado debido al gran éxito en la resolución de problemas optimización, que se pueden implementar para nuestro problema, al igual que se expone brevemente la historia de los algoritmos genéticos como la estrategia seleccionada de las que se explican a continuación.

* + - 1. Timetabling y métodos de resolución implementados

A pesar de la gran dificultad para la 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…ver Figura 1…

* **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 [[2]](#footnote-2).

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:

Figura 1. Técnicas en la búsqueda de soluciones a problemas de optimización



Fuente: Mejía & Arboleda [2]

* **Templado o recocido Simulado (*Simulated Annealing*).** El recocido simulado fue propuesto y estudiado en primera instancia por Metrópolis en el año de 1953[[3]](#footnote-3). Es un método heurístico que tiene más relación con la termodinámica (similar al proceso de enfriamiento del metal. Como es una variante de la búsqueda local, puede quedar atrapado prematuramente en un óptimo local[[4]](#footnote-4). 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”[[5]](#footnote-5).

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"[[6]](#footnote-6).

Gómez[[7]](#footnote-7) comenta que las principales desventajas se presentan por el tiempo computacional y la dificultad para ajustar adecuadamente los parámetros que controlan el algoritmo. Abramson[[8]](#footnote-8) presenta algunas aplicaciones de esta técnica.

* **Búsqueda Tabú (*Tabu Search*).** La meta heurística búsqueda Tabú es introducida y desarrollada por Fred Glover[[9]](#footnote-9) en 1989. 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”[[10]](#footnote-10).

Su funcionamiento se ve expuesto de la siguiente manera: “Una búsqueda con lista tabú o Tabu 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”[[11]](#footnote-11).

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. Glover y Laguna[[12]](#footnote-12) dicen: "…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…".  
Restrepo[[13]](#footnote-13) expresa que la memoria es representada mediante una lista tabú, la cual contiene para las 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 existen implementaciones de Suarez[[14]](#footnote-14) y Cardemil[[15]](#footnote-15).

* **Colonia de Hormigas (*Ant Colony*).** Una de la meta-heurística más empleada recientemente para enfrentar problemas de optimización, desde su inicio por Dorigo, Birattari y Stutzle[[16]](#footnote-16) en la primera mitad de la década de los 90’.

Cada hormiga en la colonia realiza inicialmente trayectorias aleatorias en búsqueda de su alimento, al hallarlo estudia la cantidad y la calidad según Gómez[[17]](#footnote-17), 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.[[18]](#footnote-18) 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[[19]](#footnote-19) 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 [[20]](#footnote-20) y [[21]](#footnote-21).

* **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[[22]](#footnote-22). Esta metodología se desarrolla mediante un proceso iterativo, dividido en dos fases principales, la fase de construcción y la fase de mejoramiento, explicado en [[23]](#footnote-23) 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 [[24]](#footnote-24) y [[25]](#footnote-25).

* **Algoritmos Genéticos (*Genetics Algorithms*).** Cabezas [[26]](#footnote-26) hace referencia a Díaz [[27]](#footnote-27) para presentar este método meta-heurístico:

Definición: 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 de 1859 (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.

* + - 1. Comparación de métodos meta-heurísticos

Del trabajo investigativo realizado con cada una de las metaheurísticas, se describen características, ventajas y desventajas y cantidad de soluciones que maneja cada método, se sintetizan en la siguiente tabla:

Tabla 1. Características de las técnicas meta-heurísticas estudiadas

|  |  |
| --- | --- |
| **Meta-Heurística** | Características |
| **Recocido Simulado**  ***(1983)*** | Facilidad de implementación, sin embargo, complejo para problemas muy grandes  Facilidad para combinar con otras técnicas, para obtener sistemas híbridos  Dependiendo de los parámetros, las soluciones que se van encontrando pueden ser poco estables.  Soluciones que maneja: 1 |
| **Búsqueda Tabú**  ***(1986)*** | Requiere solución inicial  Más complejo de implementar  Buenos resultados en poco tiempo  Soluciones que maneja: 1 |
| **Colonia de Hormigas**  **(1996)** | La más novedosa  Tiempo para encontrar solución de calidad es alto  Ofrece buenas soluciones  Soluciones que maneja: N, determinadas por la naturaleza del problema. |
| **GRASP**  ***(1989)*** | Adaptativa de acuerdo a las condiciones del problema  Requiere alto tiempo para hallar solución  Búsqueda aleatoria  Dificultad de adecuar los parámetros  Soluciones que maneja: 1 |
| **Algoritmos Genéticos**  ***(1975)*** | Implementación relativamente simple  No necesitan conocimientos específicos sobre el problema a resolver.  Altamente estudiados, documentados y con gran soporte.  Utilizan operadores probabilísticos, sin embargo recorren el espacio de soluciones en forma "más inteligentes" que la búsqueda aleatoria.  Útiles en casos donde no es necesario obtener una solución óptima al problema, sino que una buena solución aproximada sería suficiente.  Manejan 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. |

Fuente: Trabajo investigativo.

Chambers [[28]](#footnote-28) realizó un estudio comparativo entre las técnicas meta-heurísticas de recocido simulado, búsqueda tabú y algoritmos genéticos, teniendo en cuenta una serie de características a cada una se les asignó 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 próxima tabla se muestra de los resultados del estudio, dando como mejor calificado la técnica de algoritmos genéticos.

Tabla 2. Comparación y calificación de meta-heurísticas

|  |  |  |  |
| --- | --- | --- | --- |
| Característica | Algoritmos Genéticos | Búsqueda Tabú | Recocido Simulado |
| Simplicidad | 1 | 0.75 | 0.75 |
| Independencia | 1 | 1 | 1 |
| Coherencia | 1 | 0.75 | 1 |
| Efectividad | 1 | 1 | 0.75 |
| Eficacia | 0.75 | 0.75 | 0.5 |
| Eficiencia | 1 | 0.75 | 0.5 |
| Generalidad | 0.75 | 1 | 1 |
| Adaptabilidad | 1 | 1 | 0.75 |
| Robustez | 0.75 | 0.75 | 0.75 |
| Interactividad | 0.75 | 0.75 | 0.75 |
| Diversidad | 1 | 0.5 | 0.5 |
| Autonomía | 1 | 1 | 1 |
| Puntaje Final | 11 | 10 | 9.25 |

Fuente: Naupari y Rosales [22]

De los estudios anteriores se destaca que el algoritmo genético, posee una implementación relativamente simple, de tal manera que no se necesita mucho conocimiento del problema a resolver por lo adaptable que es. Cuenta con diversidad ya que maneja un conjunto de soluciones. Cabe destacar que al igual que las otras técnicas del estudio un AG es autónomo, es decir que al implementarlo en un programa puede realizar la búsqueda por cuenta propia, característica útil en el caso de de resolver nuestro problema, puesto que eliminaría gran parte del costo necesario en realizar un horario de clases para una institución educativa.

* + - 1. 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 Tolmos[[29]](#footnote-29), diseñó 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[[30]](#footnote-30).

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 [[31]](#footnote-31).

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[[32]](#footnote-32). 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 Holland 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[[33]](#footnote-33). 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.

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.

* + 1. Marco teórico

En la próximo segmento se profundiza sobre un algoritmo genético, se explica el proceso que sigue, los componentes y parámetros que lo conforman. También se explica la metodología a seguir en el desarrollo de este proyecto RUP.

* + - 1. Introducción a los Algoritmos Genéticos y su analogía con la naturaleza

La técnica de Algoritmos Genéticos se basa en los mecanismos de selección natural de la naturaleza, donde los individuos más aptos de una población son los que sobreviven, debido a que se adaptan a los cambios de su entorno. De tal manera que estos individuos tienen mayor probabilidad de reproducirse y generar una descendencia que contenga información de sus padres, que con el paso de generaciones esta descendencia posea mejor aptitud y adaptabilidad al medio en que sobreviven.

Los Algoritmos Genéticos pertenecen a la Computación Evolutiva, esta se dice que es una rama de la Inteligencia Artificial, que involucra problemas de optimización combinatoria y se inspira en los mecanismos de Evolución biológica. Un AG es una técnica que resuelve problemas de la siguiente manera: genera poblaciones con el paso de generaciones, en cada una selecciona los individuos más aptos para aplicar operadores de reproducción y mutación en los genes, para crear una nueva población y repetir el proceso.

En la naturaleza individuos que pertenecen a una población compiten entre ellos por recursos como alimento, agua, territorio y en algunos casos por una pareja. Los más adaptados a las circunstancias del medio, son los que tienen mayor probabilidad de tener descendencia y propagar sus genes con el paso de generaciones, probablemente producir descendientes con mejores aptitudes que sus antecesores y adaptarse mejor a las características su medio ambiente. Por su parte el AG maneja una población de soluciones del problema a resolver, es en ella donde se aprecia la diversidad que tiene un algoritmo genético, esta diversidad se obtiene al conocer las aptitudes del los individuos y diferencias entre sí. Más adelante se elige los individuos que van a evolucionar mediante un proceso de selección, para obtener unos hijos que serán los que conformarán la nueva población.

* + - 1. Componentes de un Algoritmo Genético

Es momento de describir los componentes que posee esta técnica seleccionada para resolver nuestro problema en las escuelas. Estos componentes son:

* Una **representación de las posibles soluciones** a la problemática. Para que un algoritmo pueda iniciar, se debe definir la estructura que cada individuo debe tener, recordando que cada individuo representa una posible solución al problema, es necesario determinar la forma del cromosoma compuesto de genes. Por lo tanto un cromosoma es la estructura que contiene toda la información de una solución, comúnmente en un AG un cromosoma es una cadena binaria (0s y 1s), sin embargo, esta cadena puede tener otra codificación [[34]](#footnote-34).
* Un procedimiento para **crear la población inicial** de posibles soluciones. Tradicionalmente la población inicial, es decir la población con que el AG va a empezar la búsqueda se inicializa de manera aleatoria, esto significa que debe existir un método para asignar valores a los genes en cada individuo de la población; por otra parte, últimamente también se ha utilizado técnicas para que no se realice aleatoriamente como el uso de redes neuronales; sin embargo, lo esencial es garantizar diversidad en la población [[35]](#footnote-35).
* Una **función de evaluación** que clasifique a los individuos por su aptitud. En un AG es imprescindible que exista esta función para determinar que tan apto es un individuo frente a la solución que se busca, por este motivo la función es dependiente del problema que se desea resolver y determina como el operador de selección actúa en escoger soluciones a evolucionar.
* Un **conjunto de operadores** de evolución que modifican los individuos en cada generación. Todo AG debe implementar un operador para seleccionar los individuos más aptos para reproducir; un operador de cruce, que permita combinar información genética de dos o más individuos; un operador de mutación, que pueda alterar uno o más genes para explorar otras zonas del espacio de búsqueda y un operador de reemplazo que determine que individuos pasan a la siguiente generación y cuáles no [[36]](#footnote-36).
* Una **conjunto de parámetros** que permitan configurar al AG para ejecutarse de una manera determinada. Parámetros como el tamaño de la población, probabilidad de cruce y de mutación, criterio de término; ayudan a generalizar el AG en base al problema que se intenta solucionar [[37]](#footnote-37).
  + - 1. Funcionamiento de un Algoritmo Genético

Tomando la definición de algoritmo de wordreference.com: un conjunto ordenado y finito de operaciones que permite hallar la solución de un problema; un algoritmo genético debe seguir una serie de subprocesos para su funcionamiento, en seguida se explican, tomando como referencia a [[38]](#footnote-38):

1. [*Inicio*] se configuran los parámetros y se genera la población aleatoria de *n* cromosomas (posibles soluciones).
2. [*Aptitud*] se evalúa la aptitud *f (x)* de cada cromosoma *x* de la población.
3. [*Prueba*] si la condición de término se satisface, se detiene el algoritmo, se devuelve la mejor solución de la población actual y se dirige al paso VI.
4. [*Nueva población*] se crea una nueva población mediante:
5. [*Selección*] se selecciona los cromosomas padres, de una población, según su aptitud (cuanto mejor es la aptitud, mayor es la probabilidad de ser seleccionado).
6. [*Emparejamiento*] con una probabilidad de emparejamiento, dos padres se emparejan para formar a dos nuevos descendientes (hijos). Si no se realiza emparejamiento alguno, los descendientes son la copia exacta de los padres.
7. [*Mutación*] con una probabilidad de mutación, el nuevo descendiente muta (en alguna posición de su cromosoma).
8. [*Sustituir*] la nueva población generada es aplicada para otra iteración del algoritmo.
9. [*Ciclo*] se regresa al paso II.
10. Fin del algoritmo.

Figura 2. Diagrama de flujo de un AG

INICIALIZACIÓN

Generación de N Cromosomas

PRUEBA

Se solucionó ?

APTITUD

Evaluación de fitness

SELECCIÓN

Aptitud - Probabilidad

EMPAREJAMIENTO

Formar nuevos individuos

MUTACIÓN

Mutar uno o varios genes

REEMPLAZO

Generación t + 1

Fuente: [26]

* + - 1. Codificación del genotipo

Como se ha señalado un AG está compuesto de una población, que contiene diversos individuos (cromosomas), estos están conformados por un número determinado de genes. En la Figura 3, se observa que un gen contiene datos valiosos para solución del problema, estos datos debe manejar una codificación, generalmente es binaria, sin embargo dependiendo de la naturaleza del problema se determina cual se implementará.

Figura 3. Individuo binario de un AG



Fuente: [26]

En [[39]](#footnote-39) se citan 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.
  + - 1. Operadores genéticos

En la sección 1.7.2.2 se han comentado los componentes de un AG, dentro de los cuales está un conjunto de operadores, ahora se detallan estos operadores genéticos que se aplican a la población de individuos en cada generación:

* **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 [[40]](#footnote-40) y [[41]](#footnote-41) dan a conocer algunos como: elitista, por estado estacionario, por torneo, escalada, entre otras.
* **Reproducción**, **Emparejamiento o Cruce** (Crossover): Cruzan los cromosomas de dos padres, para formar dos descendientes (hijos). Algunas variaciones son: cruce de n puntos, uniforme, segmentada, aritmético, etc. [[42]](#footnote-42) y [[43]](#footnote-43)
* **Mutación**: es encargada de modificar uno o más genes de un descendiente, para buscar un factor de diversificación. Se realiza de manera aleatoria, siguiendo a la probabilidad de mutación establecida. Según [[44]](#footnote-44) existen algunas técnicas como: mutación de bit, de gen, multibit, multigen, de intercambio, heurística.
* **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.
  + - 1. Parámetros de un AG

Al igual que los operadores genéticos, los parámetros que configuran un AG son esenciales para la búsqueda, estos pueden determinar la probabilidad de éxito de un algoritmo genético, a continuación se detallan los parámetros según Guerra y Pardo [[45]](#footnote-45):

* **Tamaño de la población**: es importante determinarlo conforme la naturaleza del problema y las variables que maneja, estamos tratando de la cantidad de individuos que conformará 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 evolución al realizar una búsqueda escaza y poco óptima. Por otro lado, si la población es excesiva, el algoritmo genético será excesivamente lento.
* **Probabilidad o porcentaje de cruce**: Determina con qué frecuencia se cruzan los individuos; si es 0% los hijos serán una copia 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
* **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. La mutación trata de impedir que la búsqueda caiga en óptimos locales, es conveniente que ocurra de vez en cuando; por el contrario, si ocurre continuamente, se convierte en una búsqueda aleatoria.
  + - 1. Metodología RUP

A continuación se determina la metodología a seguir en el desarrollo del proyecto, esta se conoce como RUP (Rational Unified Process) un proceso de ingeniería de software. Se compone de 4 fases.

Según describe Kruchten [[46]](#footnote-46) 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.

**Fases:** Martinez y Martinez [[47]](#footnote-47) 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

Figura 4. Diagrama Metodología RUP



Fuente: Martinez, Raul & Martinez, Alejandro [29]

* + 1. Marco conceptual

A continuación se definen términos que en el transcurso del documento se hacen referencia, de este modo, se muestra las definiciones y explicaciones técnicas acerca del timetabling como un problema de alta complejidad que pertenece al grupo de NP – Completos, debido a los recursos a asignar y las restricciones que estos deben cumplir.

* + - 1. 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:

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”[[48]](#footnote-48). Anterior a ellos Anthony Wren en 1996, determina el Timetabling, como un caso especial de Programación (scheduling), a esta la define como: “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”[[49]](#footnote-49)

Lance D. Chambers [[50]](#footnote-50) toma la definición de una forma más general: “El Timetabling 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 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), en [[51]](#footnote-51) se enuncian estos:

* Transport Timetabling
* Sports Timetabling
* Employee Timetabling and Rostering
* Educational Timetabling
  + - 1. Educational Timetabling

De esta rama del timetabling del ámbito educativo, los principales problemas son los de programación de horarios tanto en colegios (School Timetabling) como en universidades (University or Course Timetabling), 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. Otro campo subsecuente de la educación tiene que ver con la carga de exámenes y su calendarización (Exam o Examination Timetabling).

Dentro del universo Educational Timetabling se encuentran dos variaciones generales, como lo son la asignación de horarios escolares y la asignación de horarios universitarios los cuales difieren en ciertos aspectos que en [[52]](#footnote-52) 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 cursos, considerando un número de salas y bloques de tiempo.”

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. Existen otras medidas que difieren entre el School y el University Timetabling:

Tabla 3. Comparación entre la asignación de horarios escolares y universitarios

|  |  |  |
| --- | --- | --- |
| 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 [34]

* + - 1. Clasificación de restricciones

Dentro del problema de creación de horarios académicos, se deben manejar diferentes restricciones para una correcta asignación, estas se dividen en dos grupos, Larrosa [[53]](#footnote-53) 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 en una clase de dos periodos consecutivos). La violación de alguna seguirá ocasionando un horario factible, pero no de la calidad deseada.
  + - 1. Complejidad computacional

En la búsqueda de soluciones a una gran cantidad de problemas en el área de la computación, se nota 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 tres clases principales: P, NP y NP-COMPLETOS.[[54]](#footnote-54)

* Clase 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 situé en P hace parte de los problemas en NP.
* Clase 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; 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.

* Clase 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[[55]](#footnote-55).

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, que no están presentes dentro de 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.

Figura 5. Complejidad Computacional



Fuente: Elaboración propia.

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 [[56]](#footnote-56) [[57]](#footnote-57) [[58]](#footnote-58), por su gran dificultad de resolución.

* 1. Factibilidad
     1. Factibilidad técnica

La factibilidad técnica consistió en realizar una evaluación de la tecnología existente del grupo de desarrollo, este estudio estuvo destinado a recolectar información sobre los componentes técnicos que se poseen y la posibilidad de hacer uso de ellos en el desarrollo e implementación de la aplicación propuesta. De acuerdo a la tecnología necesaria para la implantación del proyecto se evaluó dos enfoques: hardware y software.

* + - 1. Hardware

El hardware a considerar son los computadores personales de los integrantes del proyecto, es decir que, para la implantación y ejecución del aplicativo no se tendrán mayores inconvenientes por poseer los computadores físicamente accesibles en cualquier momento; no se requiere realizar una inversión inicial para la adquisición de equipos nuevos, ni tampoco repotenciar o actualizar los equipos existentes; en seguida se detalla la descripción del hardware disponible que fue utilizado para el diseño, construcción y puesta en marcha de la aplicación. Además hay que agregar que estos componentes se encuentran en el mercado actualmente a precios bajos.

Tabla 4. Características técnicas del hardware existente

|  |  |  |
| --- | --- | --- |
| **PARTE** | **PRINCIPAL** | **SECUNDARIO** |
| **MEMORIA RAM** | 2 GB | 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 |
| **TARJETA DE VIDEO** | NVIDIA GeForce 9500 GT 512 MB – DDR3 | INTEL HD Graphics 3000 - 1.5 GB |
| **TARJETA DE RED** | * Marvell Yukon 88E8056 PCI-E Gigabit Ethernet Controller. * 802.11n Network Adapter. | * Controladora Realtek PCIe FE Family * Ralink RT5390R 802.11 bgn Wi-Fi Adapter |

Fuente: Trabajo investigativo

Teniendo en cuenta que la aplicación empleará una base de datos se debe determinar la forma de conectar tanto el equipo donde se ejecuta la aplicación, como el equipo donde se encuentra almacenada la base de datos, inicialmente esta base de datos se almacena localmente en el mismo equipo por lo que no se necesita de un componente adicional para un correcto funcionamiento del aplicativo, sin embargo para la implantación del sistema, la base de datos puede estar alojada en otro equipo, es decir que ésta debe ser accedida de forma remota, para ello es necesario tener una conexión a internet mediante una tarjeta o adaptador de red que los dos equipos mencionado la poseen.

* + - 1. Software

El sistema operativo que viene instalado por defecto en los computadores mencionados es Windows 7 y Windows 8 respectivamente, por lo que no amerita inversión alguna, sin embargo la aplicación a desarrollar tiene la característica de poder ser ejecutada en diferentes plataformas de sistema operativo; al igual que portable, de esta manera se puede portar en un dispositivo de almacenamiento como un CD e incluso USB flash drive (memoria USB) y ser ejecutado en otro computador sin mayor dificultad.

Técnicamente la creación del algoritmo genético es posible, ya que la plataforma a desarrollar es JAVA, lenguaje de programación de propósito general con la característica de tener pocas dependencias de implementación y licencia GNU GPL de software libre. Para el manejo de los datos y la persistencia de los mismos se decide utiliza el sistema gestor de base de datos (SGBD) PostgreSQL, que también es de código abierto, dirigido por una comunidad con lanzamientos y actualizaciones continuas; además, mediante una librería se acopla a JAVA fácilmente. Por lo tanto el software requerido para el desarrollo no necesita algún tipo de inversión debido a las licencias que poseen, solo la instalación y configuración para que la aplicación funcione correctamente.

Como resultado de este estudio se determinó que en los actuales momentos, el grupo de desarrollo cuenta con la infraestructura tecnológica (Hardware y Software) necesaria para la creación, construcción y puesta en funcionamiento del aplicativo propuesto.

* + 1. Factibilidad operativa

La factibilidad operativa permite predecir, si se pondrá en marcha la aplicación propuesta, aprovechando los beneficios que ofrece a todos los usuarios involucrados con el mismo, ya sean los que interactúan en forma directa con este, como aquellos que reciben información producida por el sistema.

La posibilidad y el deseo de cambio en la manera que se realiza la creación de un horario de clase hacia un proceso más tecnológico, sistemático y automatizado, conlleva a la aceptación de los usuarios involucrados, donde se tiene una interfaz sencilla y los datos transmitan información clara, útil y oportuna, trae beneficios a los usuarios de la aplicación. Basándose en entrevistas y conversaciones sostenidas con el personal involucrado (coordinadores académicos, rectores, profesores) se demostró que estos no demuestran una oposición al cambio, de tal manera que el proyecto es factible operacionalmente.

Con el objetivo de garantizar el correcto funcionamiento de la aplicación y con la finalidad de que el impacto sea positivo a los usuarios, fue desarrollado en diversos componentes siguiendo un proceso con formularios y reportes que son familiares en términos y aspectos académicos que se tratan una institución educativa, de manera que el adiestramiento sea sencillo contando con la opinión de los usuarios implicados para cualquier modificación de la aplicación.

* + 1. Factibilidad económica

A continuación se describe el estudio que arroja como resultado la factibilidad de la aplicación de escritorio para la creación de un horario de clases para una institución determinada. Se determinaron los recursos para desarrollar, implantar y mantener en operación el sistema propuesto, realizando una evaluación que existe de los costos intrínsecos del aplicativo y los beneficios que se derivan de este, de tal manera se logró apreciar las bondades del sistema programado.

* + - 1. Análisis Costos – Beneficios.

Este análisis permitió realizar una comparación entre los costos del nuevo sistema, teniendo en cuenta los costos que tiene la realización del horario de forma manual y los beneficios que se obtienen al trasladar esta labor a un nivel automatizado.

En la factibilidad técnica se indicó que el grupo de desarrollo contaba con las herramientas necesarias para la puesta en marcha del proyecto, de tal manera que no se requiere una inversión inicial, sin embargo, durante el desarrollo de la aplicación se puede generar un costo imprevisto por ejemplo, costes derivados de la curva de aprendizaje por parte del personal involucrado.

En seguida se presenta una estimación de los costos de desarrollo, costos de operación y gastos del sistema; luego se determinan los beneficios que no necesariamente para la aplicación son monetarios o cuantificables.

* **Costos de desarrollo**
* Personal

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Cargo | Horas Semanales | Costo por hora | Costo Semanal | Costo Mensual | Total (17 meses) |
| Tutor del proyecto | 3 | $ 40.000 | $ 120.000 | $ 480.000 | $ 8.160.000 |
| Analista / Diseñador | 40 | $ 12.000 | $ 480.000 | $ 1.920.000 | $ 32.640.000 |
| Programador | 48 | $ 8.000 | $ 384.000 | $ 1.536.000 | $ 26.112.000 |
|  | | | | Total | $ 66.912.000 |

* Hardware y Software

|  |  |  |
| --- | --- | --- |
| Uso Informático (Internet y Energía) | 4.800 horas a $ 1.500 / hora | $ 7.200.000 |

* **Gastos**
* Transporte

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Cantidad | Descripción | Costo Diario | Costo Semanal | Costo Mensual | Costo (Anual) |
| 2 | (3 veces / semana) | $ 3.200 | $ 19.200 | $ 76.800 | $ 921.600 |

* **Costo operacional (anual)**
* Personal

|  |  |  |  |
| --- | --- | --- | --- |
| Cargo | Horas Anuales | Costo por hora | Total (anual) |
| Analista / Mantenimiento | 48 | $ 10.000 | $ 480.000 |

* Software

|  |  |  |
| --- | --- | --- |
| Uso Informático | 72 horas a $ 1.500 / hora | $ 108.000 |

* Insumos

|  |  |  |  |
| --- | --- | --- | --- |
| Cantidad | Descripción | Costo | Total |
| 1 | Resma de Papel A4 | $ 15.000 | $ 15.000 |
| 2 | Cartucho de impresora | $ 30.000 | $ 60.000 |

* **Beneficios**

Los beneficios de la aplicación propuesta están orientados a mejorar la calidad del horario de clases mediante el cumplimiento de restricciones y la velocidad en la realización de dicho horario, de esta manera la *Implementación de Algoritmos Genéticos para la Resolución del problema School Timetabling en las Instituciones Educativas*, producirá diversos beneficios para las instituciones en que se aplique, estos se clasifican en dos tipos:

* **Beneficios Tangibles:** los beneficios aportados por la aplicación propuesta que se pueden comprobar en términos de tiempo y economía.  
    
  Dentro de estos beneficios se encuentra una reducción del tiempo empleado en actividades involucradas en la realización e implementación de un horario de clases, por ejemplo:
  + Obtener información de un grado (cantidad de cursos, número de estudiantes por curso).
  + Obtener la carga académica de un determinado grado (asignaturas, intensidad horaria semanal).
  + Obtener la lista de docentes que conforman la institución y sus respectivos datos (nombre, horas semanales contratadas y periodos de disponibilidad).
  + Obtener el horario de clases de un determinado curso / profesor.
  + Obtener y generar un informe de los profesores que se encuentran libres en un determinado intervalo de periodos de tiempo.

Actividades de las cuales en el programa de escritorio se realizan en menos de 1 minuto cada una aproximadamente, lo que beneficia a tener la información más rápida que de forma manual como lo realizan muchas instituciones, de esta manera si se compara estos dos sistemas se logra una disminución no solo en tiempo sino en productividad, por ejemplo:

Tabla 5. Ejemplo beneficios tangibles

|  |  |  |
| --- | --- | --- |
| Actividad: *Conocer y generar un informe de la cantidad de lecciones impartidas de todas las asignatura en todos los cursos de la institución*. | | |
| Tiempo en forma manual:  *Entre 5 a 30 minutos*. | Tiempo en aplicativo:  *2 minutos aprox*. | Beneficios tangibles:   * Reducción del tiempo empleado. * Disminución de errores. * Incremento en la productividad (Se pueden realizar otras actividades) |

Otro aspecto que va unido con la reducción del tiempo de las actividades, es un ahorro a nivel económico tanto en la disminución del costo en la realización de actividades, como en los recursos que se emplean en estas labores. Existe un ahorro de energía ya que algunas instituciones requieren utilizar personal extra para la creación del horario de clases y la aplicación propuesta simplifica este costo para trasladarlo a un nivel más sistematizado y económico mediante disminución en papelería, con el programa de escritorio se disminuye los formatos en papel para representar información determinada y se reemplazan por datos electrónicos almacenados en computadores.

* **Beneficios Intangibles:** de otro lado se encuentran estos beneficios que aunque no representan un resultado medible o cuantificable son determinantes en aportar funcionalidades y optimización de los recursos involucrados.
  + Generar más de un horario de clases de manera autónoma, después de ingresar todos los datos involucrados en la creación del mismo para ser implementado en la institución, lo que beneficia y evita realizar esta labor de forma manual por parte del personal encargado.
  + La flexibilidad al manejar un volumen de datos con rapidez y precisión que brinda esta herramienta al personal, optimizará sus labores.
  + Generación de reportes que aporten información más eficiente y confiable que sirva de apoyo a una toma de decisiones, minimiza el esfuerzo lo que propicia con el mejor aprovechamiento de los recursos.
  + Capacidad de búsqueda y actualización de datos de manera práctica, más rápido que de forma manual y reduciendo la fuerza de trabajo.
    - 1. Relación Costo – Beneficio.

El análisis costo – beneficio expone grandes ventajas tanto para el desarrollo como para la implantación del proyecto, si le agregamos que el programa pueda solucionar el problema de la calendarización para diversas instituciones es de gran beneficio para el personal delegado a elaborar el horario de calases del centro educativo.

Asimismo la aplicación conlleva a mejoras significativas para la institución que se beneficie del software, reduciendo el tiempo por medio de un procesamiento autónomo y con la generación de información valiosa como determinados reportes, de esta manera se reduce las cargas de trabajo a los usuarios al aumentar la velocidad en el procesamiento, ofreciendo buenos resultados en menor tiempo.

El beneficio más significativo que se adjudicaría al implantar el programa automatizado, es la característica de generar información de ciertos datos ingresados, transformándose en una herramienta computacional versátil, útil y poderosa, capaz de dar solución al problema del school timetabling.

Cabe destacar que tener al alcance una herramienta que genere información a una institución u organización, determinaría un ahorro de tiempo y dinero, de tal forma que sea el punto de apoyo para mejorar y agilizar la toma de decisiones, y los procesos se conviertan en mejores resultados para la institución, representa un aumento en la productividad.

De este modo reduciendo el empleo de recursos humanos y materiales, disminuyendo actividades redundantes, optimizando los procesos que involucra la realización de un horario de clases para determinada institución educativa, trasladando el procesamiento manual de los datos a automático conlleva a la *Implementación de Algoritmos Genéticos para la Resolución del problema School Timetabling en las Instituciones Educativa,* como la alternativa a mejorar el sistema manual y cumplir los objetivos de la investigación.

* + 1. Factibilidad legal

Para el desarrollo del proyecto se ha utilizado dos computadores especificados anteriormente en la factibilidad técnica, estos ordenadores utilizan el sistema operativo Windows, es importante enunciar que éste opera bajo una licencia de software propietario por la cual el uso del producto sólo está permitido para un único usuario; bajo este criterio, para la implantación de la aplicación de escritorio es indispensable que no viole o atente contra alguna ley o reglamento.

El desarrollo del software se realizará en la plataforma y lenguaje de programación JAVA, que está licenciado como software libre bajo GNU GPL con las libertades de descarga, copia y distribución de la tecnología siempre que se cumpla con los términos y condiciones que estipulan; por tanto legalmente el requerimiento para la descarga e instalación de la tecnología es ser propietario del equipo que en nuestro caso se cumple sin inconvenientes.

Dentro de las características de desarrollar bajo este lenguaje, es que el software funcional puede operar en múltiples plataformas, en este sentido el sistema operativo Windows no debe ser obligatorio para la implantación de la aplicación ya que puede ser sustituido por otro sistema que este licenciado con más libertades como es caso del software libre.

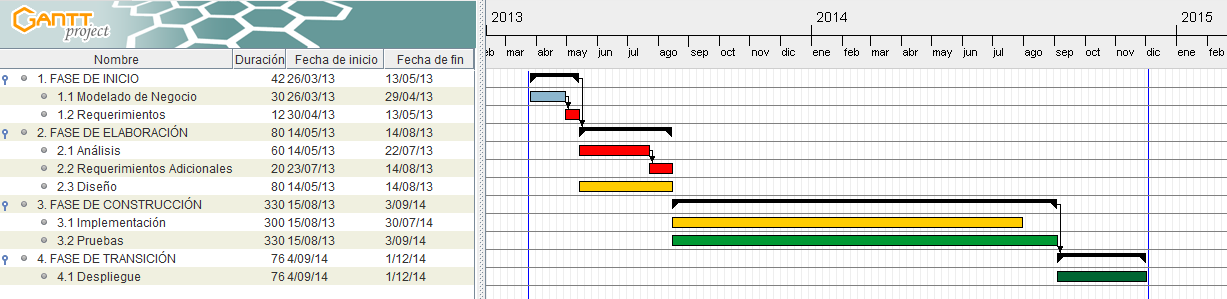
Como herramientas para el desarrollo se utilizará el IDE Netbeans, se puede descargar sin ningún costo desde su página web; funciona bajo un licenciamiento dual de Licencia CDDL de código abierto y GPL2 de software libre, de tal manera que garantiza la libertad de uso. Otra herramienta a implementar es PostgreSQL que opera bajo la licencia BSD, también de software libre y de código abierto, se descarga desde su página web sin costo.

En resumen se tiene las licencias de Windows, libertad de instalar Java para el desarrollo, descarga e instalación de las herramientas mencionadas compatibles entre sí, de tal forma que tanto el desarrollo como la implantación del proyecto es factible en términos legales, teniendo en cuenta que el proyecto es únicamente desarrollado con fines investigativos y no lucrativos.

* 1. Cronograma de actividades

A continuación se ilustra el cronograma de actividades, las cuales deben realizarse para el desarrollo exitoso del proyecto. El siguiente cronograma se ha adecuado a la metodología propuesta en el marco teórico, para seguir un correcto flujo de trabajo y cumplir con los objetivos y plazos establecidos.

Figura 6. Diagrama Gantt de actividades



1. Fase de inicio

En el siguiente capítulo se exponen dos flujos de trabajo de la metodología RUP, el modelado del negocio, describiendo el proceso que se realiza para la creación de horarios académicos en el colegio María Mercedes Carranza IED. En la segunda parte de este capítulo se listan y la clasifican los requisitos a tener en cuenta para el análisis y diseño posterior del proyecto.

* 1. Modelado del negocio

El modelado del negocio es un proceso de la metodología RUP que se realiza para comprender la estructura y la dinámica de la organización; en este caso cómo funciona el negocio que se desea automatizar para garantizar que el software ya desarrollado cumpla con su propósito; de esta manera, se debe realizar un estudio del dominio del negocio para determinar elementos que intervienen.

Una vez definido en qué consiste es te proceso, el siguiente paso es describir el procedimiento que se sigue para la elaboración de horarios académicos, es aquí donde tomamos como guía el colegio María Mercedes Carranza IED. Este proceso se realiza una vez al año, específicamente en los meses de diciembre y enero cuando se está finalizando un ciclo académico y en periodos de vacaciones, ya que en este lapso de días se recopila la información requerida para la elaboración de un horario para el siguiente ciclo; datos como el número de cursos habilitados para cada grado, las asignaturas de dictado, la plana de docentes y las aulas disponibles son los recursos principales para la creación del horario de clases.

El presente trabajo se refiere a la elaboración de horarios para el ciclo académico del año 2014, de los grados de sexto a undécimo que conforman la educación básica secundaria y educación media (6 grados de bachillerato); cada uno de estos grados está compuesto por varios grupos de estudiantes a los que se conoce en los colegios y otras instituciones educativas como cursos. Por ejemplo en esta institución la apertura de cursos está condicionada por la Secretaria de Educación del Distrito, quien se encarga de los cupos académicos y matriculas para los estudiantes que deseen acceder a la educación (básica secundaria y media) en la ciudad de Bogotá.

Con relación a las asignaturas de dictado, estas materias pertenecen a un área general (por ejemplo, el área de ciencias básicas) habitualmente son estimadas para cada área o asignaturas en específico una intensidad horaria semanal (IHS), que también la determina la Secretaria de Educación del Distrito siguiendo indicaciones del Ministerio de Educación. Una vez que el colegio posee la información de la IHS para las asignaturas, decide la cantidad de lecciones y cómo distribuirlas a lo largo de la semana para los diferentes grados (bloques de clases o una lección, generalmente), este proceso se conoce como definir la carga académica para el nuevo año escolar.

Con relación a los docentes, éstos pueden dictar un conjunto de asignaturas para uno o varios grados de acuerdo a su ámbito de enseñanza y a su preferencia, además, cada profesor posee una disponibilidad horaria que puede variar en cada año académico. Esta información es recopilada por el Coordinador Académico meses antes del inicio de cada año, quien se encarga de enviar un formato para ser llenado por cada docente.

Con respecto a las aulas, estas cuentan con una disponibilidad total durante toda la semana académica y pueden pertenecer tanto a un curso (grupo de estudiantes), como ser asignadas a un profesor para el periodo escolar. Esto determina si son los estudiantes o los docentes, quienes deben trasladarse de aulas cada cambio de clases.

Al recopilar toda la información necesaria, el Coordinador Académico puede iniciar el proceso de elaboración de horarios académicos, el cual es creado por un número mayor de personas y puede demorar una semana o más en su realización por ser una actividad manual. Una vez el horario es creado, se notifica a los profesores para finalmente publicar el horario de clases al alumnado del nuevo año académico.

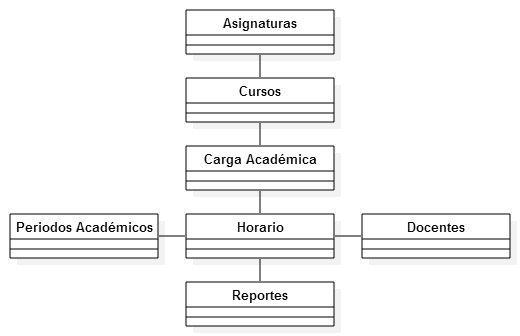
* + 1. Modelo del dominio

El modelo del dominio “captura los tipos más importantes de objetos en el contexto del sistema. Los objetos del dominio representan ‘cosas’ que existen o los eventos que suceden en el entorno en el que trabaja el sistema”[[59]](#footnote-59).

La elaboración de horarios académicos la determina la carga académica para el año escolar, esta carga está compuesta por las asignaturas a dictar al conjunto de cursos de la institución, y la plana de docentes encargados de las lecciones de clase. Esta realización genera reportes y estadísticas de la información académica del colegio.

Los objetos del dominio que se obtuvieron y sus relaciones se encuentran en siguiente diagrama de clases.

Figura 7. Modelo del dominio



Fuente: Pardo ERWIN, Guerra MAURICIO

* 1. Requerimientos

Los requerimientos del sistema fueron determinados en primer lugar utilizando una investigación de antecedentes del problema del school timetabling y por otro lado se recurre al uso de entrevistas con el coordinador académico del colegio María Mercedes I.E.D. donde se realizará la prueba piloto. Como resultado se notó que un requerimiento del sistema era la división funcional por medio de módulos, de tal manera que uno de estos se encargara de los datos necesarios para crear un horario y otro módulo para la gestión de los horarios generados de forma autónoma por la técnica de algoritmos genéticos.

A continuación se exponen los requerimientos en una lista categorizada según su ámbito e influencia entorno a la aplicación del proyecto. Esta lista se ha elaborado siguiendo las recomendaciones del estándar IEEE 830-1998.

* + 1. Requisitos de la interfaz externa
       1. Interfaz con el usuario

La aplicación debe ser intuitiva y predictiva con el usuario ya que lo podrá utilizar cualquier persona que sepa operar un equipo de cómputo. El manejo de las funcionalidades de la aplicación debe ser lo más intuitiva posible, de tal forma que las posibles acciones de llevar acabo sean claras.

* + - 1. Interfaz con el hardware

El programa requiere una salida de impresora, para lograr imprimir los reportes que se puedan generar. La interacción con la aplicación se realiza mediante el mouse y el teclado para lograr ingresar los datos requeridos, así como una tarjeta de red para poder trabajar en red.

* + - 1. Interfaz con el software

La aplicación de escritorio implantada en Java, siempre y cuando en la plataforma sobre la que esté instalado (Windows XP, Windows vista, Windows 7, Windows 8, Linux,…) contenga la máquina virtual de Java; también debe interactuar con la base de datos Postgresql por jdbc.

* + 1. Requisitos funcionales

A continuación se exponen los requerimientos funcionales de la aplicación de escritorio, estos se han modelado en diagramas de caso de uso para una mejor comprensión y visualización; más adelante se muestra la documentación de cada caso expresado.

* + - 1. Diagramas de caso de uso del negocio

Figura 8. Diagrama de Caso de Uso del Negocio 1

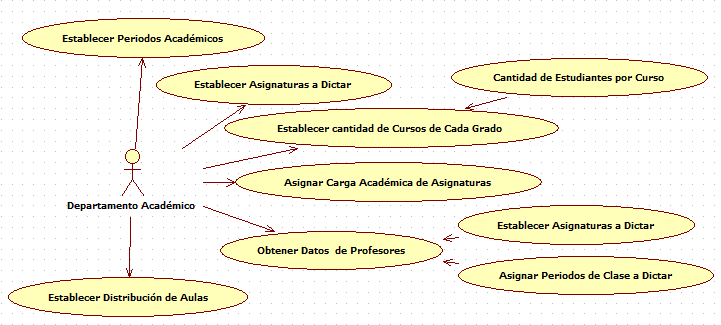


Tabla 6. Documentación caso de uso del negocio 1

|  |  |
| --- | --- |
| ID | No. 1 |
| Descripción | El departamento académico del colegio recopila y organiza toda la información necesaria para la realización y generación del horario de clases. |
| Actores | Departamento Académico |

Figura 9. Diagrama de Caso de Uso del Negocio 2

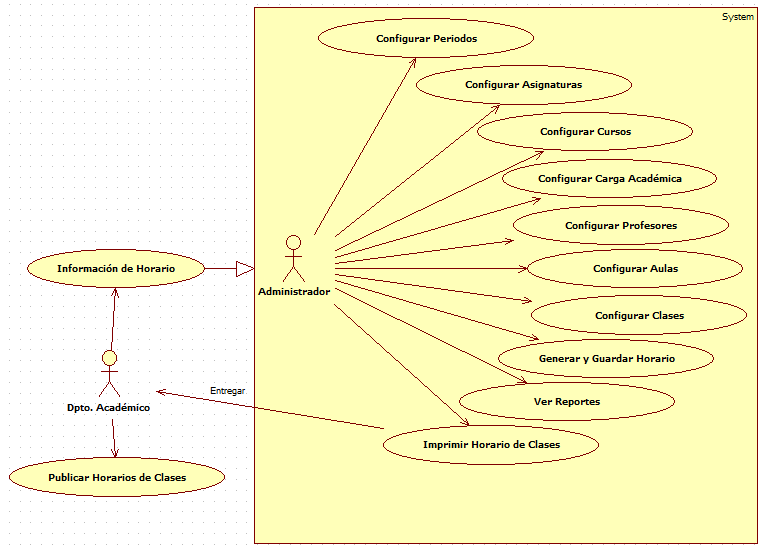


Tabla 7. Documentación caso de uso del negocio 2

|  |  |
| --- | --- |
| ID | No. 2 |
| Descripción | El departamento académico del colegio brinda al administrador del sistema la información obtenida anteriormente.  El administrador del sistema se encarga de ingresar toda esa información y finalmente de generar el horario de clases y de este modo entregarle al departamento académico el horario impreso para que este se encargue de publicarlo para toda la comunidad educativa. |
| Actores | Departamento Académico, Administrador del sistema |

* + 1. Requerimientos de desarrollo y restricciones de diseño.

Como requerimientos planteados por el equipo de desarrollo se encuentran los siguientes:

El aplicativo se debe desarrollar en el entorno IDE Netbeans en la versión 7.2.1 que se encuentra disponible en su página web. Otro requisito es el uso del sistema de gestión de base datos Postgresql en la versión 9.2 para el manejo de base de datos, este también se puede descargar de forma gratuita desde la página web y permitirá almacenar los datos que se ingresan el programa. Como requisito opcional es la posibilidad de tener acceder de forma remota con el servicio de base de datos SQL heroku-postgres, que se acopla sin inconvenientes mediante un driver jdbc a Java.

Adicionalmente para la persistencia de los datos se debe utilizar Eclipselink versión 2.3.2, que junto con el driver jdbc permite enlazar la aplicación con la base de datos que permita manejar los datos como objetos.

El diseño de la interfaz gráfica se realizará con la biblioteca Swing para Java, para tener una independencia y menos limitaciones de la plataforma del sistema operativo. Otro requisito de software para la creación y generación de los reportes, en nuestro proyecto el uso de la librería JasperReports en la versión 4.7.0.

Finalizando estos requerimientos de desarrollo, para la creación de este documento se implementará el paquete de Microsoft Office en las versiones 12.0 y 14.0 para Windows. Y para el modelado de diagramas se decide utilizar StarUML en la versión 1.5.0.

* + 1. Atributos del sistema.

Los requerimientos no funcionales o atributos del sistema se exponen en la tabla

Tabla 8. Requerimientos no funcionales (atributos del sistema)

|  |  |
| --- | --- |
| **CRITERIO** | **REQUERIMIENTO** |
| Desempeño | El tiempo de respuesta durante la inserción, modificación o eliminación de registros y operaciones generales no debe ser superior a 2.5 segundos.  Ahora bien, la ejecución del algoritmo genético para obtener la solución tiene un tiempo de respuesta dependiente del tamaño del problema. |
| Escalabilidad | El software permite el desarrollo e integración de nuevas funcionalidades a nivel de código que permitan el mejor desempeño del sistema.  Ej: Carga masiva de información, roles de acceso al sistema, autenticación de acceso al sistema y a la información. |
| Flexibilidad | La ejecución de la aplicación no depende del sistema operativo, de la red o de algún parámetro externo, es decir, funciona en múltiples plataformas sin presentar fallos. |
| Mantenibilidad | Inicialmente, se brinda un manual de usuario final, en el cual se especifica el uso del sistema y un manual de instalación.  En caso de errores internos del software, se debe entrar directamente a revisar el código fuente. |
| Fiabilidad | Para la puesta en marcha del sistema se hacen pruebas que verifiquen que las funcionalidades del sistema se ejecuten de manera correcta en cada uno de los casos que se puedan presentar, esto para evitar la mayor cantidad de fallos del software y mejorar la calidad de este. |
| Disponibilidad | Se debe esperar una disponibilidad del software del 98%, ya que no depende de una conexión de red que pueda presentar errores, sin embargo, se tiene en cuenta la conexión con los datos de la BD para el uso de la información. |
| Portabilidad | El sistema es multiplataforma, por lo cual se puede ejecutar en diversos sistemas operativos ya sean de 32 o 64 bits. |
| Seguridad | El acceso al sistema se limita al manejo de un administrador, es decir, existe un único rol para el manejo de un usuario final. |

* + 1. Otros requisitos.

Se requiere personal humano capacitado con conocimientos avanzados en el lenguaje de programación Java y SQL, al igual que conocimientos de las herramientas utilizadas para el desarrollo del software (IDE Netbeans), el manejo de base de datos con el motor Postgresql. De igual manera, es preciso contar con una o varias personas con conocimientos en school timetabling e inteligencia artificial con la implementación de algoritmos genéticos.

De esta manera para el desarrollo de este proyecto se cuenta con el siguiente recurso humano:

* **Inicio, elaboración, construcción y transición del proyecto**
  + Mauricio Andrés Guerra Cubillos
  + Erwin Hamid Pardo Quiroga
* **Tutor**
  + Roberto Emilio Salas Ruiz
    1. Flujos de trabajo

Un flujo de trabajo es una secuencia de actividades que producen un resultado de valor observable; “su origen parte de identificar los trabajadores que participan en el proceso y a continuación los artefactos que se necesitan crear durante el proceso para cada tipo de trabajador”[[60]](#footnote-60), de esta manera, se pueden identificar las actividades de los actores que participan en la aplicación.

A continuación se presentan los principales flujos de trabajo propuestos por medio de la representación gráfica de diagramas de actividad.

* Diagrama de actividad No. 1: Configurar Días

Figura 10. Diagrama de actividad - Configurar Días



* Diagrama de actividad No. 2: Configurar cambios de Clase

Figura 11. Diagrama de actividad - Configurar cambios de Clase



* Diagrama de actividad No. 3: Registrar Asignatura

Figura 12. Diagrama de actividad - Registrar Asignatura



* Diagrama de actividad No. 4: Editar Asignatura

Figura 13. Diagrama de actividad - Editar Asignatura



* Diagrama de actividad No. 5: Eliminar Asignatura

Figura 14. Diagrama de actividad - Eliminar Asignatura



* Diagrama de actividad No. 6: Registrar Grado

Figura 15. Diagrama de actividad - Registrar Grado



* Diagrama de actividad No. 7: Editar Grado

Figura 16. Diagrama de actividad - Editar Grado



* Diagrama de actividad No. 8: Eliminar Grado

Figura 17. Diagrama de actividad - Eliminar Grado



* Diagrama de actividad No. 9: Registrar Nuevo Curso

Figura 18. Diagrama de actividad - Registrar Nuevo Curso



* Diagrama de actividad No. 10: Editar Curso

Figura 19. Diagrama de actividad - Editar Curso



* Diagrama de actividad No. 11: Eliminar Curso

Figura 20. Diagrama de actividad - Eliminar Curso



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

Figura 21. Diagrama de actividad - Registrar Nueva Carga Académica



* Diagrama de actividad No. 13: Editar Carga Académica

Figura 22. Diagrama de actividad - Editar Carga Académica



* Diagrama de actividad No. 14: Eliminar Carga Académica

Figura 23. Diagrama de actividad - Eliminar Carga Académica



* Diagrama de actividad No. 15: Registrar Nuevo Profesor

Figura 24. Diagrama de actividad - Registrar Nuevo Profesor



* Diagrama de actividad No. 16: Editar Profesor

Figura 25. Diagrama de actividad - Editar Profesor



* Diagrama de actividad No. 17: Eliminar Profesor

Figura 26. Diagrama de actividad - Eliminar Profesor



* Diagrama de actividad No. 18: Configurar Preferencia de Periodos de Profesor

Figura 27. Diagrama de actividad - Configurar Preferencia de Periodos de Profesor



* Diagrama de actividad No. 19: Configurar Preferencia de Asignaturas de Profesor

Figura 28. Diagrama de actividad - Configurar Preferencia de Asignaturas de Profesor



* Diagrama de actividad No. 20: Registrar Aula

Figura 29. Diagrama de actividad - Registrar Aula



* Diagrama de actividad No. 21: Editar Aula

Figura 30. Diagrama de actividad - Editar Aula



* Diagrama de actividad No. 22: Eliminar Aula

Figura 31. Diagrama de actividad - Eliminar Aula



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

Figura 32. Diagrama de actividad - Crear Configuración de Clases



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

Figura 33. Diagrama de actividad - Agregar Clase a Configuración Existente



* Diagrama de actividad No. 25: Editar Clase a Configuración Existente

Figura 34. Diagrama de actividad - Editar Clase a Configuración Existente



* Diagrama de actividad No. 26: Eliminar Clase a Configuración Existente

Figura 35. Diagrama de actividad - Eliminar Clase a Configuración Existente



* Diagrama de actividad No. 27: Generar Horario

Figura 36. Diagrama de actividad - Generar Horario



1. fase de elaboración
   1. análisis

Este flujo de trabajo se basa en un modelo de objetos conceptual, que ayuda a refinar los requisitos logrando una comprensión más precisa de estos y razonando sobre los aspectos internos del sistema. Este análisis es fundamental para el Diseño y la Implementación posteriores, por lo que provee una visión general.

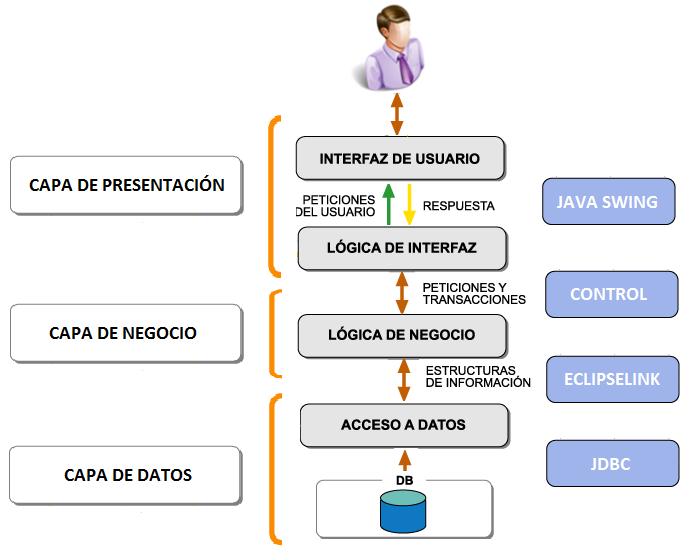
* + 1. Modelos de Casos de Uso - Análisis

Los modelos de caso de uso durante la fase de análisis describen todos los procedimientos que se realizan en el sistema. Estos modelos se encuentran en el Anexo A. Modelos de caso de uso.

* + 1. Arquitectura de software

La arquitectura del software representa la interacción entre los diferentes niveles o capas de la aplicación, mostrando así un flujo de eventos mediante los cuales se visualiza el funcionamiento interno y externo del software.

Figura 37. Arquitectura de software



Este proyecto está realizado bajo el patrón MVC y su implementación en Java Swing, por lo cual el análisis de la arquitectura del software se basa en dicho patrón y a continuación se describe cada uno de los niveles y sus componentes.

* **Capa de Presentación:** Es la capa o nivel que interactúa directamente con el usuario, en el caso de este proyecto con la vista de componentes de *Java Swing*. Esta capa de presentación contiene la interfaz gráfica que recibe las peticiones enviadas por el usuario, a su vez pasa estas peticiones a la capa de negocio y al finalizar el proceso envía la respuesta por medio de la misma interfaz al usuario.
* **Capa de Negocio:** Contiene todos los componentes de la lógica de la aplicación. Mediante *controladores,* la capa de negocio recibe las peticiones enviadas desde la capa de presentación y realiza las transacciones de negocio mediante el ORM eclipselink que se encarga del manejo de los datos y a su vez de la comunicación entra las capas de negocio y la capa de datos.
* **Capa de Datos:** Para la conexión de las aplicaciones de java y la base de datos, se hace uso del api JDBC. La capa de datos o de persistencia es la encargada de representar la información con la cual trabaja la aplicación.
  1. diseño

Durante la fase de diseño se hace la representación del funcionamiento de la aplicación mediante la realización de diagramas de secuencia, así como también se representa el modelo de los datos mediante un diagrama entidad-relación de la base de datos a diseñar.

Más adelante se muestra los prototipos de interfaz de usuario, para tener en cuenta en la siguiente fase de construcción.

* + 1. Diagramas de secuencia

Mediante los diagramas de secuencia se describe la interacción de los objetos del diseño, teniendo en cuenta los casos de uso del sistema, enunciados en la etapa de análisis.

A continuación, se muestran los diagramas de secuencia que modelan el flujo de la información y de los componentes principales del sistema, claves para el diseño de la aplicación.

* Diagrama de Secuencia No 1. Configurar Periodos

Figura 38. Diagrama de Secuencia - Configurar Periodos



* Diagrama de Secuencia No 2. Configurar Asignaturas

Figura 39. Diagrama de Secuencia - Configurar Asignaturas



* Diagrama de Secuencia No 3. Configurar Grados

Figura 40. Diagrama de Secuencia - Configurar Grados



* Diagrama de Secuencia No 4. Configurar Cursos

Figura 41. Diagrama de Secuencia - Configurar Cursos



* Diagrama de Secuencia No 5. Configurar Carga Académica

Figura 42. Diagrama de Secuencia - Configurar Carga Académica



* Diagrama de Secuencia No 6. Configurar Profesores

Figura 43. Diagrama de Secuencia - Configurar Profesores



* Diagrama de Secuencia No 7. Configurar Aulas

Figura 44. Diagrama de Secuencia - Configurar Aulas



* Diagrama de Secuencia No 9. Generar Horario

Figura 45. Diagrama de Secuencia - Generar Horario



* Diagrama de Secuencia No 8. Configurar Clases

Figura 46. Diagrama de Secuencia - Configurar Clases



* Diagrama de Secuencia No 10. Reportes

Figura 47. Diagrama de Secuencia - Reportes



* + 1. Modelo de datos

El modelo de datos muestra mediante el diagrama Entidad-Relación la estructura y diseño de la base de datos encargada del manejo de la información del sistema.

Figura 48. Diagrama Entidad-Relación



* + 1. Diccionario de datos

Mediante el diccionario de datos se hace la descripción de cada uno de los campos presentados en el modelo de datos, de esta manera se presenta de una forma más organizada y detallada de la información requerida en el sistema.

Tabla 9. Tabla aula\_tipo

|  |  |  |  |
| --- | --- | --- | --- |
| **NOMBRE DE TABLA** | aula\_tipo | | |
| **DECRIPCIÓN** | Contiene la información de los tipos de aulas que existen. | | |
| **COLUMNA** | **TIPO DE DATO** | **ENLACE A TABLA** | **DESCRIPCIÓN** |
| id\_tipo\_aula | Int |  | Identificador único del aula |
| nombre\_tipo\_aula | text |  | Nombre del tipo de aula |

Tabla 10. Tabla aula

|  |  |  |  |
| --- | --- | --- | --- |
| **NOMBRE DE TABLA** | aula | | |
| **DECRIPCIÓN** | Contiene la información de las aulas. | | |
| **COLUMNA** | **TIPO DE DATO** | **ENLACE A TABLA** | **DESCRIPCIÓN** |
| num\_aula | Int |  | Identificador único del aula |
| nombre\_aula | text |  | Nombre del aula |
| Tipo\_aula | Int | Aula\_tipo | Identificador único de la tabla aula\_tipo |
| Profesor | Int | Profesor | Identificador único de la tabla profesor |
| Curso | Int | Curso | Identificador único de la tabla curso |

Tabla 11. Tabla curso

|  |  |  |  |
| --- | --- | --- | --- |
| **NOMBRE DE TABLA** | curso | | |
| **DECRIPCIÓN** | Contiene la información de los cursos. | | |
| **COLUMNA** | **TIPO DE DATO** | **ENLACE A TABLA** | **DESCRIPCIÓN** |
| Id\_curso | Int |  | Identificador único del curso |
| nombre\_curso | text |  | Nombre del curso |
| Num\_estudiantes | Int |  | Número de estudiantes en el curso |
| Id\_grado | Int | grado | Identificador único de la tabla grado |

Tabla 12. Tabla area

|  |  |  |  |
| --- | --- | --- | --- |
| **NOMBRE DE TABLA** | area | | |
| **DESCRIPCIÓN** | Contiene la información de las areas. | | |
| **COLUMNA** | **TIPO DE DATO** | **ENLACE A TABLA** | **DESCRIPCIÓN** |
| cod\_area | Int |  | Identificador único del área |
| nombre\_area | text |  | Nombre del área |

Tabla 13.Tabla asignatura

|  |  |  |  |
| --- | --- | --- | --- |
| **NOMBRE DE TABLA** | asignatura | | |
| **DESCRIPCIÓN** | Contiene la información de las asignaturas. | | |
| **COLUMNA** | **TIPO DE DATO** | **ENLACE A TABLA** | **DESCRIPCIÓN** |
| Id\_materia | Int |  | Identificador único de la materia |
| nombre\_materia | text |  | Nombre de la asignatura |
| Cod\_area | Int | area | Identificador de la tabla área. |

Tabla 14. Tabla grado

|  |  |  |  |
| --- | --- | --- | --- |
| **NOMBRE DE TABLA** | grado | | |
| **DECRIPCIÓN** | Contiene la información de los grados. | | |
| **COLUMNA** | **TIPO DE DATO** | **ENLACE A TABLA** | **DESCRIPCIÓN** |
| Id\_grado | Int |  | Identificador único del grado |
| nombre\_materia | text |  | Nombre del grado |

Tabla 15. Tabla carga\_academica

|  |  |  |  |
| --- | --- | --- | --- |
| **NOMBRE DE TABLA** | Carga\_academica | | |
| **DECRIPCIÓN** | Contiene la información de carga académica. | | |
| **COLUMNA** | **TIPO DE DATO** | **ENLACE A TABLA** | **DESCRIPCIÓN** |
| Id\_materia | Int | asignatura | Identificador único de la tabla asignatura |
| Id\_curso | int | curso | Nombre de la tabla curso |
| Horas\_semana | Int |  | Horas de carga académica |

Tabla 16. Tabla pref\_profesor\_materia

|  |  |  |  |
| --- | --- | --- | --- |
| **NOMBRE DE TABLA** | Pref\_profesor\_materia | | |
| **DECRIPCIÓN** | Contiene la información de preferencia de materias de los profesores. | | |
| **COLUMNA** | **TIPO DE DATO** | **ENLACE A TABLA** | **DESCRIPCIÓN** |
| Id\_materia | Int | Carga\_academica | Identificador de la materia de la tabla carga\_academica |
| Id\_curso | int | Carga\_academica | Identificador del curso de la tabla carga\_academica |
| Id\_profesor | Int | profesor | Identificador único de la tabla profesor |

Tabla 17. pref\_profesor\_periodo

|  |  |  |  |
| --- | --- | --- | --- |
| **NOMBRE DE TABLA** | Pref\_profesor\_periodo | | |
| **DECRIPCIÓN** | Contiene la información de preferencia de periodos de los profesores. | | |
| **COLUMNA** | **TIPO DE DATO** | **ENLACE A TABLA** | **DESCRIPCIÓN** |
| Periodo | Int | periodo | Identificador de la materia de la tabla periodo |
| profesor | Int | profesor | Identificador único de la tabla profesor |

Tabla 18. Tabla periodo

|  |  |  |  |
| --- | --- | --- | --- |
| **NOMBRE DE TABLA** | periodo | | |
| **DECRIPCIÓN** | Contiene la información de los periodos académicos | | |
| **COLUMNA** | **TIPO DE DATO** | **ENLACE A TABLA** | **DESCRIPCIÓN** |
| Cod\_periodo | Int | periodo | Identificador de la materia de la tabla periodo |
| dia | Int | Dia | Identificador único de la tabla día |
| hora | int | Hora | Identificador único de la tabla hora |

Tabla 19. Tabla profesor

|  |  |  |  |
| --- | --- | --- | --- |
| **NOMBRE DE TABLA** | profesor | | |
| **DECRIPCIÓN** | Contiene la información de los profesores | | |
| **COLUMNA** | **TIPO DE DATO** | **ENLACE A TABLA** | **DESCRIPCIÓN** |
| Id\_profesor | Int | periodo | Identificador de la materia de la tabla profesor |
| Nombre\_profesor | text |  | Nombre del profesor |
| Horas\_max | Int |  | Horas máximas de trabajo del profesor |
| correo | Text |  | Correo electrónico del profesor |

Tabla 20. Tabla dia

|  |  |  |  |
| --- | --- | --- | --- |
| **NOMBRE DE TABLA** | dia | | |
| **DECRIPCIÓN** | Contiene la información de los días de la semana | | |
| **COLUMNA** | **TIPO DE DATO** | **ENLACE A TABLA** | **DESCRIPCIÓN** |
| Cod\_dia | Int |  | Identificador de la materia de la tabla dia |
| Nombre\_dia | text |  | Nombre del día |

Tabla 21. Tabla hora

|  |  |  |  |
| --- | --- | --- | --- |
| **NOMBRE DE TABLA** | hora | | |
| **DECRIPCIÓN** | Contiene la información de las horas de la semana | | |
| **COLUMNA** | **TIPO DE DATO** | **ENLACE A TABLA** | **DESCRIPCIÓN** |
| Cod\_hora | Int |  | Identificador de la materia de la tabla hora |
| Inicio | time |  | Hora de inicio de la de clase |
| Fin | Time |  | Hora de finalización de la clase |
| Duracion | Float |  | Duración de la clase |

Tabla 22. pref\_carga\_periodo

|  |  |  |  |
| --- | --- | --- | --- |
| **NOMBRE DE TABLA** | Pref\_carga\_periodo | | |
| **DECRIPCIÓN** | Contiene la información de la carga académica de los periodos de clase. | | |
| **COLUMNA** | **TIPO DE DATO** | **ENLACE A TABLA** | **DESCRIPCIÓN** |
| Id\_materia | Int | Carga\_academica | Identificador de la materia de la tabla carga\_academica |
| Id\_curso | Int | Carga\_academica | Identificador del curso de la tabla carga\_academica |
| periodo | int | Periodo | Identificador único de la tabla periodo |

Tabla 23. Tabla clase

|  |  |  |  |
| --- | --- | --- | --- |
| **NOMBRE DE TABLA** | clase | | |
| **DECRIPCIÓN** | Contiene la información de las clases. | | |
| **COLUMNA** | **TIPO DE DATO** | **ENLACE A TABLA** | **DESCRIPCIÓN** |
| Id\_clase | Int |  | Identificador único de la tabla clase |
| Materia | Int | Carga\_academica | Identificador del curso de la tabla carga\_academica |
| Curso | int | Carga\_academica | Identificador del curso de la tabla carga\_academica |
| Profesor | Int | profesor | Identificador único de la tabla profesor |
| Aula | Int | aula | Identificador único de la tabla aula |
| duracion | int |  | Duración de la clase |

Tabla 24. Tabla restriccion\_tipo

|  |  |  |  |
| --- | --- | --- | --- |
| **NOMBRE DE TABLA** | Restriccion\_tipo | | |
| **DECRIPCIÓN** | Contiene la información de los tipos de restricciones | | |
| **COLUMNA** | **TIPO DE DATO** | **ENLACE A TABLA** | **DESCRIPCIÓN** |
| Inicial | char |  | Identificador único de la tabla restriccion\_tipo |
| nombre | text | Carga\_academica | Nombre del tipo de restricción |

Tabla 25. Tabla restriccion

|  |  |  |  |
| --- | --- | --- | --- |
| **NOMBRE DE TABLA** | Restriccion | | |
| **DECRIPCIÓN** | Contiene la información de las restricciones | | |
| **COLUMNA** | **TIPO DE DATO** | **ENLACE A TABLA** | **DESCRIPCIÓN** |
| Id\_restriccion | char |  | Identificador único de la tabla restricción |
| Id\_horario | text | horario | Identificador único de la tabla horario. |
| Inicial | Char | Restricción\_tipo | Identificador único de la tabla restricción\_tipo |
| mensaje | text |  | Mensaje de la restricción |

Tabla 26. Tabla horario

|  |  |  |  |
| --- | --- | --- | --- |
| **NOMBRE DE TABLA** | Horario | | |
| **DECRIPCIÓN** | Contiene la información de los horarios | | |
| **COLUMNA** | **TIPO DE DATO** | **ENLACE A TABLA** | **DESCRIPCIÓN** |
| Id\_horario | int |  | Identificador único de la tabla horario |
| Nombre | Text |  | Nombre del horario |
| Fecha | Timestamp |  | Fecha de generación del horario |
| fitness | Float |  | Valor de la aptitud obtenida del horario |

Tabla 27. Tabla solucion

|  |  |  |  |
| --- | --- | --- | --- |
| **NOMBRE DE TABLA** | solucion | | |
| **DECRIPCIÓN** | Contiene la información de las soluciones | | |
| **COLUMNA** | **TIPO DE DATO** | **ENLACE A TABLA** | **DESCRIPCIÓN** |
| Id\_horario | int | Horario | Identificador único de la tabla horario |
| Id\_clase | int | Clase | Identificador único de la tabla clase |
| Cod\_periodo | Int | periodo | Identificador único de la tabla periodo |

* + 1. Prototipos de interfaz de usuario

Los prototipos de interfaz de usuario consisten en el diseño de las pantallas de la aplicación, que ayudan a comprender los casos de uso mejor y a especificar las interacciones entre los actores humanos y el programa, además lograr que la aplicación sea flexible, coherente, eficiente y sencilla de usar.

El objetivo de esta sección es especificar cada formato individual de la interfaz de las principales pantallas del sistema. En la definición y bosquejos se consideran aquellos aspectos importantes para el diseño y la construcción de la aplicación. En seguida se muestran diferentes pantallas de interfaz de usuario con su descripción.

* Ventana Principal

Este prototipo presenta el Menú Principal donde el usuario puede realizar todas las tareas definiendo un orden por etapas, señalando el avance en cada una, así se consigue una interfaz predecible e intuitiva para la configuración de los módulos, facilitando su manejo.

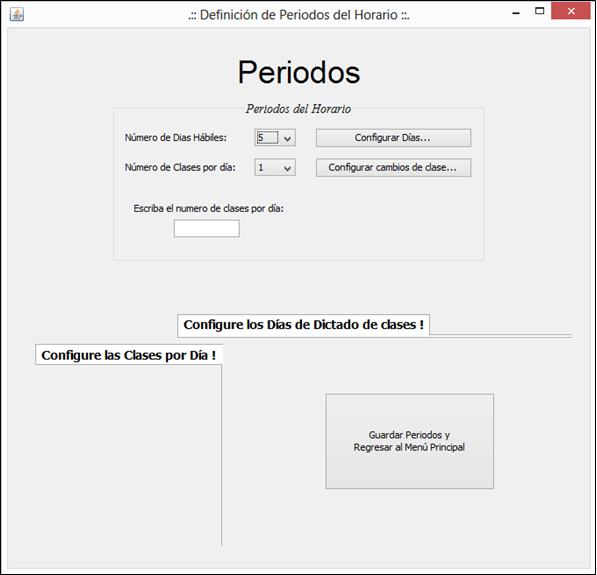
Figura 49. Interfaz de usuario - Menú Principal



* Ventana de Configuración de Periodos

La siguiente pantalla hace referencia a los casos de uso “Configurar Días de dictado” y “Configurar Horas de Clase”, a través de los cuales el usuario ingresará los periodos en que se pueda realizar el dictado de clases a la semana.

Figura 50. Interfaz de usuario - Configurar Periodos



* Ventana de Configuración de Asignaturas

Esta pantalla hace referencia a los casos de uso “Ingresar una Asignatura”, “Editar los datos de una Asignatura”, “Eliminar una Asignatura”, pueden visualizar las asignaturas de la base de datos en la lista (izquierda) y un panel para la edición (derecha); si se desea registrar una nueva asignatura o eliminarla se mostraran otras ventana y se actualizará la lista en cada caso.

Figura 51. Interfaz de usuario - Configurar Asignaturas



* Ventana de Configuración de Grados

Esta pantalla referencia los casos de uso No. 7, 8, 9, 10 y 11 relacionados con la administración de Grados y Cursos (grupos de estudiantes). Al presionar el botón “Eliminar” y “Nuevo Grado” aparecen otras ventanas y al finalizar la operación se actualiza la lista de la izquierda. Cabe destacar que el botón “Configuración de Cursos”, conlleva a que la aplicación abra otra ventana para la administración de los cursos, dependiendo del grado seleccionado en la lista.

Figura 52. Interfaz de usuario - Configurar Grados



* Ventana de Configuración de Carga Académica

El siguiente prototipo hace referencia a los casos de uso No. 12, 13 y 14, relacionados con la creación, edición y eliminación de cargas académicas, en el centro de la pantalla el usuario de la aplicación selecciona el grado y se visualiza los datos de las asignaturas en la tabla inferior (centro). Presionar el botón “Nueva Carga Académica”, conlleva a que se abra una ventana para la creación de una carga.

También está pantalla hace referencia al caso de uso “Crear una nueva configuración de Clases”, a través del cual el usuario realiza la configuración de las lecciones de dictado para la carga académica seleccionada.

Figura 53. Interfaz de usuario - Configurar Carga Académica



* Ventana de Configuración de Profesores

La pantalla siguiente referencia a los casos de uso “Ingresar un nuevo Profesor”, “Editar los datos de un Profesor”, “Eliminar un Profesor”, de tal manera que el usuario puede realizar la administración de la plana de docentes (lista de la izquierda), si se selecciona un docente en la lista, se puede eliminar, editar y realizar la configuración de preferencia de periodos y asignaturas, referenciando los casos de uso No. 18 y 19 respectivamente.

Figura 54. Interfaz de usuario - Configurar Docentes



* Ventana de Configuración de Aulas

Esta pantalla hace referencia a los casos de uso No. 20, 21 y 22 relacionados con la inserción, edición y borrado de aulas del sistema. El usuario selecciona de la tabla (centro) una fila que representa los datos de un aula y puede presionar “Editar” o “Eliminar” para modificar y actualizar la base de datos. La creación de Aulas se realiza en otra pantalla.

Figura 55. Interfaz de usuario - Configurar Aulas



1. Fase de construcción

En el siguiente capítulo se encuentra un espacio detallado del algoritmo genético que se utilizara para la solución del problema, se especifica cómo crear los componentes de un AG en términos de software, mediante el pseudocódigo y el código fuente se aprecia como el AG realizará la búsqueda de soluciones (horarios académicos) a nuestro problema.

* 1. Adaptación del algoritmo genético

Tomando en cuenta las bases teóricas mostradas en la sección y considerando las razones por las cuales se seleccionó los Algoritmos Genéticos, la sección siguiente tiene como objetivo diseñar un algoritmo que trate el problema.

* + 1. Unidad de asignación

En el problema del timetabling se asignan recursos a un elemento en particular, para nuestro problema en las escuelas los recursos son los docentes, aulas, materias (asignaturas) que se asignan a un grupo de estudiantes (curso) en unos periodos de tiempo.

La entidad en que vamos a asignar los recursos anteriores la llamaremos clase, la cual la definimos como la materia que pertenece a un curso impartida por un docente, con una duración, ocupando un aula determinada.

Para nuestra comodidad vamos a establecerle a la clase un identificador único que la diferencie de otras clases.

Figura 56. Estructura unidad de asignación (Clase)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | MATERIA | CURSO | PROFESOR | AULA | HORAS DE DICTADO |

Una vez definida la unidad, nuestro algoritmo se encargará de asignar un horario (periodos académicos) a cada clase.

Mediante un ejemplo veamos cómo es la estructura de diferentes clases: consideremos la asignatura Matemáticas del grado sexto con 4 cursos y una intensidad horaria semanal de 4 periodos académicos dictadas en las aulas asignadas a cada curso, en la siguiente tabla se organizan estos datos para cada curso, si las clases se organizan en bloques académicos de 2 horas:

Tabla 28. Horas de dictado de Matemáticas para el grado sexto

|  |  |  |
| --- | --- | --- |
| **Curso** | **Cantidad de clases** | **Horas de dictado** |
| 601 | 2 | 2 |
| 602 | 2 | 2 |
| 603 | 2 | 2 |
| 604 | 2 | 2 |

Fuente: Colegio María Mercedes Carranza IED, año 2014.

Teniendo en cuenta la tabla anterior las clases generadas de la materia Matemáticas del grado sexto son:

Tabla 29. Clases generadas para la asignatura Matemáticas del grado sexto

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID** | **Materia** | **Curso** | **Profesor** | **Aula** | **Horas** |
| 1001 | Matemáticas | 601 |  | Aula 601 | 2 |
| 1002 | Matemáticas | 601 |  | Aula 601 | 2 |
| 1003 | Matemáticas | 602 |  | Aula 602 | 2 |
| 1004 | Matemáticas | 602 |  | Aula 602 | 2 |
| 1005 | Matemáticas | 603 |  | Aula 603 | 2 |
| 1006 | Matemáticas | 603 |  | Aula 603 | 2 |
| 1007 | Matemáticas | 604 |  | Aula 604 | 2 |
| 1008 | Matemáticas | 604 |  | Aula 604 | 2 |

* + 1. Representación del Problema

En un algoritmo genético existe una población de individuos, donde cada uno de estos representa la solución codificada de un problema. En nuestro caso el recurso que no se asigna en la estructura clase son los horarios en que se dictará dicha clase, por lo tanto la estructura del cromosoma que representará la solución de nuestro problema será una matriz de dimensión [(cantidad de periodos)\*(cantidad de cursos) x 2], que contenga todos los periodos semanales y las clases de cada curso:

* Cursos: cantidad de grupos de estudiantes del colegio.
* Horas: número de horas para el dictado de clases en un día.
* Días: número de días hábiles para el dictado de clases.
* Periodos = Días \* Horas.

Por ejemplo, en el Colegio María Mercedes Carranza IED para el año 2014 se estableció que la cantidad de cursos (grupos de estudiantes) era 24, el número de horas o ranuras de tiempo es de 6 para 5 días de la semana.

Entonces,

***Cursos*** = 24

***Horas*** = 6

***Días*** = 5

***Periodos*** = 5 \* 6 = 30

Retomando, nuestra matriz tendrá una dimensión [720 x 2], donde 720 se obtiene de multiplicar la cantidad de cursos por el número de periodos, mientras que 2 son las columnas de la matriz, cada una representa respectivamente:

1. Identificador del periodo académico.
2. Identificador de la clase.

En la siguiente figura se puede apreciar la estructura de cada cromosoma, representado en una matriz, donde los valores de la columna 1 (periodos) se mantendrán sin variar (estáticos), mientras que los datos de la columna 2 (clases) serán los que cambiaran de posición en el cromosoma, mediante los operadores de reproducción y mutación que se explican más adelante. Un ejemplo del cromosoma es el siguiente:

Figura 57. Representación matricial del cromosoma (horario de clases)

|  |  |
| --- | --- |
| **Periodo** | **Clase** |
| 1 | 1001 |
| 2 | 1001 |
| 3 | 1020 |
|  |  |
| 29 | 1031 |
| 30 | 1032 |
| 1 | 1003 |
| 2 | 1004 |
| 3 | 1014 |
|  |  |
| 29 | 1021 |
| 30 | 1022 |

De la figura anterior se diferencias dos colores rojo y azul, cada uno representa el horario (calendario) de un grupo de estudiantes (curso), es decir que los calendarios de cada curso se situarán uno debajo del otro.

También se puede apreciar que los alelos (valores de los genes) del cromosoma serán números enteros, es decir que la codificación utilizada es una codificación directa.

* + 1. Codificación mediante alfabetos duros.

El tipo de codificación influye en los tipos de operadores que se puedan utilizar, de esta manera la elección del tipo de codificación se debe realizar tratando en lo posible que se refleje la naturaleza del problema. En la sección 1.7.2.4 se expuso los dos tipos de codificación: directa e indirecta, siendo esta última más común en la literatura y en la implementación de algoritmos mediante la representación binaria, con ventajas de cómputo y programación debido a que presenta una estructura simple [0, 1].

Sin embargo al utilizar un alfabeto binario en casos complejos, hay un costo computacional que implica traducir la solución y en muchos casos valores que no se encuentran dentro del rango permitido, por ejemplo: Si tenemos un conjunto de 20 clases para un curso determinado, utilizando combinación binaria se necesitan 5 bits para representar todas las clases, es decir 25 = 32 posibles combinaciones, sin embargo al solo utilizar 20 de estas, las 12 restantes no se utilizarían ya que quedarían fuera del rango. Por el contrario, si utilizamos una codificación directa como los números enteros solo tenemos un intervalo [1, 20], además si estos números representan el identificador de cada clase, nos evitamos el procedimiento de traducir el dato y no tendremos valores fuera del rango.

Para nuestro caso el cromosoma utilizará una codificación directa mediante la representación de números enteros, evitando que los genes tomen valores fueras del espacio factible de soluciones, específicamente esto significa que, el algoritmo asigna todas y cada una de las clases de un curso a un periodo diferente desde la primera generación (iteración) hasta la última; bajo este criterio la codificación dura protege al modelo de la violación de restricciones.

* + 1. Inicialización de la población

Definida la estructura de nuestro cromosoma, lo siguiente es pasar a la primera etapa de un Algoritmo Genético: la generación de la población inicial. Para este paso es indispensable conocer cuántos individuos van a conformar la población, parámetro que se establece previo al inicio del algoritmo.

Generalmente el proceso de iniciar la población de un Algoritmo Genético se realiza de manera aleatoria, aunque existen varios trabajos y propuestas para que esta labor se realice más guiada a la naturaleza del problema. La ventaja de inicializar la población aleatoriamente es que se obtiene diversidad en los individuos, esta diversidad una de las causas que permite a un AG (Algoritmo Genético) alcance excelentes resultados con técnicas simples, ya que permite manejar un conjunto de diferentes soluciones y con el paso de las iteraciones explotarlas para obtener individuos más adaptados al problema.

Inicializar la población consiste en tomar cada individuo y asignarle un valor a cada gen del cromosoma, en nuestro caso, por cada curso se tendrá un conjunto de clases y se obtiene el identificador único de cada clase para asignarlo aleatoriamente a un índice de la matriz, con la condición que este índice pertenezca al intervalo del curso.

Un curso *i*, tiene asignado un conjunto de clases:

Figura 58. Matriz de Clases para un curso i

{

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Clase 1 | Clase 2 | … | Clase k-1 | Clase k |

}

Supongamos que la matriz que representa al cromosoma está indexada a 0, de esta manera el intervalo de índices para asignar las clases es: donde Periodos son la cantidad de periodos habilitados para dictar una clase y es la posición del curso i en el matriz (recordemos que un curso está debajo de otro, por tanto existe un orden de cursos en el cromosoma).

Ejemplo: Supongamos que , luego, el conjunto de clases de este curso.

Figura 59. Matriz de Clases para el curso 601

{

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID** | **Materia** | **Curso** | **Profesor** | **Aula** | **Horas** |
| 1001 | Matemáticas | 601 |  | Aula 601 | 2 |
| 1002 | Matemáticas | 601 |  | Aula 601 | 2 |
| 1035 | C. Sociales | 601 |  | Aula 601 | 2 |
| 1036 | C. Sociales | 601 |  | Aula 601 | 2 |
| 1083 | Educación Artística | 601 |  | Aula 601 | 2 |
| 1107 | Educación Física | 601 |  | Aula 601 | 2 |
| 1138 | Español | 601 |  | Aula 601 | 2 |
| 1139 | Español | 601 |  | Aula 601 | 2 |
| 1172 | Ética y Religión | 601 |  | Aula 601 | 2 |
| 1214 | Gestión Social | 601 |  | Aula 601 | 2 |
| 1245 | Inglés | 601 |  | Aula 601 | 2 |
| 1246 | Inglés | 601 |  | Aula 601 | 2 |
| 1293 | C. Naturales | 601 |  | Aula 601 | 2 |
| 1294 | C. Naturales | 601 |  | Aula 601 | 2 |
| 1344 | Tecnología | 601 |  | Aula 601 | 2 |

}

Además que la posición del curso *i* en la matriz es la tercera, entonces el intervalo de índices es: (Periodos = 30 para nuestro ejemplo como se determinó en secciones anteriores).

El procedimiento a seguir es tomar la primera clase de la lista (1001) y generar ***n*** números aleatorios de los periodos académicos (30), donde n son las horas de dictado de la clase (2) y al final asignar el identificador de la clase en el cromosoma donde la columna 1 corresponda al periodo aleatorio encontrado:

Figura 60. Estructura del procedimiento encargado de inicializar un cromosoma

***Procedimiento*** *InicializarCromosoma* ***(****Cromosoma,**ListaCursos****)***

*// Comentario: Cromosoma es la matriz que representa la solución*

*// Comentario: ListaCursos es la lista que contiene los índices de los cursos.*

***Para*** *i = 0* ***Hasta*** *ListaCursos.length* ***Hacer***

*Curso = ListaCursos i*

*LimInferior = i \* periodos*

*LimSuperior = ((i + 1) \* periodos) - 1*

***Para******Cada*** *clase ListaClases curso*

*j = 1*

*n = clase.Horas*

***Mientras*** *j <= n* ***Hacer***

*Periodo = aleatorio (periodos)*

*Índice = encontrarIndice (Periodo, Cromosoma, limInferior, limSuperior)*

*Cromosoma [índice] [2] = clase.ID*

*j = j + 1*

***Fin Mientras***

***Fin Para***

***Fin Para***

***Fin Procedimiento***

Donde es una función que retorna un número aleatorio entre [1, k] y es una función que retorna el índice de la fila, cuando recorre la matriz desde el límite Inferior hasta el límite Superior (ambos inclusive) si el valor de ésta en la columna 1 es el periodo enviado como primer parámetro.

En la siguiente figura se aprecia un ejemplo de inicialización del cromosoma para el curso 601:

Figura 61. Representación de una sección del cromosoma inicializado

|  |  |  |
| --- | --- | --- |
| **Índice** | **Periodo** | **Clase** |
|  |  |  |
| [90] | 1 | 1172 |
| [91] | 2 | 1172 |
| [92] | 3 | 1083 |
| [93] | 4 | 1083 |
| [94] | 5 | 1002 |
| [95] | 6 | 1002 |
| [96] | 7 | 1293 |
| [97] | 8 | 1293 |
| [98] | 9 | 1035 |
| [99] | 10 | 1035 |
|  |  |  |
| [110] | 21 | 1344 |
| [111] | 22 | 1344 |
| [112] | 23 | 1107 |
| [113] | 24 | 1107 |
| [114] | 25 | 1139 |
| [115] | 26 | 1139 |
| [116] | 27 | 1036 |
| [117] | 28 | 1036 |
| [118] | 29 | 1246 |
| [119] | 30 | 1246 |
|  |  |  |

De la figura anterior se puede verificar que las clases con el mismo identificador se encuentran consecutivas, esto beneficia al AG porque permite dar cumplimiento a la restricción dura de *lecciones seguidas*; para conseguirlo se debe modificar el procedimiento visto en la figura anterior.

De la misma forma como se inicializa un cromosoma, se puede realizar el mismo procedimiento con todos los individuos de la población, llamando la función declarada en la sección anterior:

Figura 62. Estructura del procedimiento que inicializa la población

***Procedimiento*** *InicializarPoblacion* ***(****Población, ListaCursos****)***

*// Comentario: Población es la lista que contiene todos los individuos.*

*// Comentario: ListaCursos es la lista que contiene los índices de los cursos.*

***Para******Cada*** *individuo Población*

*InicializarCromosoma (individuo.Cromosoma, ListaCursos)* ***Fin Para***

***Fin Procedimiento***

* + 1. Función de evaluación y manejo de las restricciones

Después de tener cada cromosoma de la población con valores en sus genes, es momento de determinar si estas soluciones iniciales son factibles para dar solución al problema, para ello necesitamos un criterio para evaluar. Es aquí donde el AG requiere la implementación de una función de evaluación, que como indica su nombre permita evaluar a cada individuo de la población.

La función de evaluación es creada debido a la naturaleza del problema, por lo que es diferente para cada algoritmo genético. El objetivo de esta función es evaluar cada individuo para determinar qué tan apta es la solución al problema, es decir que esta función establece que “*tan buena*” es una solución o el caso contrario que “*tan mala*”.

Para nuestro AG la función de evaluación, comprueba cada horario de clases (representado en el cromosoma) para conocer si este es viable para ser usado en la institución o colegio. El criterio de evaluación de nuestra función, lo determinan las diferentes restricciones que el horario debe cumplir ya sean obligatorias y deseadas. Naupari y Rosales [[61]](#footnote-61) proponen un modelo para una función de evaluación para el problema de “University Timetabling” que se puede adaptar a nuestro problema, de tal manera que:

Donde son las aptitudes calculadas para cada restricción obligatoria, mientras que representan las aptitudes para las restricciones deseadas, es decir que ***n*** es el número de restricciones obligatorias y ***m*** es el número de restricciones deseadas, de este modo:

Es decir que una restricción obligatoria (dura) está cumplida si **,** dado que hay 0 violaciones a la restricción. Igualmente se dice que una restricción deseada se cumple si **,** ya que existen 0 violaciones.

Asimismo, y son los pesos o penalidades de las restricciones obligatorias y deseadas respectivamente, cumpliendo que:

Como se ha indicado las restricciones obligatorias deben ser cumplidas en su totalidad, es decir que para penalizar a los individuos que no cumplan con estas restricciones debe ser mucho mayor que

Cabe destacar que el símbolo (letra pi mayúscula) Π es el operador productorio o producto, que representa una multiplicación de una cantidad arbitraria, veamos un ejemplo:

Si una solución cumple con todas las restricciones, entonces no existe alguna violación, es decir:

Y la función de evaluación quedaría simplificada a la suma de los pesos de las restricciones, como se especificó en párrafos anteriores esta suma es 1. De esta manera si el resultado de la función de evaluación aplicada a un individuo es 1, se concluye que éste es una solución viable al problema.

* + 1. Restricciones del problema “school timetablig” a evaluar

A continuación se mencionan las restricciones tanto obligatorias (duras), como deseadas (blandas), que verificará la función de evaluación con el propósito de saber que tan lejos se está de una solución perfectamente factible[[62]](#footnote-62).

* + - 1. Restricciones duras (Obligatorias)
* **Sesiones de Clase:** Todas las sesiones (lecciones) de clase de las materias (asignaturas) deben programarse y asignarse a diferentes periodos. Se considera una violación si la lección no está programada.
* **Aulas Ocupadas:** Dos sesiones de clase no pueden ser programadas en la misma aula, durante el mismo periodo de tiempo. Se cuenta como una violación adicional cualquier sesión de clase extra en una misma aula y periodo.
* **Conflictos**: Las lecciones de clase de las asignaturas dictadas por un mismo de profesor deben programarse en diferentes periodos. Una violación ocurre si dos o más lecciones están en conflicto.
* **Días de Dictado**: Las sesiones de clase deben ser dictadas en un número x de días. Cada día por debajo de esta cantidad (x), se considera una violación.
* **Lecciones Seguidas**: Las lecciones de clase con una duración mayor a 1, que pertenecen a una misma asignatura y un mismo curso, deben ser programadas en periodos consecutivos. Si existe un bloque de clases donde sus sesiones no se encuentren seguidas, se considera una violación.
* **Disponibilidad**: Si un profesor de una asignatura no está disponible para dictarla en un periodo dado, entonces ninguna sesión de clase de debe programarse en ese periodo, que dicte ese profesor. Cada lección de clase en un periodo de no disponible para ese profesor, se cuenta como una violación.
  + - 1. Restricciones blandas (Deseadas)
* **Distribución Intensidad Horaria**:Si una asignatura de un curso debe dictarse en 3 días de 5 disponibles, entonces debe existir 1 día entre las lecciones de clases. Si no se cumple al menos un día entre las lecciones, se considera una penalización.
* **Disponibilidad**: Esta restricción es la misma señalada en las restricciones duras, pero en este caso como una restricción deseada, se considera una penalización si una lección de clase se programa en un periodo no disponible para el profesor asignado.

Existen otras restricciones que se tienen en cuenta al resolver este problema, con las restricciones expuestas se puede lograr una solución factible. Otras restricciones se pueden encontrar en [[63]](#footnote-63), como por ejemplo una restricción blanda que cumpla con la capacidad de las aulas, y una restricción que tenga en cuenta la distancia de traslado de los estudiantes en los cambios de clase.

* + 1. Criterio de parada

Después de evaluar cada individuo de la población mediante la función de adaptación, se debe verificar si el algoritmo ha encontrado la solución del problema y debe detenerse. En nuestro AG si se cumple al menos una de las siguientes condiciones el algoritmo debe culminar:

1. Se ha alcanzado un número determinado de iteraciones, de esta manera, antes de iniciar el AG se debe estipular el número máximo de generaciones. Este criterio es importante, porque si no se tiene en cuenta el AG nunca se detendría.
2. Se ha detectado que un alto porcentaje de la población converge a un solo individuo, quiere decir que el algoritmo se estancó en un óptimo local.
3. Se ha encontrado que existen 5 o más soluciones factibles al problema.

Según lo estipulado en la sección…4.1.5…un horario de clases es viable para implementar, si el resultado de aplicarle la función de adaptación es 1.

* + 1. Operadores genéticos

Para el paso de una generación a otra se aplican una serie de operadores genéticos, es decir mientras la condición de término o parada no se cumpla se deben ejecutar los operadores en el siguiente orden: selección, cruce (reproducción), mutación y sustitución (reemplazo) de individuos cada generación o iteración.

* + - 1. Selección

Este operador genético en un AG escoge individuos para la reproducción. El proceso de selección tiene en cuenta la aptitud de cada individuo (resultado de aplicar la función de evaluación), para tomar los individuos con mejor aptitud al problema.

Existen diversos métodos para realizar este proceso de selección, el objetivo de este procedimiento es que los individuos con mejor aptitud tengan más probabilidad de ser escogidos para reproducirse. El método que nuestro AG va implementar es conocido como:

**Selección por torneo**

La idea principal de este procedimiento trata de realizar la selección en base a comparaciones entre individuos. Este método contiene de forma simultánea características aleatorias y determinísticas.

Primero se elige aleatoriamente un número **p** de individuos a comparar (donde p se conoce como: ) y luego se escoge el individuo más apto del conjunto. Un tamaño de torneo común es: a este torneo se le denomina *.* En la siguiente figura se puede apreciar la estructura del procedimiento.

Figura 63. Estructura del procedimiento para la selección por torneo

***Procedimiento*** *SeleccionPorTorneo* ***(****Población, Tamaño****)***

*// Comentario: Población es la lista que contiene los individuos evaluados.*

*// Comentario: Tamaño es el número que representa el tamaño del torneo.*

*NuevaPoblación // NuevaPoblación es la Población de individuos a reproducir.*

***Para*** *i = 0* ***Hasta*** *Población.length / 2* ***Hacer***

*Competidores // Competidores es una lista de Individuos.*

***Para*** *j = 0* ***Hasta*** *Tamaño* ***Hacer***

*Competidores j = escogerIndividuo (Población)*

***Fin Para***

*Ganador = escogerGanador (Competidores)*

*NuevaPoblación i = Ganador*

***Fin Para***

***Fin Procedimiento***

Donde es otro procedimiento que elige un individuo de la población de forma aleatoria. Además, es el método que determina el individuo más apto del conjunto de competidores, en nuestro AG selecciona el individuo que tenga mayor aptitud.

* + - 1. Cruce

Este operador también se conoce como emparejamiento, reproducción o crossover es imprescindible en un AG. En este proceso se seleccionan dos (generalmente) individuos de manera aleatoria de la población obtenida del operador anterior, para cruzarlos y obtener una descendencia que contenga información de los padres, para que conformen la nueva población en la siguiente generación.

Habitualmente este operador no se aplica a todos los individuos seleccionados para emparejarse, sino que se aplica de forma aleatoria, normalmente con una probabilidad comprendida entre [0.5, 1.0). En el caso en que el operador de cruce no se aplique, la descendencia se obtiene duplicando los padres.

En la implementación del algoritmo para nuestro problema se utiliza un método compuesto por 2 técnicas desarrolladas que se encuentran en la literatura como el *Operador de Cruce Basado en el Orden OX2* propueto por Syswerda (1991), modificando los estudios realizados del operador OX1 porpuesto por Davis (1985); y el Otro operador se conoce como *inver-over* propuesto por Michalewicz.

**OX2** selecciona al azar varias posiciones en el string de uno de los padres, para a continuación imponer en el otro padre, el orden de los elementos en las posiciones seleccionadas[[64]](#footnote-64).

Por ejemplo consideremos los padres p1: (1 2 3 4 5 6 7 8) y p2: (2 4 6 8 7 5 3 1); y supongamos que en el segundo padre se seleccionan las posiciones segunda, tercera y sexta. Los elementos en dichas posiciones son 4, 6 y 5 respectivamente. En el primer padre dichos elementos se encuentran en las posiciones cuarta, quinta y sexta. El descendiente coincidirá con el primer padre si exceptuamos las posiciones cuarta, quinta y sexta:

(1 2 3 \* \* \* 7 8):

A continuación rellenamos los huecos del descendiente teniendo en cuenta el orden con el que aparecen en el segundo padre. Como resultado obtenemos

(1 2 3 4 6 5 7 8):

Cambiando el papel entre el primer y segundo padre, y utilizando las mismas posiciones seleccionadas, obtendremos el segundo descendiente:

(2 4 3 8 7 5 6 1):

**Inver-Over** por su parte este operador incorpora, en una solución, el conocimiento tomado desde otros individuos de la población. El cual fusiona la inversión y la recombinación. La inversión se aplica a una parte de un único individuo, pero la selección del segmento a invertir depende de otros individuos pertenecientes a la población[[65]](#footnote-65).

* + - 1. Mutación

Después de obtener una población de hijos por el operador anterior, un AG implementa el operador de mutación, en este caso se elige aleatoriamente un individuo de la población, para modificar su información, es decir cambiar el valor de uno o varios genes del cromosoma.

Al igual que el operador de cruce, la mutación se aplica con una probabilidad generalmente baja, para evitar que se pierda información valiosa heredada de los padres. El objetivo de este operador es permitir alcanzar zonas del espacio de búsqueda que no estaban cubiertas por los individuos de la población actual.

Para nuestra AG se ha decidido en primera instancia implementar el siguiente método para el aplicar el operador de mutación:

**Mutación Heurística**

En [[66]](#footnote-66) explican un tipo de mutación propuesta por Cheng y Gen. Está diseñada con la técnica del vecindario para producir un hijo mejorado. A partir de un cromosoma este se transforma al intercambiar **λ** genes, para producir un conjunto de cromosomas considerado el vecindario; donde el mejor vecino es tomado para reemplazar al hijo, este operador sigue los siguientes pasos:

1. Toma **λ** genes aleatoriamente.
2. Genera vecinos de acuerdo a todas las permutaciones posibles de los genes seleccionadas.
3. Evalúa a todos los vecinos y selecciona el mejor como hijo mutado.

La siguiente figura ilustra este procedimiento.

Figura 64. Demostración del operador de mutación heurística

1. Se tiene el siguiente cromosoma y se eligen al azar las posiciones 3, 6, 8:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

1. Generamos los vecinos de las permutaciones posibles:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 8 | 7 | 6 | 9 |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 8 | 4 | 5 | 3 | 7 | 6 | 9 |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 8 | 4 | 5 | 6 | 7 | 3 | 9 |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 6 | 4 | 5 | 8 | 7 | 3 | 9 |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 6 | 4 | 5 | 3 | 7 | 8 | 9 |

1. Supondremos que el mejor de los vecinos es:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 8 | 4 | 5 | 6 | 7 | 3 | 9 |

* + - 1. Reemplazo

En este operador el proceso a determinar es cómo será el reemplazo de la población actual con la población descendiente obtenida por los tres anteriores operadores expuestos.

Existen diversas técnicas para aplicar este operador, de las cuales:

* **Reemplazo de padres**: se obtiene un espacio para la nueva descendencia, liberando el espacio ocupado por los padres. Es decir, los hijos reemplazan a los padres.
* **Reemplazo de los peores**: de un porcentaje de los peores individuos de la población se seleccionan aleatoriamente los necesarios para dejar sitio a la descendencia.

**Elitismo**

El elitismo pertenece al proceso de reemplazo, consiste en conservar un número n de individuos o quitar un número n de malos individuos, con el objeto de no tener un retroceso evolutivo.

Un ejemplo de elitismo que se usa comúnmente y que se usará en este AG, es que el mejor individuo se mantenga en cada generación.

* 1. Pruebas

En esta sección se muestran las pruebas que se realizaron para determinar los parámetros de configuración del algoritmo genético, que permiten mejorar en la búsqueda de soluciones óptimas al problema. Además se presentan los resultados para dar fundamento al análisis para la selección de la mejor configuración, de tal manera que pueda dar solución al problema de la generación de horarios de clase.

Cabe destacar que el usuario puede configurar algunos de los parámetros en que se realizaron las pruebas, como la probabilidad de cruce y mutación, al igual que pueda configurar las diferente restricciones que evaluará el algoritmo genético.

* + 1. Introducción

Como propósito de esta actividad, se propuso encontrar una solución aproximada a través de la técnica de algoritmos genéticos para la resolución del problema de la generación de horarios académicos para los colegios o diversas instituciones escolares. Como caso de pruebas piloto se escogió el colegio María Mercedes Carranza IED con los datos para el periodo académico del año 2014.

* + 1. Caso de análisis

Para una correcta evaluación de los resultados se tendrán en cuenta dos casos de análisis, ambos se basan en la información del colegio mencionado para el ciclo académico del año 2014, la cual fue elaborada manualmente por el coordinador académico del colegio y demás personal responsable en la creación del horario de clases para la institución.

A continuación se describen los casos de análisis donde se realizarán las pruebas del algoritmo genético. En ambos casos se tendrán en cuenta una configuración de 30 periodos semanales para el dictado de clases, distribuidos en 6 horas de clase para 5 días de la semana.

* El primer caso de análisis, tomará el 50% de los datos para la generación de los horarios del ciclo mencionado. Este porcentaje de la información corresponde a las asignaturas que son dictadas desde el grado sexto (6to) hasta el grado octavo (8vo) de educación secundaria. En seguida se detalla los recursos tomados en cuenta para este caso:   
  + Se cuenta con un total de 12 cursos. (grupos de estudiantes)
  + Se cuenta con un total de 20 docentes de diferentes asignaturas, con sus respectivas disponibilidades y preferencia de dictado.
  + Se cuenta con un total de 12 aulas asignadas para cada curso.
* El segundo caso de análisis, tomará el 100% de los datos para la generación de los horarios del ciclo mencionado. Este porcentaje de la información corresponde a las asignaturas que son dictadas desde el grado sexto (6to) hasta el grado undécimo (11mo) de educación secundaria y media. En seguida se detalla los recursos tomados en cuenta para este caso:  
  + Se cuenta con un total de 24 cursos. (grupos de estudiantes)
  + Se cuenta con un total de 33 docentes de diferentes asignaturas, con sus respectivas disponibilidades y preferencia de dictado.
  + Se cuenta con un total de 24 aulas asignadas para cada curso.
    1. Esquema de pruebas

Cada prueba del algoritmo se realizará para diferentes valores de los parámetros, sin embargo para un análisis más detallado, las pruebas se realizarán con la misma población de individuos cada vez que se ejecute el algoritmo, así se consiguen resultados determinados directamente por el conjunto de parámetros a configurar.

Para medir el desempeño del algoritmo genético en base a los parámetros, como por ejemplo la probabilidad de cruce, se van a utilizar los siguientes tipos de mediciones:

* **El número de generaciones:** El objetivo principal es que se cumpla el número determinado de iteraciones sin que el algoritmo se detenga en un óptimo local sin llegar a una solución.
* **La frecuencia de éxito:** Hace referencia a la frecuencia con la cual el algoritmo encuentra soluciones óptimas durante diferentes ejecuciones de este.
* **La frecuencia cantidad de soluciones:** Corresponde a la frecuencia con la que el algoritmo encuentra al menos 5 soluciones en varias ejecuciones.
* **Promedio de la mejor aptitud obtenida:** Representa la media de la mejor aptitud obtenida en las diferentes ejecuciones. La aptitud se obtiene al aplicar la función evaluación vista en 4.1.5 donde el rango va de [0 a 1].
* **La frecuencia del mejor parámetro:** Indica el porcentaje de veces donde el algoritmo con ese valor del parámetro obtuvo la mejor aptitud.
  + - 1. Prueba sobre el tamaño del torneo

Esta prueba se realizó con el primer caso de análisis, ejecutando el algoritmo genético 10 veces y 80 generaciones para cada valor del tamaño del torneo.

Para realizar esta prueba se deciden proporcionar diferentes valores al tamaño del torneo y evaluar cuál es mejor para configurarlo, en seguida se muestra una tabla que resume el resultado de la prueba.

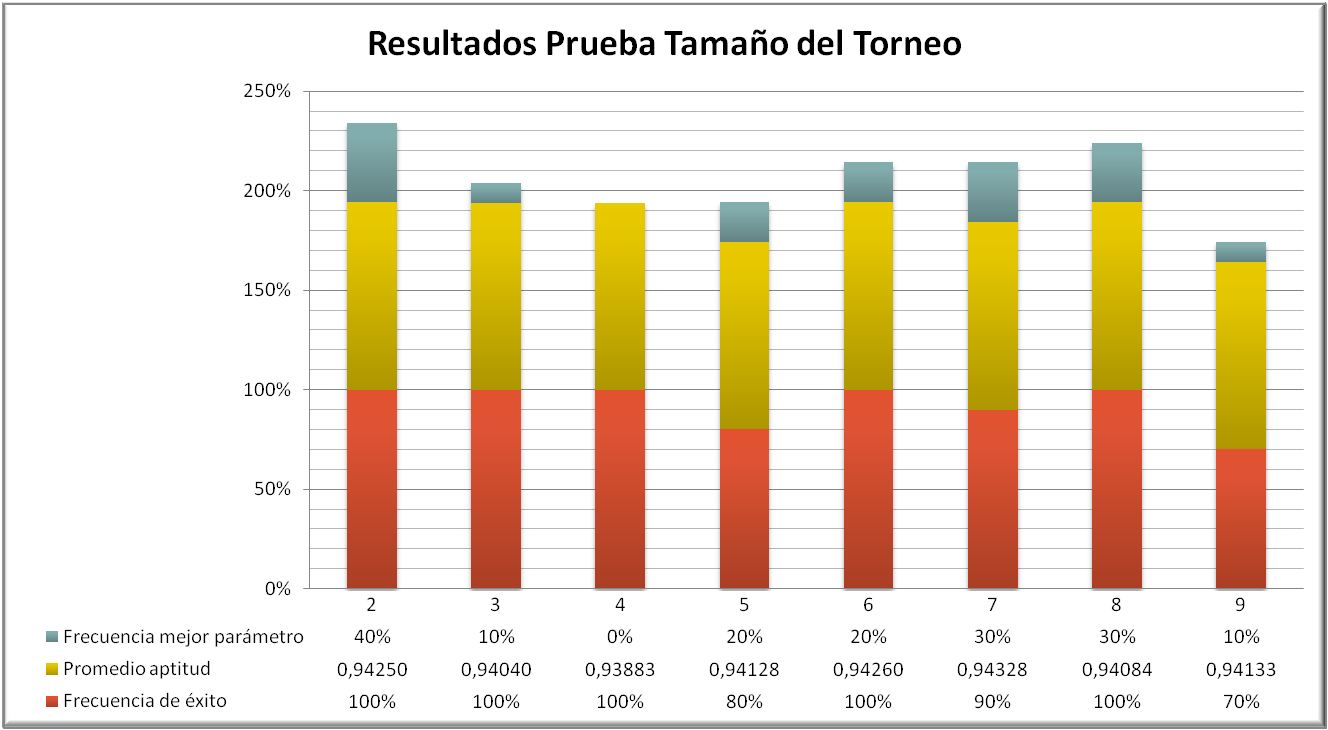
Tabla 30. Resultados prueba tamaño del torneo

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Tamaño del Torneo** | **Generaciones** | **Frecuencia soluciones** | **Prom. aptitud** | **Frecuencia mejor parámetro** |
| 2 | 100 % | 100 % | 0,94250 | 40% |
| 3 | 100 % | 100 % | 0,94040 | 10% |
| 4 | 100 % | 100 % | 0,93883 | 0% |
| 5 | 100 % | 80 % | 0,94128 | 20% |
| 6 | 100 % | 100 % | 0,94260 | 20% |
| 7 | 100 % | 90 % | 0,94328 | 30% |
| 8 | 100 % | 100 % | 0,94084 | 30% |
| 9 | 100 % | 70 % | 0,94133 | 10% |

Del anterior resultado de la prueba se puede decir que para todos los valores del tamaño del torneo, se consiguió llegar a las 80 generaciones establecidas. Para la columna de cantidad de soluciones los resultados arrojados traducen que para los tamaños 5, 7 y 9 no se obtuvo las 5 soluciones esperadas.

Para determinar cuál será el tamaño del torneo apropiado, se muestra el siguiente gráfico en relación a la tabla anterior, arrojando como el mejor resultado el 2 para el tamaño del torneo, destacando que es el tamaño donde existe una mayor frecuencia como el parámetro con mejor aptitud obtenida comparada con los demás.

Figura 65. Resultados prueba tamaño del torneo



* + - 1. Prueba de porcentajes de cruce

Una vez definido el tamaño del torneo, es importante determinar qué porcentaje de emparejamiento guiará la búsqueda de soluciones factibles al problema. Esta prueba se realizó ejecutando el algoritmo genético 15 veces, cada una de estas, generó una sola población inicial, la cual fue iterada por cada porcentaje de ocurrencia de cruce, durante 100 generaciones.

Los porcentajes a evaluar en la prueba se encuentran en la siguiente tabla:

Tabla 31. Porcentajes de Cruce

|  |
| --- |
| Porcentaje |
| 80 % |
| 85 % |
| 90 % |
| 95 % |

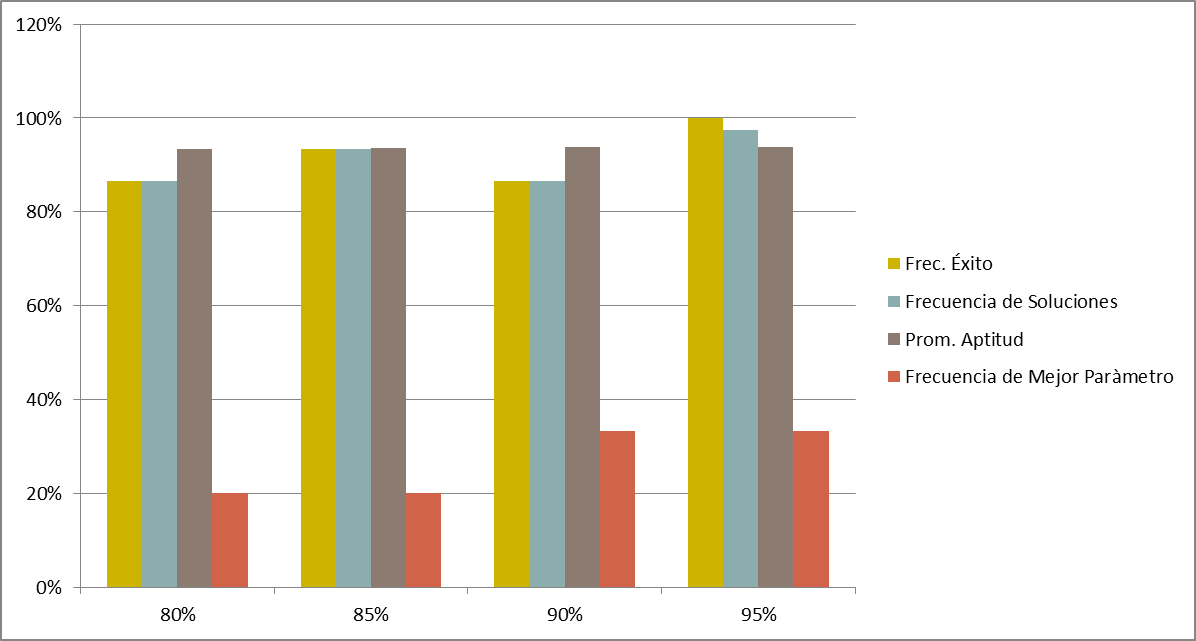
En la siguiente tabla se muestra los resultados de la prueba del parámetro de la probabilidad e emparejamiento donde el único parámetro que logro un porcentaje del 100% en la frecuencia de éxito fue 0.95 de probabilidad de emparejamiento teniendo al mismo tiempo la aptitud más alta de satisfactibilidad de las restricciones.

Tabla 32. Resultados Prueba Porcentaje de Cruce

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Porcentaje de Cruce** | **Generaciones** | **Frec. Éxito** | **Frecuencia de Soluciones** | **Prom. Aptitud** | **Frecuencia de Mejor Parámetro** |
| 80% | 100 | 87% | 87% | 0,93423268 | 20% |
| 85% | 100 | 93% | 93% | 0,93687937 | 20% |
| 90% | 100 | 87% | 87% | 0,93794403 | 33% |
| 95% | 100 | 100% | 97% | 0,93902842 | 33% |

El siguiente grafico representa los resultados expuesto en la tabla anterior.

Figura 66. Resultados prueba porcentaje de cruce



* + - 1. Prueba de porcentaje de mutación

Otro parámetro importante en la ejecución de un algoritmo genético es determinar la probabilidad de mutación. La siguiente prueba se realizó haciendo 8 ejecuciones tomando en cuenta los parámetros de las pruebas anteriores en un total de 200 iteraciones para cada uno de los siguientes porcentajes.

Tabla 33. Porcentajes de Mutación

|  |
| --- |
| Porcentaje |
| 20 % |
| 25 % |
| 30 % |

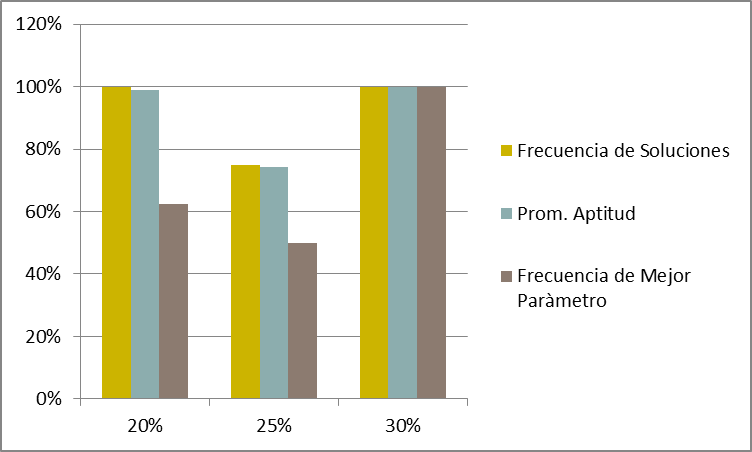
Los resultados de la prueba obtenida se aprecian en la siguiente tabla, donde se aprecia que el número de generaciones configuradas no se alcanzó debido a que se encontró una solución óptima al problema, es decir, con una aptitud de 1.

Tabla 34. Resultados de la Prueba de Porcentaje de Mutación

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Porcentaje de Mutación** | **Promedio Generaciones** | **Frec. Éxito** | **Frecuencia de Soluciones** | **Prom. Aptitud** | **Frecuencia de Mejor Parámetro** |
| 20% | 107,5 | 100% | 100% | 0,98964284 | 63% |
| 25% | 111,5 | 75% | 75% | 0,74333333 | 50% |
| 30% | 96,375 | 100% | 100% | 0,9975 | 100% |

Los resultados mostrados en la tabla anterior se muestran en el siguiente gráfico, donde se visualiza que un porcentaje de mutación 30% mejora en la búsqueda de soluciones óptimas debido a que el promedio de aptitud es más cercano a 1.

Figura 67. Resultados prueba porcentaje de mutación



1. Fase de transiciòn

Esta fase del proyecto comprende la ejecución del programa con el fin de obtener y evaluar el cumplimiento de los objetivos planteados y los requisitos expuestos en la fase de inicio.

* 1. Despliegue

Para realizar el procedimiento de despliegue de la aplicación se ha decidido alojar la base de datos en un servidor de tal manera que el usuario pueda acceder remotamente, esto trae como beneficio al usuario ya que evita la instalación y configuración de un gestor de base de datos, que en el caso de este proyecto es PostgreSQL.

Teniendo en cuenta esto se implementó el uso del servicio heroku-postgres para administrar y alojar la base de datos, la aplicación se ha programado de tal manera que se pueda acceder con una conexión a internet para la carga de la información.

Antes de realizar la puesta en marcha del programa, se debe tener en cuenta el manual de instalación ANEXO C: Manual de Instalación siguiendo las recomendaciones de este.

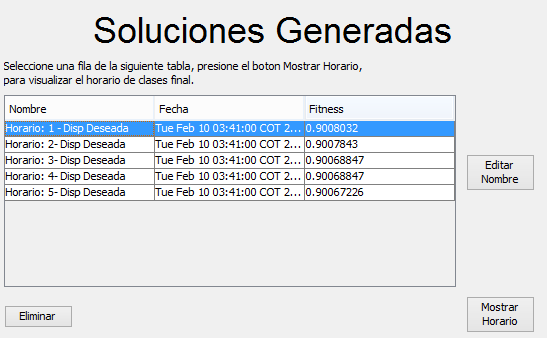
Una vez instalada la máquina virtual de java JVM, se puede proceder con la ejecución de la aplicación de escritorio para la generación de horarios académicos directamente desde su archivo ejecutable Horarios\_0.1.jar. ANEXO C: Manual de Instalación.

* 1. Resulados

Una vez es recopilados y almacenados los datos correspondientes al ciclo académico del año 2014 del colegio María Mercede Carranza IED, se arrojaron los resultados mostrados a continuación:

Al finalizar con la ejecución de la aplicación, siempre se van a guardar en la base de datos las 5 mejores soluciones encontradas

Figura 68. Listado de soluciones generadas



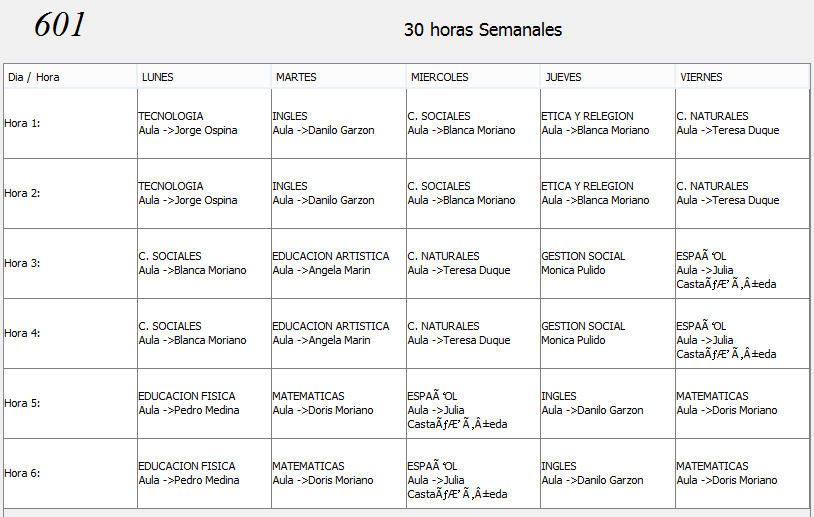
Cada una de las soluciones generadas muestra un horario diferente de tal manera que se visualiza la organización final de las clases, obteniendo así cada una un horario general que cumple con todas las restricciones y parámetros requeridos.

Figura 69. Horario completo de cursos



Una de los resultados más importante e indispensables de la generación de los horarios, es la visualización del horario de cada uno de los cursos del colegio, mostrando allí la información completa de la clase (periodo, materia, profesor, aula, curso). De allí se parte para finalmente estudiar la satisfactibilidad de las restricciones y la obtención o no de un horario óptimo.

Figura 70. Horario individual de un curso



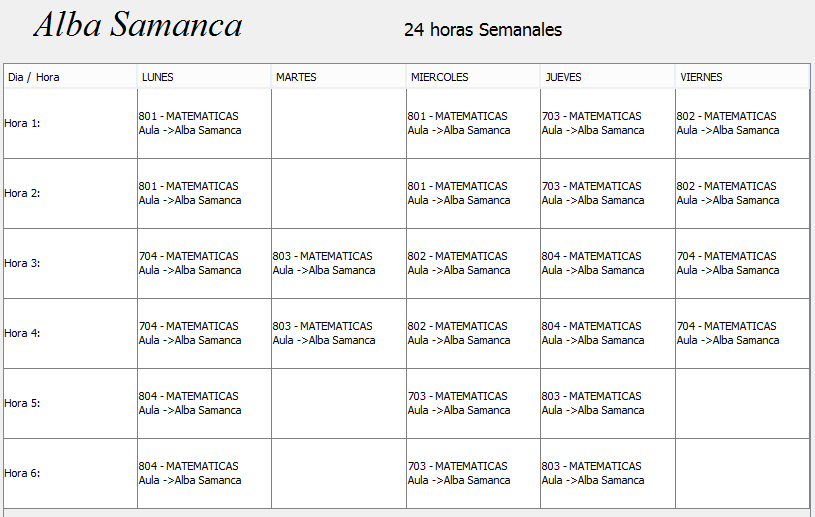
La segunda instancia indispensable en la revisión de la solución es la visualización del horario de los profesores del colegio durante la semana. El sistema muestra de manera general la asignación de clases con respecto a los docentes.

Figura 71. Horario completo de profesores



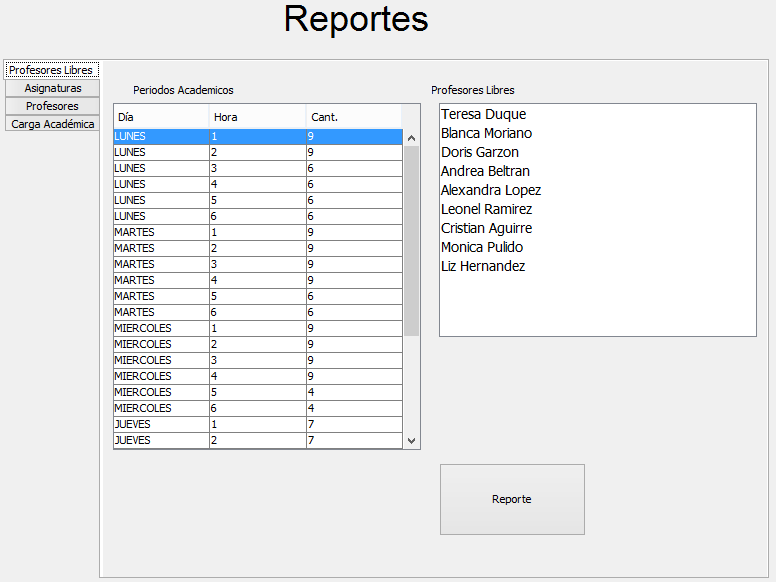
De igual manera, el sistema permite visualizar uno por uno el horario de cada uno de los profesores que dictaran las diferentes asignaturas a los cursos programados.

Figura 72. Horario individual de un profesor



Por último, los reportes son información para la revisión de los recursos de la institución, además, permiten la mejora en la toma de decisiones en cambio que el usuario requiera o se vea obligado a realizar.

Figura 73. Reportes del horario generado



# CONCLUSIONES

Con los resultados obtenido y teniendo en cuenta el objetivo general de este proyecto, se puede decir que se creó una aplicación computacional capaz de realizar la generación automática de horarios académicos para los colegios mediante en estudio e implementación de algoritmos genéticos.

Al finalizar este proyecto se cuenta con el conocimiento acerca de la técnica metaheurística de algoritmos genéticos, distinguiendo sus componentes, estructura y funcionamiento, aplicándola en la resolución del problema del School Timetabling.

Mediante el desarrollo de esta aplicación de escritorio que cuenta con módulos para la captura y actualización de los datos el usuario final cuenta con una herramienta computacional que le permite la generación de horarios académicos óptimos y el manejo de la información de una manera organizada y actualizada mediante la visualización de reportes.

Implementar el algoritmo genético traslada la creación de horarios académicos de forma manual a un campo autónomo y sistematizado, reduciendo los costos operativos y los recursos humanos encargados de realizar esta labor.

Otra ventaja de implementar algoritmos genéticos es la obtención de más de un resultado al problema que se desea tratar, en nuestro caso, el resultado final es la generación de más 1 horario factible que puedas ser utilizado en la institución.

Teniendo en cuenta los resultados de las pruebas realizadas en cuanto a la ejecución de los parámetros del algoritmo genético, se puede decir con total seguridad que la implementación de la técnica para el operador de selección conocida como el torneo binario, es decir, que el tamaño de este es de 2 individuos que compiten entre sí, cumple con su objetivo de guiar la búsqueda y escoger los mejores individuos de la población para un posterior cruce.

Una probabilidad alta para el emparejamiento de los individuos obtenidos por el método de selección permite explorar el espacio de búsqueda y garantiza diversidad en los individuos de la población, así mismo, su objetivo es que gran parte de la información genética de los padres se conserve de generación en generación, aportando mejores características a la solución a buscar.

El método de mutación heurística implementado en este proyecto permite explotar el espacio de búsqueda garantizando siempre un mejor o igual individuo, que se adapte al problema.

# TRABAJOS FUTUROS

# BIBLIOGRAFIA

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

ABRAMSON, David; KRISHNAMOORTHY, Mohan; DANG, Henry. Simulated annealing cooling schedules for the school timetabling problem. Asia-Pacific Journal of Operational Research, 1999, vol. 16, no 1, p. 1-22. Disponible en: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.35.994

BEJARANO, Gissella. Planificación de horarios del personal de cirugía de un hospital del Estado aplicando algoritmos genéticos (Time Tabling Problem). 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

BURKE, E. K.; KINGSTON, Jeffrey; DE WERRA, D. 5.6: Applications to Timetabling. Handbook of graph theory, 2004, p. 445. Disponible en: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.188.1458

CABEZAS GARCIA, Jose Javier. Diseno e implementacion de una heurística para resolver el problema de calendarización de horarios para universidades. 2009. Tesis Doctoral. Disponible en: https://www.dspace.espol.edu.ec/bitstream/123456789/12233/3/tesisxcimp.pdf

CARDEMIL, Andrés. Optimización de fixtures deportivos: Estado del arte y un algoritmo tabu search para el traveling tournament problem. MasterÕs thesis, Universidad de Buenos Aires, Departamento de Computación, Buenos Aires, 2002. Disponible en: http://old.dii.uchile.cl/~gduran/docs/tesis/tesis\_andres.pdf

CHAMBERS, Lance. (ed.). Practical handbook of genetic algorithms: complex coding systems. CRC press, 1998. Disponible en: https://books.google.es/books?hl=es&lr=&id=04a\_fHI9F1YC&oi=fnd&pg=PA17&dq=Practical+handbook+of+genetic+algorithms:+complex+coding+systems.+CRC+press&ots=d9p8LxX8MC&sig=pY-BHWmPkaDzwgLsjmscUoDGFas#v=onepage&q=Practical%20handbook%20of%20genetic%20algorithms%3A%20complex%20coding%20systems.%20CRC%20press&f=false

DIAZ, Belarmino Adenso. Optimización heurística y redes neuronales: en dirección de operaciones e Ingeniería. 1996. Disponible en: http://biblioteca.ucm.es/tesis/19972000/S/3/S3024907.pdf

DORIGO, Marco; BIRATTARI, Mauro; STUTZLE, Thomas. Ant colony optimization. Computational Intelligence Magazine, IEEE, 2006, vol. 1, no 4, p. 28-39. Disponible en: http://ieeexplore.ieee.org/xpls/abs\_all.jsp?arnumber=4129846

DOWSLAND, Kathryn A.; ADENSO-DÍAZ, Belarmino. Diseño de Heurísticas y Fundamentos del Recocido Simulado. Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial, 2003, vol. 7, no 19, p. 93-102. Disponible en: http://sci2s.ugr.es/docencia/metaheuristicas/Enfriamiento-simulado.pdf

FEO, Thomas A.; RESENDE, Mauricio GC. A probabilistic heuristic for a computationally difficult set covering problem. Operations research letters, 1989, vol. 8, no 2, p. 67-71. Disponible en: http://www.gardeux-vincent.eu/These/Papiers/Bibli2/Feo89.pdf

FRANCO FLORES, Abel. Estudio comparativo de algoritmos genéticos y algoritmos de búsqueda tabú para la resolución del Flow Shop Problem. 2009. Disponible en: http://upcommons.upc.edu/pfc/bitstream/2099.1/7484/1/MEMORIA\_Y\_PRESUPUESTO.pdf

FRANCO, John Fredy, et al. Problema de asignación óptima de salones resuelto con Búsqueda Tabú. Ingeniería y desarrollo, 2008, no 24, p. 149-175. Disponible en: http://www.scielo.org.co/scielo.php?script=sci\_arttext&pid=S0122-34612008000200011&lng=en&tlng=es

GLOVER, Fred. Tabu search-part I. ORSA Journal on computing, 1989, vol. 1, no 3, p. 190-206. Disponible en: http://joc.journal.informs.org/content/1/3/190.short

GLOVER, Fred; LAGUNA, Manuel. Tabu search. Springer US, 1999. Disponible en: http://dl.acm.org/citation.cfm?id=549765

GÓMEZ TORO, Jennifer Andrea; VANEGAS CASTELLANOS, Juan David; ZULUAGA GÓMEZ, Natalia. 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

GUERRA, Mauricio Andres; PARDO, Erwin Hamid; SALAS, Roberto Emilio. Problema del School Timetabling y algoritmos genéticos: una revisión. Vínculos, 2014, vol. 10, no 2, p. 259-276. Disponible en: http://revistavinculos.udistrital.edu.co/files/2013/09/Problema-del-School-Timetabling-y-algoritmos-geneticos.pdf

JACOBSON, Ivar; BOOCH, Grady; RUMBAUGH, James. El proceso unificado de desarrollo de software. Reading: Addison Wesley, 2000. Disponible en: http://unpprogespcn.com/material/RUP/ElProcesoUnificadodeDesarrollodeSoftware.pdf

KRUCHTEN, Philippe. The rational unified process: an introduction. Addison-Wesley Professional, 2004. Disponible en: https://books.google.es/books?hl=es&lr=&id=RYCMx6o47pMC&oi=fnd&pg=PR13&dq=The+rational+unified+process:+an+introduction&ots=h1cnDUPbTe&sig=w-yODFMv2dbabdK5RhqlDMCowcQ#v=onepage&q=The%20rational%20unified%20process%3A%20an%20introduction&f=false

LARROSA, Javier; MESEGUER, Pedro. Restricciones blandas: modelos y algoritmos. Inteligencia Artificial, 2003, vol. 20. Disponible en: http://redalyc.uaemex.mx/redalyc/src/inicio/ArtPdfRed.jsp?iCve=92572006

LÜ, Zhipeng; HAO, Jin-Kao. Adaptive tabu search for course timetabling.European Journal of Operational Research, 2010, vol. 200, no 1, p. 235-244. Disponible en: http://www.info.univ-angers.fr/pub/hao/papers/EJOR08.pdf

MARTÍNEZ, Alejandro; MARTÍNEZ, Raúl. Guía a Rational Unified Process.Escuela Politécnica Superior de Albacete–Universidad de Castilla la Mancha, 2002.Disponible en: https://93377ec7-a-62cb3a1a-s-sites.googlegroups.com/site/softqma/programa/unidad-iv-metodologias-utilizadas-para-el-desarrollo-del-software/Trabajo-GuiaRUP.pdf

MARTINEZ, Francisco, et al. Timetabling Académico Usando Algoritmos Genéticos y Programación Celular. Universidad Autónoma de Zacatecas. Departamento de Ingeniería en Computación. Disponible en: http://ingsw.ccbas.uaa.mx/sitio/images/pdfpublicaciones/artiCoNaCiCo05-20.pdf

MAYORDOMO, Elvira. NP-completos. Universidad de Zaragoza. Zaragoza–España. Disponible en: http://webdiis.unizar.es/~elvira/mac/npcompletos.pdf

MEJÍA CABALLERO, José María; PATERNINA ARBOLEDA, Carlos Daniel. Asignación de horarios de clases universitarias mediante algoritmos evolutivos. 2009. Tesis Doctoral. Disponible en: http://manglar.uninorte.edu.co/bitstream/handle/10584/80/84032706.pdf?sequence=1&isAllowed=y

METROPOLIS, Nicholas, et al. Equation of state calculations by fast computing machines. The journal of chemical physics, 1953, vol. 21, no 6, p. 1087-1092. Disponible en: http://www.stat.cmu.edu/~acthomas/724/Metropolis.pdf

MINETTI, Gabriela F. Una solución de computación evolutiva para el TSP, su posible aplicación en las organizaciones. 2000. Tesis Doctoral. Facultad de Informática. Disponible en: http://sedici.unlp.edu.ar/bitstream/handle/10915/4059/Documento\_completo.pdf?sequence=13

NAUPARI, Raúl; ROSALES, Gissela. 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.

PÉREZ, Fernando. 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

PINO, R., et al. Application of GRASP methodology to Vehicle Routing Problem (VRP). Disponible en: http://elrond.informatik.tu-freiberg.de/papers/WorldComp2012/ICA6018.pdf

PITOL, Fermín. Uso de Algoritmos Evolutivos para resolver el Problema de Asignación de Horarios Escolares de la Facultad de Psicologia de la Universidad Veracruzana. Facultad de Física e Inteligencia Artificial. Disponible en: http://www.lania.mx/~emezura/util/files/tesis\_FerminFinal.pdf

RESTREPO, Gerley, et al. Modelo para la asignación de recursos académicos en instituciones educativas utilizando técnicas metaheurísticas. Avances en Sistemas e Informática; Vol. 8, núm. 3 (2011); 111-124 Avances en Sistemas e Informática; Vol. 8, núm. 3 (2011); 111-124 1909-0056 1657-7663, 2012. Disponible en: http://digital.unal.edu.co/index.php/avances/article/view/22350

SUÁREZ, Joseph Gallart, et al. Generación Inteligente de Horarios empleando heurísticas GRASP con Búsqueda Tabú para la Pontifica Universidad Católica del Perú. Revista Ingeniería Informática, 2010, vol. 1, no 1, p. 15-24. Disponible en: http://www.revistas.pucp.edu.pe/rii/sites/revistas.pucp.edu.pe.rii/files/Joseph\_Gallart.pdf

TOLMOS PIÑERO, Piedad. Introducción a los algoritmos genéticos y sus aplicaciones. Universidad Rey Juan Carlos, Servicio de Publicaciones, 2003. Disponible en: http://www.uv.es/asepuma/X/J24C.pdf

TORO, Eliana Mirledy; TABARES, Pompilio; GRANADA, Mauricio. Método de colonia de hormigas aplicado a la solución del problema de asignación generalizada. Revista Tecnura, 2004, vol. 8, no 15, p. 66-76. Disponible en: http://tecnura.udistrital.edu.co/ojs/index.php/revista/article/view/151

VÁZQUEZ ESPÍ, Mariano. Recocido simulado: un nuevo algoritmo para la optimización de estructuras. 1994. Tesis Doctoral. Arquitectura. Disponible en: http://oa.upm.es/968/

WREN, Anthony. Scheduling, timetabling and rostering—a special relationship?. En Practice and theory of automated timetabling. Springer Berlin Heidelberg, 1996. p. 46-75.

1. NAUPARI, Raúl; ROSALES, Gissela. 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. [↑](#footnote-ref-1)
2. BURKE, E. K.; KINGSTON, Jeffrey; DE WERRA, D. 5.6: Applications to Timetabling. Handbook of graph theory, 2004, p. 445. [↑](#footnote-ref-2)
3. METROPOLIS, Nicholas, et al. Equation of state calculations by fast computing machines. The journal of chemical physics, 1953, vol. 21, no 6, p. 1087-1092. [↑](#footnote-ref-3)
4. DOWSLAND, Kathryn A.; ADENSO-DÍAZ, Belarmino. Diseño de Heurísticas y Fundamentos del Recocido Simulado. Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial, 2003, vol. 7, no 19, p. 93-102. [↑](#footnote-ref-4)
5. GÓMEZ TORO, Jennifer Andrea; VANEGAS CASTELLANOS, Juan David; ZULUAGA GÓMEZ, Natalia. Diseño e implementación de un algoritmo para dar solución al problema de asignación de salones (Timetabling) usando el método de colonia de hormigas. 2009. [↑](#footnote-ref-5)
6. VÁZQUEZ ESPÍ, Mariano. Recocido simulado: un nuevo algoritmo para la optimización de estructuras. 1994. Tesis Doctoral. Arquitectura. [↑](#footnote-ref-6)
7. GÓMEZ TORO, Jennifer Andrea; VANEGAS CASTELLANOS, Juan David; ZULUAGA GÓMEZ, Natalia. Diseño e implementación de un algoritmo para dar solución al problema de asignación de salones (Timetabling) usando el método de colonia de hormigas. 2009. [↑](#footnote-ref-7)
8. ABRAMSON, David. Constructing school timetables using simulated annealing: sequential and parallel algorithms. Management science, 1991, vol. 37, no 1, p. 98-113. [↑](#footnote-ref-8)
9. GLOVER, Fred. Tabu search-part I. ORSA Journal on computing, 1989, vol. 1, no 3, p. 190-206. [↑](#footnote-ref-9)
10. CABEZAS GARCIA, Jose Javier. Diseno e implementacion de una heurística para resolver el problema de calendarización de horarios para universidades. 2009. Tesis Doctoral [↑](#footnote-ref-10)
11. FRANCO FLORES, Abel. Estudio comparativo de algoritmos genéticos y algoritmos de búsqueda tabú para la resolución del Flow Shop Problem. 2009. [↑](#footnote-ref-11)
12. GLOVER, Op. cit. [↑](#footnote-ref-12)
13. RESTREPO, Gerley, et al. Modelo para la asignación de recursos académicos en instituciones educativas utilizando técnicas metaheurísticas. Avances en Sistemas e Informática; Vol. 8, núm. 3 (2011); [↑](#footnote-ref-13)
14. SUÁREZ, Joseph Gallart, et al. Generación Inteligente de Horarios empleando heurísticas GRASP con Búsqueda Tabú para la Pontifica Universidad Católica del Perú. Revista Ingeniería Informática, 2010, vol. 1, no 1, p. 15-24. [↑](#footnote-ref-14)
15. CARDEMIL, Andrés. Optimización de fixtures deportivos: Estado del arte y un algoritmo tabu search para el traveling tournament problem. MasterÕs thesis, Universidad de Buenos Aires, Departamento de Computación, Buenos Aires, 2002. [↑](#footnote-ref-15)
16. DORIGO, Marco; BIRATTARI, Mauro; STUTZLE, Thomas. Ant colony optimization. Computational Intelligence Magazine, IEEE, 2006, vol. 1, no 4, p. 28-39. [↑](#footnote-ref-16)
17. GÓMEZ TORO, Jennifer Andrea; VANEGAS CASTELLANOS, Juan David; ZULUAGA GÓMEZ, Natalia. Diseño e implementación de un algoritmo para dar solución al problema de asignación de salones (Timetabling) usando el método de colonia de hormigas. 2009. [↑](#footnote-ref-17)
18. RESTREPO, Op. cit. [↑](#footnote-ref-18)
19. GÓMEZ TORO, Op. cit. [↑](#footnote-ref-19)
20. GÓMEZ TORO, Jennifer Andrea; VANEGAS CASTELLANOS, Juan David; ZULUAGA GÓMEZ, Natalia. Diseño e implementación de un algoritmo para dar solución al problema de asignación de salones (Timetabling) usando el método de colonia de hormigas. 2009. [↑](#footnote-ref-20)
21. TORO, Eliana Mirledy; TABARES, Pompilio; GRANADA, Mauricio. Método de colonia de hormigas aplicado a la solución del problema de asignación generalizada. Revista Tecnura, 2004, vol. 8, no 15, p. 66-76. [↑](#footnote-ref-21)
22. FEO, Thomas A.; RESENDE, Mauricio GC. A probabilistic heuristic for a computationally difficult set covering problem. Operations research letters, 1989, vol. 8, no 2, p. 67-71. [↑](#footnote-ref-22)
23. PÉREZ, Fernando. Una metodología de solución basada en la metaheurística Grasp para el problema de diseño de red con incertidumbre. [↑](#footnote-ref-23)
24. SUÁREZ, Joseph Gallart, et al. Generación Inteligente de Horarios empleando heurísticas GRASP con Búsqueda Tabú para la Pontifica Universidad Católica del Perú. Revista Ingeniería Informática, 2010, vol. 1, no 1, p. 15-24. [↑](#footnote-ref-24)
25. PINO, R., et al. Application of GRASP methodology to Vehicle Routing Problem (VRP). [↑](#footnote-ref-25)
26. CABEZAS GARCIA, Jose Javier. Diseno e implementacion de una heurística para resolver el problema de calendarización de horarios para universidades. 2009. Tesis Doctoral. [↑](#footnote-ref-26)
27. DIAZ, Belarmino Adenso. Optimización heurística y redes neuronales: en dirección de operaciones e Ingeniería. 1996. [↑](#footnote-ref-27)
28. CHAMBERS, Lance. (ed.). Practical handbook of genetic algorithms: complex coding systems. CRC press, 1998. [↑](#footnote-ref-28)
29. TOLMOS PIÑERO, Piedad. Introducción a los algoritmos genéticos y sus aplicaciones. Universidad Rey Juan Carlos, Servicio de Publicaciones, 2003. [↑](#footnote-ref-29)
30. GUERRA, Mauricio Andres; PARDO, Erwin Hamid; SALAS, Roberto Emilio. Problema del School Timetabling y algoritmos genéticos: una revisión. Vínculos, 2014, vol. 10, no 2, p. 259-276. [↑](#footnote-ref-30)
31. Ibid. [↑](#footnote-ref-31)
32. Ibid. [↑](#footnote-ref-32)
33. PITOL, Fermín. Uso de Algoritmos Evolutivos para resolver el Problema de Asignación de Horarios Escolares de la Facultad de Psicologia de la Universidad Veracruzana. Facultad de Física e Inteligencia Artificial. [↑](#footnote-ref-33)
34. NAUPARI, Raúl; ROSALES, Gissela. 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. [↑](#footnote-ref-34)
35. Ibid. [↑](#footnote-ref-35)
36. NAUPARI, Raúl; ROSALES, Gissela. 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. [↑](#footnote-ref-36)
37. Ibid. [↑](#footnote-ref-37)
38. GUERRA, Mauricio Andres; PARDO, Erwin Hamid; SALAS, Roberto Emilio. Problema del School Timetabling y algoritmos genéticos: una revisión. Vínculos, 2014, vol. 10, no 2, p. 259-276. [↑](#footnote-ref-38)
39. GUERRA, Mauricio Andres; PARDO, Erwin Hamid; SALAS, Roberto Emilio. Problema del School Timetabling y algoritmos genéticos: una revisión. Vínculos, 2014, vol. 10, no 2, p. 259-276. [↑](#footnote-ref-39)
40. PINO, R., et al. Application of GRASP methodology to Vehicle Routing Problem (VRP). [↑](#footnote-ref-40)
41. MARTINEZ, Francisco, et al. Timetabling Académico Usando Algoritmos Genéticos y Programación Celular. Universidad Autónoma de Zacatecas. Departamento de Ingeniería en Computación. [↑](#footnote-ref-41)
42. GUERRA; PARDO, Op.cit. [↑](#footnote-ref-42)
43. MARTINEZ, Op. Cit. [↑](#footnote-ref-43)
44. NAUPARI, Raúl; ROSALES, Gissela. 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. [↑](#footnote-ref-44)
45. GUERRA, Mauricio Andres; PARDO, Erwin Hamid; SALAS, Roberto Emilio. Problema del School Timetabling y algoritmos genéticos: una revisión. Vínculos, 2014, vol. 10, no 2, p. 259-276. [↑](#footnote-ref-45)
46. KRUCHTEN, Philippe. The rational unified process: an introduction. Addison-Wesley Professional, 2004. [↑](#footnote-ref-46)
47. MARTÍNEZ, Alejandro; MARTÍNEZ, Raúl. Guía a Rational Unified Process.Escuela Politécnica Superior de Albacete–Universidad de Castilla la Mancha, 2002. [↑](#footnote-ref-47)
48. LÜ, Zhipeng; HAO, Jin-Kao. Adaptive tabu search for course timetabling.European Journal of Operational Research, 2010, vol. 200, no 1, p. 235-244. [↑](#footnote-ref-48)
49. WREN, Anthony. Scheduling, timetabling and rostering—a special relationship?. En Practice and theory of automated timetabling. Springer Berlin Heidelberg, 1996. p. 46-75. [↑](#footnote-ref-49)
50. CHAMBERS, Lance. (ed.). Practical handbook of genetic algorithms: complex coding systems. CRC press, 1998. [↑](#footnote-ref-50)
51. GUERRA, Mauricio Andres; PARDO, Erwin Hamid; SALAS, Roberto Emilio. Problema del School Timetabling y algoritmos genéticos: una revisión. Vínculos, 2014, vol. 10, no 2, p. 259-276. [↑](#footnote-ref-51)
52. FRANCO, John Fredy, et al. Problema de asignación óptima de salones resuelto con Búsqueda Tabú. ingeniería y desarrollo, 2008, no 24, p. 149-175. [↑](#footnote-ref-52)
53. LARROSA, Javier; MESEGUER, Pedro. Restricciones blandas: modelos y algoritmos. Inteligencia Artificial, 2003, vol. 20. [↑](#footnote-ref-53)
54. NAUPARI, Raúl; ROSALES, Gissela. 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. [↑](#footnote-ref-54)
55. MAYORDOMO, Elvira. NP-completos. Universidad de Zaragoza. Zaragoza–España. [↑](#footnote-ref-55)
56. PINO, R., et al. Application of GRASP methodology to Vehicle Routing Problem (VRP). [↑](#footnote-ref-56)
57. MAYORDOMO, Op. Cit. [↑](#footnote-ref-57)
58. BEJARANO, Gissella. Planificación de horarios del personal de cirugía de un hospital del Estado aplicando algoritmos genéticos (Time Tabling Problem). 2011. [↑](#footnote-ref-58)
59. JACOBSON, Ivar; BOOCH, Grady; RUMBAUGH, James. El proceso unificado de desarrollo de software. Reading: Addison Wesley, 2000. [↑](#footnote-ref-59)
60. JACOBSON, Ivar; BOOCH, Grady; RUMBAUGH, James. El proceso unificado de desarrollo de software. Reading: Addison Wesley, 2000. [↑](#footnote-ref-60)
61. NAUPARI, Raúl; ROSALES, Gissela. 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. [↑](#footnote-ref-61)
62. CABEZAS GARCIA, Jose Javier. Diseno e implementacion de una heurística para resolver el problema de calendarización de horarios para universidades. 2009. Tesis Doctoral. [↑](#footnote-ref-62)
63. CABEZAS GARCIA, Jose Javier. Diseno e implementacion de una heurística para resolver el problema de calendarización de horarios para universidades. 2009. Tesis Doctoral. [↑](#footnote-ref-63)
64. MINETTI, Gabriela F. Una solución de computación evolutiva para el TSP, su posible aplicación en las organizaciones. 2000. Tesis Doctoral. Facultad de Informática. [↑](#footnote-ref-64)
65. Ibid. [↑](#footnote-ref-65)
66. MINETTI, Gabriela F. Una solución de computación evolutiva para el TSP, su posible aplicación en las organizaciones. 2000. Tesis Doctoral. Facultad de Informática. [↑](#footnote-ref-66)