Diseño de una herramienta de software que asigne eficientemente las aulas para todos los cursos de pregrado de la Universidad Tecnológica de Bolívar usando un algoritmo genético.

Daniel Rafael Colón Montalvo

Universidad Tecnológica de Bolívar

Programa de Ingeniería de Sistemas

Cartagena, Colombia

Junio de 2012

Diseño de una herramienta de software que asigne eficientemente las aulas para todos los cursos de pregrado de la Universidad Tecnológica de Bolívar usando un algoritmo genético.

Daniel Rafael Colón Montalvo

DIRECTOR:

Moisés Quintana Álvarez.

Magister en Informática.

Universidad Tecnológica de Bolívar

Programa de Ingeniería de Sistemas

Cartagena, Colombia

Junio de 2012

Cartagena de Indias D.T. y C. 29 de Junio 2012

Señores
COMITÉ CURRICULAR DE EVALUACIÓN DE PROYECTOS
Programa de Ingeniería de Sistemas
Facultad de Ingeniería
Universidad Tecnológica de Bolívar
Ciudad

Apreciados señores,

Comedidamente les solicito someter a consideración la propuesta de trabajo de grado titulado "**Diseño de una herramienta de software que asigne eficientemente las aulas para todos los cursos de pregrado de la Universidad Tecnológica de Bolívar usando un algoritmo genético**", realizado por el estudiante DANIEL RAFAEL COLON MONTALVO, para optar al título de Ingeniero de Sistemas.

Cordialmente,

_____

Moisés Quintana Álvarez.
Dir. Del Trabajo de Grado.

Cartagena de Indias D. T. y C., Junio 29 de 2012

Señores:
Comité curricular del programa de Ingeniería de Sistemas
Universidad Tecnológica de Bolívar
Ciudad

Cordial saludo:

La presente tiene como fin el presentar la propuesta de trabajo de grado titulada "**Diseño de una herramienta de software que asigne eficientemente las aulas para todos los cursos de pregrado de la Universidad Tecnológica de Bolívar usando un algoritmo genético**", la cual desarrollaré como estudiante del programa de Ingeniería de Sistemas de la UTB.

Agradeciendo de antemano su atención.

Atentamente,

_____

Daniel Rafael Colón Montalvo.
CC No. 1.143.339.653 de Cartagena.

Nota de aceptación:

_____

_____

_____

_____
Firma del presidente del jurado

_____
Firma jurado

_____
Firma jurado

Cartagena, 29 de Junio de 2012

DEDICATORIA

A mi madre y a mi abuela por su apoyo y amor incondicional, por creer ciegamente en mí y estar ahí en todo momento.

Daniel Colón Montalvo.

AGRADECIMIENTOS

El autor expresa sus agradecimientos a:

Moisés Quintana Álvarez, Magister en informática, por todo su apoyo incondicional en el desarrollo de esta investigación, por el tiempo dedicado y sus bien intencionadas recomendaciones.

Sergio Leottau Sanmiguel y Jaime Padrón Cano, por toda la ayuda recibida de su parte en la elaboración de este proyecto. Sus opiniones profesionales y personales en este proceso han sido de vital importancia.

A José Barrios y Omer Salcedo, ingenieros de sistemas, del departamento SIRIUS en la Universidad Tecnológica de Bolívar. Gracias por facilitar toda la información requerida para la elaboración del presente proyecto y dedicarle un espacio de su tiempo al mismo.

Al cuerpo de docentes del programa de Ingeniería de Sistemas de la Universidad Tecnológica de Bolívar, por todas las enseñanzas adquiridas a nivel profesional y personal.

CONTENIDO

LISTA DE TABLAS

LISTA DE FIGURAS

LISTA DE ECUACIONES

ANEXOS

INTRODUCCION

La asignación de aulas es uno de los sub-problemas a solucionar en la programación de horarios universitarios (*University Curse Timetable Programming*, UCTP), en conjunto con la asignación de docentes y la programación de los cursos (Chaudhuri & Kajal, 2010). Este tipo de soluciones implican una gran cantidad de operaciones combinatorias, lo que conlleva a que sean algoritmos de un gran gasto computacional. Para la solución este tipo de problemas es común emplear búsquedas de optimización, las cuales no necesariamente arrojen un resultado óptimo, pero pueden ofrecer una buena solución que dependerá del tiempo permitido para su ejecución. Dentro de éstas técnicas se encuentran los algoritmos genéticos (Peña & Zulmenzu, 2010).

Este trabajo busca optimizar la asignación de aulas para todos los cursos de pregrado en la Universidad Tecnológica de Bolívar, por medio del uso de un algoritmo genético que minimice el desperdicio de cupos.

En la primera parte de la investigación se muestra en detalle el diseño del algoritmo genético que será usado para resolver el problema. En la segunda parte se encuentra el diseño de la herramienta de software que utilizará el algoritmo genético para realizar la asignación eficiente de aulas.

En el capítulo III se encuentra en detalle la especificación del prototipo elaborado, de la herramienta de software diseñada para la asignación de aulas. Y por último, en el capítulo IV se muestran los resultados al probar la herramienta prototipo.

1. OBJETIVOS

OBJETIVO GENERAL:

Diseñar y construir un prototipo de una herramienta de software que asigne eficientemente todas las clases de pregrado de la Universidad Tecnológica de Bolívar, minimizando el desperdicio de cupos por medio de la utilización de un algoritmo genético.

OBJETIVOS ESPECÍFICOS:

- Realizar un análisis del proceso de asignación de aulas para los cursos de pregrado en la Universidad Tecnológica de Bolívar con el fin de comprender el dominio del problema y así poder ofrecer una solución factible.
- Diseñar un algoritmo genético que minimice el desperdicio de cupos al realizar la asignación de aulas para las clases de pregrado en la Universidad Tecnológica de Bolívar.
- Modelar una herramienta de software que optimice por medio del algoritmo genético diseñado la asignación de aulas para todos los cursos de pregrado en la Universidad Tecnológica de Bolívar.

- Construir un prototipo del sistema de asignación de aulas, usando el lenguaje de programación java, que permita realizar la programación eficiente de aulas para las clases de tipo teórico.
- Evaluar la herramienta prototipo, comparando el desperdicio de cupos al asignar las aulas para todas las clases de pregrado de tipo teórico del primer semestre del 2012, con el desperdicio de cupos generado por la programación del mismo tipo de clases realizada por la universidad.

2. PROBLEMA DE INVESTIGACIÓN

2.1. DESCRIPCIÓN DEL PROBLEMA

La Universidad Tecnológica de Bolívar está ubicada en Cartagena de Indias D. T y C. (Colombia, América del Sur), actualmente posee 2 campus educativos: campus casa Lemaitre y campus Tecnológico. El campus casa Lemaitre está situado en la calle del Bouquet del barrio Manga , en él se imparten cursos de programas de formación avanzada como postgrados, maestrías y cursos de verano; El campus tecnológico, por otra parte, se encuentra en el parque industrial Carlos Vélez Pombo al sur de la cuidad en el kilómetro 1 vía a Turbaco. Es el lugar más importante para el desarrollo académico de la institución, puesto que en él se dictan aproximadamente 2000 cursos semanales a los cuales asisten más de 4000 alumnos, pertenecientes a programas de ingenierías, ciencias administrativas y humanas, programas técnicos y tecnológicos, maestrías y especializaciones.

La programación de cada uno de los cursos impartidos en la universidad está a cargo de los profesionales de apoyo, quienes determinan la cantidad de cursos ofertados en un período académico teniendo en cuenta el plan de estudios de cada carrera, datos estadísticos e información del número de estudiantes inscritos a cursar primer semestre.

El proceso de creación de la grilla académica inicia con la información del número de estudiantes que entrarán al primer semestre de la universidad, proporcionada por el área financiera. Por otra parte, la población académica para los demás niveles se toma de acuerdo a información estadística proporcionada por la dirección académica de la universidad, dichos resultados históricos arrojan que aproximadamente el 70% de los estudiantes logra avanzar al próximo nivel sin problemas y un 30% de los estudiantes repetirán materias. Con las cifras mencionadas anteriormente se establece el número de cursos que deben ser ofertados para una misma materia del programa académico. Normalmente esta regla no se fija para los cursos de electivas, puesto que los estudiantes pueden seleccionar las de su conveniencia. Dependiendo el número de cursos ofertados para una materia en particular, se establecen los grupos que conformarán la grilla.

Cada profesional de apoyo tiene a su cargo la programación de una o varias carreras, por lo cual sus decisiones pueden diferir de acuerdo al comportamiento estadístico particular de cada carrera, sin embargo las preferencias generales de asignación son:

- El 70% de los cursos pertenecientes a materias de primer a cuarto nivel deben tener clases en las mañanas.

- Los cursos pertenecientes a materias de séptimo semestre en adelante deben estar programados en las tardes.
- La mayoría de los laboratorios deben estar programados después de la clase teórica.
- El tiempo entre bloques de una misma asignatura debe ser de por lo menos 1 día.

Además de las preferencias de asignación, el profesional de apoyo debe asignar los horarios de los cursos teniendo en cuenta que solo posee un conjunto de aulas y laboratorios disponibles para las actividades académicas de su programa. Cada materia ofertada posee un conjunto de docentes habilitados por cada director de programa, los docentes pueden ser de tiempo completo o catedráticos (externos) y ambos poseen un conjunto de horas semanales en las que pueden impartir clases. Es deber del profesional de apoyo seleccionar cada docente que impartirá un curso.

Al final de este proceso, cada curso contiene los siguientes atributos: docente que impartirá la asignatura, horario del curso acorde con las preferencias, grupo al que pertenece el curso, cupo máximo de estudiantes, tipo de curso (presencial o virtual) y recurso necesario (laboratorio, aula de sistemas, aula con audiovisuales, o aula con mesa). Esta información es enviada al departamento que opera el

sistema integrado de recursos de información universitaria para el servicio (SIRIUS).

Una vez la grilla académica ha sido creada por los profesionales de apoyo y enviada a SIRIUS, el encargado de asignación de recursos físicos selecciona el aula más conveniente para cada curso ofertado, teniendo en cuenta los siguientes criterios:

- Los estudiantes de primer nivel tendrán asignados todos sus cursos en una misma aula.
- El número de cursos dictados en simultánea no debe superar la capacidad de aulas.

En caso de presentar alguna inconsistencia en la asignación o no disponer del recurso solicitado para un curso, la grilla es devuelta al profesional de apoyo con el fin de reprogramar los cursos.

La asignación de recurso físico es una actividad dispendiosa, puesto que es necesario asignar de forma manual cada una de las 65 aulas o cada uno de los 15 laboratorios para cada curso de pregrado, que en total serían más de 2000. Por otra parte SIRIUS no optimiza el uso de aulas dependiendo el cupo máximo de

estudiantes, ya que el sistema solo ofrece las aulas disponibles a una hora determinada con el fin de evitar cruces de horarios. Esta debilidad del sistema ocasiona sobrecupo en algunas aulas, mientras que en otros casos el número de estudiantes es muy inferior a la capacidad de las mismas.

2.2. FORMULACIÓN DEL PROBLEMA.

¿Cómo automatizar el proceso de asignación de aulas y laboratorios para los cursos de pregrado en la Universidad Tecnológica de Bolívar, minimizando el desperdicio de cupos?

3. JUSTIFICACIÓN

La asignación de aulas y laboratorios es un proceso de vital importancia en la Universidad Tecnológica de Bolívar, puesto que es necesario garantizar un espacio físico para cada curso presencial, teniendo en cuenta que la principal actividad económica de la empresa es la enseñanza académica. Además la universidad posee prestigio a nivel regional, lo cual implica que todos los cursos ofertados deben impartirse en espacios físicos que cumplan con los estándares mínimos de calidad que exige el ministerio nacional para ratificar la acreditación institucional.

Cada aula de clase o laboratorio posee un número de estudiantes permitidos: por tal motivo existen aulas con diversas capacidades y recursos variados, dependiendo los requerimientos de los cursos programados. Para mantener la comodidad de los estudiantes y docentes en las aulas, es necesario respetar el límite de cupos al momento de hacer la asignación manual; sin embargo esto no es tarea fácil ya que en la oficina de profesionales de apoyo se establecen los horarios de los cursos impartidos en un período académico, pero no se tiene en cuenta la optimización del recurso físico, puesto que lo anterior es responsabilidad de otro departamento (Entrevista oficina profesionales de apoyo, no referenciado).

En el departamento encargado de administrar el sistema integrado de recursos de información universitaria para el servicio (SIRIUS), es ingresada toda la programación de los cursos ofertados en el periodo académico actual y al mismo tiempo, el ingeniero José Barrios se encarga de asignar manualmente y a su criterio las aulas para cada curso ofertado, tarea dispendiosa que requiere de semanas de trabajo para garantizar las preferencias de asignación. Es necesario hacer esta tarea de forma manual puesto que SIRIUS se limita a ofrecer un espacio físico disponible en una franja horaria determinada, pero no supervisa que el aula posea el cupo máximo necesario para impartir un curso (Entrevista José Barrios, SIRIUS, No referenciado).

Debido a lo dispendioso del proceso, la asignación de aulas se lleva acabo semanas antes de que los cursos sean ofertados en la plataforma, motivo por el cual no se tiene en cuenta el número de estudiantes que realmente se han inscrito a un curso, sino que para la programación de aulas y laboratorios siempre se trabaja con el número máximo de personas permitidas. Esta situación hace que un espacio físico con 40 cupos, sea asignado a cursos en los que solo se han inscrito menos de la mitad de estudiantes, desperdiciando un aula o laboratorio que podría ser usado por otro curso con mayor número de asistentes.

Controlar el desperdicio de cupos disponibles en un aula o laboratorio permitiría reducir al máximo el sobrecupo en los espacios físicos y facilitaría la inclusión de

un nuevo curso a la programación. El procedimiento de control anteriormente mencionado no es posible en la actualidad debido a que la asignación manual de aulas y laboratorios conlleva semanas de trabajo, con lo cual no se podría concluir a tiempo. La elaboración de una herramienta computacional, capaz de automatizar el proceso de asignación y minimizar el desperdicio de cupos en menos de un día, permitiría elaborar programaciones de manera ágil una vez se han conocido el número de estudiantes inscritos a un curso, permitiéndole a la institución optimizar la programación de los salones y laboratorios.

4. ASPECTOS METODOLÓGICOS

4.1. TIPO DE INVESTIGACIÓN.

En el proceso de investigación, se utilizará un estudio de tipo descriptivo-relacional. "La investigación descriptiva busca especificar las propiedades, características y rasgos importantes de cualquier fenómeno que se analice; mientras que los estudios correlaciónales tienen como propósito conocer la relación que exista entre dos o más conceptos, categorías o variables en un contexto particular" (Hernandez & Fernandez, 2006).

4.2. METODOLOGÍA DE INVESTIGACIÓN.

La metodología que se ha adoptado para la investigación es la investigación de diseño científico, aplicada en sistemas de información, puesto que permite integrar el diseño, la construcción y evaluación de un artefacto de ingeniería (en este caso un prototipo) con el fundamento investigativo científico (Hevner Allan, 2004). Esta metodología se basa en el cumplimiento de los siguientes 7 postulados:

1. **Crear un artefacto útil y viable:** El producto de la investigación debe servir para el propósito que fue creado, además su implementación debe ser factible.

2. **El artefacto que ha sido diseñado debe solucionar el problema del dominio de la investigación**: es necesario que el producto que origine la investigación contribuya a la solución del problema, en el que se ha centrado el proyecto.

3. **El artefacto debe ser evaluado:** se debe demostrar que el producto de la investigación realmente contribuye a la solución del problema.

4. **Debe ser efectivo y resolver el problema de forma novedosa:** además de solucionar un problema del mundo real, el artefacto de ingeniería debe tener una manera particular de hallar la solución que no se haya abordado antes.

5. **El artefacto debe estar definido, representado, esquematizado, coherente y consistente formalmente:** se deberá definir el producto por medio del uso de lenguajes de representación formal.

6. **Es necesario un proceso de investigación para hallar una solución efectiva:** para desarrollar el artefacto de ingeniería, se debe hacer una investigación extensiva que permita encontrar la mejor forma de abordar el problema de dominio.

7. **El hallazgo debe ser comunicado:** es necesario que la comunidad académica pueda acceder a los resultados de la investigación.

4.3. METODOLOGÍA DE TRABAJO.

Para garantizar el cumplimiento de las pautas inherentes a la metodología seleccionada, se tiene previsto desarrollar el proyecto en las siguientes fases:

1. **Análisis**: En esta etapa, se recogerán todos los requerimientos funcionales y no funcionales que posee el sistema, además de hará un estudio a fondo del proceso y los factores que intervienen en él.

2. **Investigación**: se hará una búsqueda de información científica que permita sentar las bases para la construcción del algoritmo.

3. **Diseño:** crear un algoritmo genético que programe la asignación de aulas para los cursos de pregrado de la Universidad Tecnológica de Bolívar, minimizando los cursos con sobrecupo.

4. **Prototipo:** diseño e implementación de una herramienta prototipo que permita usar el algoritmo diseñado.

5. **Evaluación:** probar el funcionamiento del algoritmo a través del prototipo construido por medio de pruebas unitarias y funcionales.

5. MARCO REFERENCIAL.

5.1. MARCO TEÓRICO.

5.1.1. Algoritmos heurísticos.

Científicamente, el término heurística fue mencionado por primera vez por el matemático George Polya, en su libro *How to solve it* . Con este término, Polya intentaba expresar las reglas con la que los seres humanos gestionamos el conocimiento común, y que además podían ser simplificados en los siguientes puntos:

- Buscar un problema parecido que ya haya sido resuelto.
- Determinar la técnica empleada y la solución obtenida.
- Mientras sea posible, utilizar la técnica y la solución descrita para resolver el problema.

Existen varios métodos heurísticos que solucionan problemas de diversa naturaleza, Sanz Montemayor propone la siguiente clasificación:

- **Métodos constructivos:** son capaces de construir una solución aproximada a un problema por medio de una estrategia, dentro de las que se destacan la estrategia de búsqueda voraz. En la cual se construye paso a paso una solución factible, que puede cambiar en cada iteración ya que

este tipo de algoritmos eligen la mejor solución actual sin importarles lo que ocurrirá en un futuro.

- **Métodos de búsqueda:** parten de una solución factible dada e intentan mejorarla.

  La principal desventaja de los algoritmos heurísticos es que no poseen ningún mecanismo que les permita trascender de los óptimos locales, con lo cual no pueden garantizar una exploración profunda que termine en el hallazgo de una solución factible.

5.1.2. Meta-heurística

Según Liberman, “una Metaheurística es un tipo general de método de solución que organiza la interacción entre los procedimientos de mejora local y las estrategias de más alto nivel para crear un proceso que sea capaz de escapar de un óptimo local y realizar una búsqueda vigorosa en una región factible” (Libersman, 2001). La definición anteriormente citada nos indica que una característica clave de le metaheurística es la capacidad de escapar de un óptimo local, por lo tanto después de encontrar una solución factible, toda metaheurística debe tener un mecanismo que le permita explorar más allá de esta solución con el fin de comprobar si se encuentra en un óptimo local o global. Las metaheurísticas se dividen en dos grandes bloques:

- **Metaheurísticas de trayectoria:** parten de una solución inicial, y son capaces de trazar un camino (o trayectoria) en el espacio de búsqueda a través de opciones de movimiento. Las características de la trayectoria proporcionan información sobre el comportamiento del algoritmo y su efectividad.
- **Metaheurísticas poblacionales:** emplean un conjunto de soluciones (población) en cada iteración del algoritmo, en lugar de utilizar una única solución como lo hacen las metaheurísticas trayectoriales. Este tipo de metaheurística proporciona un mecanismo de exploración en paralelo en el espacio de soluciones, por lo cual su eficiencia depende en cómo se maneje dicha población. Dentro de las metaheurísticas poblacionales se encuentra la *búsqueda dispersa,* el *re-encadenamiento de trayectorias,* los *algoritmos evolutivos*, entre otros. (Duarte Muñoz, 2007)

La ventaja de una Metaheurística es que tiende a moverse hacia soluciones muy buenas, por lo que proporciona una forma muy eficiente de abordar problemas grandes y complicados. En contraste, al usar una metaheurística no existe garantía de que la mejor solución encontrada sea una solución óptima, por lo cual siempre que se pueda usar un algoritmo que pueda garantizar el hallazgo de una solución óptima, debe usarse en vez de una metaheurística (Libersman, 2001).

5.1.3. Neo-Darwinismo

El paradigma Neo-Darwiniano se conoce como la inclusión del seleccionismo de August Weismann y la genética de Gregor Mendel en la teoría evolutiva de Charles Darwin. Este paradigma establece que la historia de la vida en el planeta puede ser explicada a través de una serie de procesos estadísticos que actúan sobre y dentro de las poblaciones y especies: la reproducción, la mutación la competencia y la selección (A. Hoffman, 1989).

La reproducción es una propiedad de todas las formas de vida de nuestro planeta, ya que de no contar con dicho mecanismo, la vida misma no tendría forma de replicarse. En cualquier sistema que se reproduce a sí mismo continuamente y que está rodeado de un medio en constante cambio, la mutación está garantizada (D. B. Fogel, 1995). El proceso de selección de los individuos es necesario, puesto que en el planeta los recursos para la vida son limitados, por lo cual es preferible que solo se prosperen los mejores individuos de una población.

5.1.4. Algoritmos Evolutivos

Los algoritmos evolutivos son metaheurísticas poblacionales, basados en la idea neo-darwiniana de la evolución de las especies (Sanz Montemayor) ; Es decir que los individuos que se adapten mejor al medio tienen mayor probabilidad de

sobrevivir, con lo cual obtienen una mayor posibilidad de generar descendencia que hereden sus buenas características. En contraste, los individuos con peor adaptación al medio poseen menos probabilidad de sobrevivir, por ende tendrán menores posibilidades de reproducirse y generar descendientes, con lo cual su código genético terminará extinguiéndose.

Santana Quintero & Coello (2006) en su investigación afirman que:

> Los algoritmos evolutivos abarcan una serie de técnicas inspiradas en los principios de la teoría Neo-Darwiniana de la evolución natural. Para simular el proceso evolutivo en una computadora se requiere:
>
> - Codificar las estructuras que se replicará.
> - Operaciones que afecten a los individuos (normalmente se usa cruce y mutación).
> - Una función de aptitud que nos indique que tan buena es la solución con respecto a las de más soluciones del conjunto.
> - Mecanismo de selección que implemente el principio de “supervivencia del más apto” de la teoría de Darwin.
>
> Aunque hoy en día es más difícil distinguir las diferencias entre los distintos tipos de algoritmos evolutivos existentes, por razones históricas, suele hablarse de tres paradigmas principales:
>
> - Programación Evolutiva: en este tipo de programación, la inteligencia se ve como un comportamiento adaptativo. La programación evolutiva enfatiza los

nexos de comportamiento entre padres e hijos, en vez de buscar emular operadores genéticos específicos (L. J. Fogel, A. J. Owens, & M. J. Walsh, 1966).

- Estrategias Evolutivas: Las estrategias evolutivas fueron desarrolladas en 1964 en Alemania para resolver problemas hidrodinámicos de alto grado de complejidad por un grupo de estudiantes de ingeniería encabezado por Ingo Richemberg (T. Back, 1996). La versión original de este algoritmo tenía un solo padre que generaba solo a un hijo, el hijo se mantenía si era mejor que el padre, o en caso contrario era eliminado (selección extintiva).
- Algoritmos Genéticos: enfatizan su importancia en el cruce sexual, la mutación (operador secundario) y usa selección probabilística. Los algoritmos genéticos serán descritos con mayor detalle más adelante.

5.1.5. Algoritmos Genéticos

5.1.5.1. Generalidades

Los algoritmos genéticos son la técnica más común de la *programación evolutiva*, son ampliamente aplicables en búsquedas estocásticas y técnicas de optimización (Gen & Cheng, 2000) . Por lo general, un algoritmo genético se compone de cinco elementos básicos, sumarizados por Michalewicz (Michalewicz, 1996):

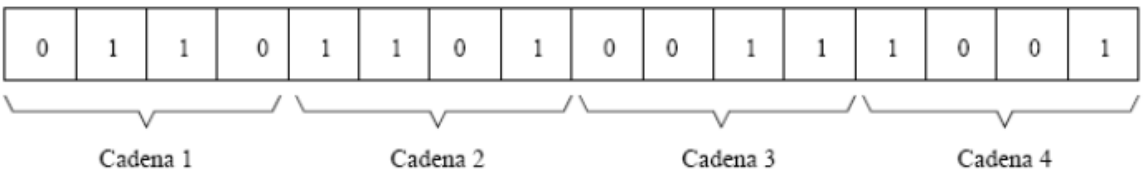
- Representación genética de la solución del problema.
- Creación de la población inicial de soluciones.

- Función de evaluación de las soluciones.
- Operadores genéticos que alteren la composición de los hijos durante la reproducción.
- Valores de los parámetros para los algoritmos genéticos.

5.1.5.2. Representación Genética.

La representación genética es la codificación de una solución desde el punto de vista computacional, en el trabajo de Holland dicha codificación se llevó a cabo por medio de una representación binaria, tal como lo muestra la figura 1. A la cadena binaria se le denomina “cromosoma”; Al bloque de bits que codifica una sola variable del problema se le denomina “gen” y al valor dentro de cada posición cromosómica se le denomina “alelo”.

FIGURA 1: Ejemplo de la codificación mediante cadenas binarias, usada tradicionalmente en los algoritmos genéticos.



FUENTE: (Santana Quintero & Coello Coello, 2006)

Los autores Mitsuo Gen y Runwei Cheng (Gen & Cheng, 2000) afirman que:

> La codificación binaria tradicional es insuficiente para abordar problemas complejos de la ingeniería industrial, por tal motivo se han realizado trabajos que usan otro tipo de codificaciones, ellos han clasificado estas representaciones en:
>
> - Representación de números reales.
> - Representación por medio de permutación lineal.
> - Representación por medio de estructuras de datos.
>
> Independiente del método de codificación seleccionado, usualmente es necesario establecer que el método de codificación sea efectivo, por lo cual se han establecido los siguientes principios de evaluación:
>
> - **No redundancia:** una representación solo debe pertenecer a una solución en el modelo y viceversa.
> - **Legalidad:** cualquier permutación de una codificación, corresponde a una solución.
> - **Completitud:** cualquier solución tiene una codificación correspondiente.
> - **Propiedad Lamarckiana:** el significado de los alelos para un determinado gen no debe ser dependiente del contexto.

- **Causalidad:** pequeñas variaciones en el espacio del genotipo por medio de la mutación, debe implicar pequeñas variaciones en el espacio del fenotipo.

Las propiedades anteriormente citadas, contribuyen a la *legalidad* y *factibilidad* de la solución, teniendo en cuenta que una solución es ilegal cuando el cromosoma no representa la solución a un problema dado; por otra parte, la no factibilidad se refiere al fenómeno en el cual la solución codificada está por fuera de la región factible del problema.

5.1.5.3. Operadores de selección.

En cada generación, los individuos se reproducen, sobreviven o desaparecen de la población vado la acción de dos operadores de selección: la *selección para la reproducción,* la cual determina cuantas veces un individuo se reproducirá en una generación; por otra parte la *selección para el reemplazo*, determina cuales individuos deben desaparecer de la población con el fin de que esta siempre permanezca constante y con los mejores individuos.

5.1.5.4. Operadores de variación.

Los operadores de variación permiten explorar soluciones en otras zonas de la región factible, con el fin de trascender de los óptimos locales y poder hallar un óptimo global. Estos son:

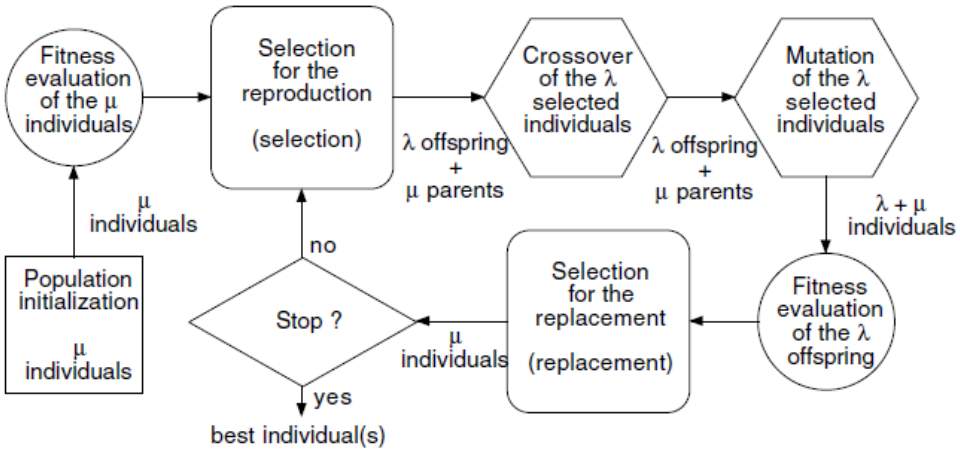
- Mutación: es el proceso por el cual un individuo es modificado con el fin de formar otro. Asemeja un cambio en la estructura genética del cromosoma teniendo en cuenta un cambio en el entorno.
- Cruce: genera una o más descendencia por medio de la combinación del código genético de los padres.

5.1.5.5. Ciclo generacional

En cada generación, la población de un algoritmo genético es afectada por los operadores anteriormente descritos de la siguiente forma, (ver gráfico 2):

1. Para la reproducción, son seleccionados los padres por medio de una población inicial que generará determinada descendencia.
2. Cruce y mutación de la población inicial para generar su descendencia.
3. Evaluación de la descendencia.
4. Construir la población que seguirá para la siguiente generación, por medio de una selección basada en los resultados obtenidos al evaluar la solución con la función de aptitud, en este paso son seleccionados los mejores individuos y descartados los demás.

FIGURA 2: muestra gráficamente el ciclo generacional.



FUENTE: (Davis, 1991)

5.1.6. Programación orientada a objetos.

La programación orientada a objetos es un paradigma de desarrollo que permite una representación más directa del modelo del mundo en la vida real. “Las técnicas orientadas a objetos proporcionan mejoras y metodologías para construir sistemas de software complejos a partir de componentes reutilizables” (Joyanes Aguilar, 1999).

Las cuatro propiedades más importantes de la programación orientada a objetos son: ¿Y la herencia no es importante?

- Abstracción: propiedad que permite representar las características más importantes y esenciales de un objeto sin preocuparse por las secundarias o aquellas que están fuera del contexto.
- Polimorfismo: es la posibilidad de referirse a diversos tipos de objetos por medio de un mismo elemento, y de igual forma que una misma operación pueda tener diversas implementaciones.
- Encapsulación: es la propiedad que permite establecer niveles de acceso a los objetos o sus propiedades desde el exterior.
- Modularidad: propiedad que posibilita la subdivisión de una aplicación en partes más pequeñas, independientes y totalmente funcionales.
- Herencia: permite establecer un orden en una abstracción. Las más importantes son: estructura de clases (<<es-un>>: generalización/especialización) y estructura de objetos (<<parte de>>: agregación).

5.1.7. Ingeniería de software

"La ingeniería de software es una disciplina de ingeniería que comprende todos los aspectos de la producción de software desde etapas iniciales de la especificación del sistema, hasta el mantenimiento de este después que se utiliza" (Sommervile, 2005).

En la ingeniería de software, el proceso del software es el conjunto de actividades y resultados asociados que producen un producto de software, estas actividades son:

- Especificación: se toman los requerimientos del cliente, estos pueden ser funcionales o no funcionales. Un requerimiento funcional es aquella característica necesaria para que el sistema solucione su problema de dominio, por otra parte, un requerimiento no funcional es una función adicional que enriquece el producto más no incide en su funcionalidad.

- Desarrollo del software: en esta etapa encontramos el periodo de diseño y programación. En el periodo de diseño se hace una abstracción de la solución planteada y esta es expresada por medio de un lenguaje de modelado, actualmente el más usado es el lenguaje unificado de modelado o UML.

  En el periodo de programación, se elabora el producto teniendo en cuenta las especificaciones de diseño y algunas modificaciones necesarias para llevarlo a la práctica. Se usa un lenguaje de programación, metodologías de desarrollo y tecnologías de apoyo para la construcción de un buen producto.

- Evolución del software: esta etapa es en la que el software es modificado para adaptarse al entorno real de uso.

Los atributos esenciales que debe tener una buena pieza de software según Sommerville son:

- Mantenibilidad: El software debe ser codificado de tal forma que pueda evolucionar para cumplir con las necesidades de cambio de los clientes.
- Confiabilidad: Un software confiable no debe causar daños físicos ni económicos en caso de falla.
- Eficiencia: saber administrar los recursos del sistema.
- Usabilidad: el sistema debe ser fácil de usar, no debe producir un gran esfuerzo adiciona para un usuario.

5.1.8. Lenguaje Unificado de Modelado (UML)

“El lenguaje unificado de modelado (UML) es un lenguaje gráfico para la visualización, especificación y construcción de artefactos de un sistema intensivo de software. UML Ofrece una vía estándar para escribir planos del sistema, incluyendo aspectos conceptuales como los procesos de negocio y funcionalidades del sistema, así como aspectos concretos tales como

declaraciones del lenguaje de programación, esquemas de bases de datos y componentes reutilizables de software” (Object Management Group).

Los diagramas UML, están divididos en dos grupos:

- Diagramas de modelado estructural: son aquellos que definen la estructura estática del modelo, ellos son usados para representar las relaciones y dependencias entre los elementos (Sparx Systems). Los diagramas estructurales más usados son:
    - Diagramas de paquetes: dividen el modelo en contenedores lógicos o “paquetes”, y describen la relación entre estos en un alto nivel.
    - Diagrama de clases: define la construcción de los bloques básicos del modelo: los tipos, clases y otros materiales usados para construir el modelo completo.
    - Diagrama de objetos: modela como los elementos estructurales interactúen entre sí en tiempo de ejecución.
    - Diagrama de Componentes: son usados para el acople con estructuras más complejas desde un alto nivel.
    - Diagrama de despliegue: define una disposición de los artefactos físicos necesarios para la ejecución del sistema.

- Diagramas de comportamiento: capturan las interacciones y los estados de los objetos en un modelo al estar el sistema en ejecución (Sparx Systems). Los diagramas de comportamiento más usados son:
    - Diagrama de casos de uso: Modela las interacciones entre los usuarios y el sistema.
    - Diagrama de actividades: puede ser usado para definir un flujo básico en el programa, o para capturar los puntos de decisión y acciones en cualquier proceso generalizado.
    - Diagrama de secuencia: muestran la secuencia de mensajes o comunicaciones entre objetos en tiempo de ejecución.

5.2. ANTECEDENTES

Desde los años 1930, la evolución natural fue vista como un proceso de aprendizaje (Santana Quintero & Coello Coello, 2006). Alan Mathinson Turing, en su libro "*Computing Machinery and Inteligence* " (Considerado hoy un clásico en Inteligencia artificial) acepta la conexión "obvia" que existe entre la evolución y el aprendizaje de máquina (Turing, 1950). A finales de 1950 y principios de 1960, Alexander S. Fraser iniciaría el camino de lo que hoy llamamos algoritmo genético. En el trabajo de Fraser se incluye: una representación binaria del la solución, un operador de cruce probabilístico, población de padres que generaban una nueva población de hijos tras recombinarse y el empleo de un mecanismo de selección.

George J. Friedman ha sido el precursor de la aplicación de técnicas computacionales evolutivas en la robótica, puesto que en su tesis de maestría propuso evolucionar una serie de circuitos de control, similares a las redes neuronales actuales, usando un procedimiento denominado "*retroalimentación selectiva*", proceso análogo a la selección natural actual usada en los algoritmos evolutivos (Friedman, 1956).

Hans Joachim Bremermann fue pionero en concebir la evolución como un proceso de optimización (Bremermann H. J., The evolution of intelligence. The nervous system as a model of its enviroment, 1958), además realizó una de las primeras simulaciones de la evolución empleando cadenas binarias procesadas por medio de técnicas reproductivas sexuales o asexuales, procedimientos de selección y mutación, lo cual convierte su trabajo en un predecesor de los algoritmos genéticos. Bremermann en un trabajo posterior utilizó una técnica evolutiva para solucionar problemas de optimización con restricciones lineales; la idea principal de esta propuesta era usar un individuo factible, el cuál era alterado por medio de un operador de mutación hacia un conjunto de direcciones posibles de movimiento (Bremermann H. J., Optimization through evolution and recombination, 1962). Bremermann y Rogson fueron unos de los primeros en utilizar el concepto de "población" en la simulación de procesos evolutivos (Bremermann & Rogson, 1964).

En 1975 fue publicado el libro "*Adaptation in Natural and Artificial Systems*" de John Holland, en el cuál se exponen los fundamentos básicos para la construcción de algoritmos genéticos que han sido usados hasta la actualidad. En el mismo año un estudiante de doctorado que apoyó a Holland en su investigación, Ken De Jong , demostró los beneficios de usar los algoritmos genéticos en problemas de optimización (Jong, 1975).

En la década posterior a la publicación de Holland, muchos estudios fueron realizados implementando algoritmos genéticos en la solución de problemas de optimización, entre estos se destaca la tesis de doctorado de David Goldberg, un estudiante de Holland, quien aplicó la teoría de algoritmos genéticos en la optimización de la implantación de tuberías de gas en 1989. Goldberg también escribió uno de los libros más influyentes en el uso de los algoritmos genéticos, llamado "*Genetic Algorithms in search, Optimization and Machine Learning*". Este fue el trabajo que finalmente impulsó la evolución de la teoría de algoritmos genéticos y el uso de los mismos en la solución de problemas de optimización.

Actualmente los algoritmos genéticos son una técnica de inteligencia artificial ampliamente usada en problemas de optimización y aprendizaje de máquina. Se aplican a problemas combinatorios complejos en los cuales hay un gasto computacional elevado, debido a procesos combinatorios. En el campo de la

optimización son ampliamente usados en los procesos de asignación de recursos, denominados “problemas de schedulling”. Un problema de asignación es un tipo de problema *NP-Hard*, lo cual nos indica que hace parte de los problemas más complejos (computacionalmente hablando), puesto que posee un número elevado de operaciones combinatorias (Libersman, 2001).

En la investigación de operaciones, el “Universe Course Timetable Problem” o UCTP (Problema de la tabla de cursos universitarios), es un problema clásico de optimización, abordado por la investigación de operaciones en el cuál es necesario hacer la mejor asignación posible del recurso físico de un establecimiento universitario con el fín de cumplir con el plan de cursos ofertados, teniendo en cuenta restricciones en el momento de la asignación y maximización en el cumplimiento de las preferencias. Teniendo en cuenta que éste es un problema cásico de optimización, hay diversos estudios que lo abordan desde múltiples perspectivas, entre estos se destacan los siguientes:

- En la universidad de Xavier en Kolkata (India) se ideó la construcción de un algoritmo genético con representación estructurada[1] y evaluado con lógica difusa[2] que busca asignar eficientemente las aulas, laboratorios, horarios y docentes. En este proyecto utilizaron una representación indirecta del cromosoma, lo cual implica que se pueden realizar sub-cálculos en ciertos

procedimientos internos que contribuyen en reducir el gasto computacional en el algoritmo principal. Por otra parte, los investigadores han introducido el concepto de restricciones fuertes y blandas: una restricción fuerte "*hard constraint*" engloba todas aquellas situaciones que hacen infactible a una solución, un ejemplo de esto es el sobrecupo en las aulas de clase o la asignación de un laboratorio al que no corresponde la asignatura; En contraste una restricción blanda "*soft constraint*" , puede ser incumplida por una solución sin necesidad de que esta sea infactible. Usando lógica difusa se puede establecer en qué grado una solución cumple con una restricción blanda en particular.

Teniendo en cuenta que en el algoritmo se destruye aquel cromosoma que incumple una o varias restricciones fuertes, la mejor solución es aquella que permite el cumplimiento (en la mayor proporción posible) de las restricciones blandas (Chaudhuri & Kajal, 2010).

- En la investigación de la universidad politécnica de Palestina, se construye un algoritmo evolucionario para encontrar la mejor solución al problema de asignación de cursos. Los autores han preferido un algoritmo evolucionario en vez de un algoritmo genético para permitir que mediante mutación se permita explorar muchas más soluciones y así evitar el problema de los mínimos locales. Su algoritmo (Al igual que la investigación anteriormente

expuesta) ,pose 2 tipos de restricciones y el índice de favorabilidad medido es superior al 70% en las preferencias y del 100% en las restricciones duras. De este trabajo el autor ha comprendido la importancia de la organización de los bloques horarios y que la selección de un espacio vacío es muy importante para el performance del algoritmo; por otra parte también se resalta la importancia del índice de mutación y las técnicas para medir la favorabilidad del producto mediante análisis estadístico (Aldasht, Adi, & Alsaheb, 2010).

- La investigación en la universidad de Zagreb (Croacia) presenta el UTCP (*University Timetable Course Problem*) como un problema NP-Hard que puede ser resuelto por medio de un algoritmo genético, teniendo en cuenta un modelo tridimensional que sitúa las asignaciones en un cubo. Este algoritmo, los cromosomas son tratados individualmente en el momento del cruce o mutación con el fin de evitar que el resultado de estas operaciones sea un individuo infactible. Además, se tienen como restricción fuerte los cruces de horarios y como preferencias las clases en las mañanas (Sigl, Golub, & Mornar, 2011).

En las investigaciones anteriormente expuestas es evidente una evolución en cuanto a la formulación del algoritmo genético, puesto que se han usado representaciones estructuradas del cromosoma, se han hecho pre-procesamientos

teniendo en cuenta que los modelos actualmente tratados son más complejos. Además, todas coinciden con la introducción de las restricciones y preferencias (“*hard constraints”* y “*soft constraints*”), además del uso de metaheurísticas auxiliares que favorezcan en hallar el optimo global con mayor facilidad.

6. DISEÑO DEL ALGORITMO GENÉTICO DE ASIGNACIÓN.

6.1. CONSIDERACIONES DE DISEÑO.

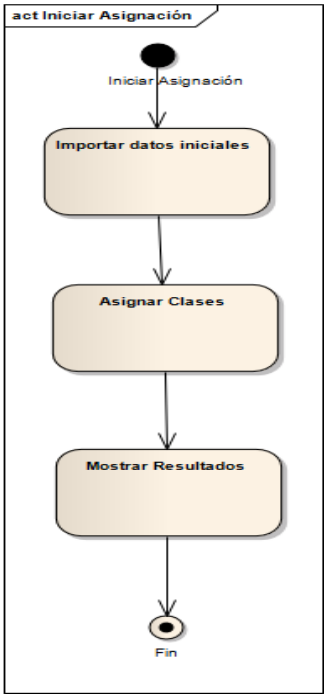
La optimización de la asignación de aulas y laboratorios para los cursos de pregrado de la Universidad Tecnológica de Bolívar, se ha definido en el presente proyecto como la minimización de los cupos desperdiciados cada hora en un aula mientras se dicta una clase. Una clase es la programación de un curso en espacio de tiempo determinado, la cual necesita de un aula para poder ser impartida. Además, contiene el número de estudiantes matriculados al curso y el tipo asociado; este último atributo determina qué tipo de aula o laboratorio es necesario para la clase teniendo en cuenta los recursos requeridos.

Un aula es un espacio físico en el cuál se dictan clases, por lo cual existen diversos tipos de aulas de clases que contienen los recursos necesarios para impartir cursos determinados, es así como tenemos: laboratorios, salas de informática, aulas de clases teóricas, salones multimedia…entre otros.

En esta investigación, la programación general clases se define como la asignación de un aula adecuada para cada clase de pregrado programada en la Universidad Tecnológica de Bolívar. Esta programación puede ser interpretada como el conjunto de programaciones individuales de cada tipo de clase por

separado, concepción que permite abstraer el problema a un modelo general que permite optimizar la asignación para cualquier conjunto de clases pertenecientes a un tipo en específico, mientras se indique el conjunto de aulas que satisfagan las necesidades de las clases seleccionadas (Ver figura 5).

Figura 3: Muestra el diagrama de secuencia, desde una perspectiva general de lo que debe hacer un algoritmo de asignación para todos los cursos de pregrado en la Universidad Tecnológica de Bolívar.



Fuente: propia.

Ecuación 1: Describe que la programación general de clases es igual al conjunto de programaciones de cada tipo. Donde P_G es la programación general de aulas y P_{tn} es la programación de aulas para un tipo de clase en particular y t_i representa el tipo de clase.

$$P_G = [P_{t1}, P_{t2}, \dots P_{tn}]$$

Fuente: Propia.

Definimos como programación de un tipo de clases en específico, como la asignación de aulas para cada clase del conjunto en una semana ordinaria. La asignación semanal de aulas para las clases de pregrado en la Universidad Tecnológica de Bolívar, es el conjunto de las programaciones individuales de los días: Lunes, Martes, Miércoles, Jueves, Viernes y Sábado. Dichas asignaciones no están relacionadas entre sí (entrevista personal SIRIUS, no referenciada). En la ecuación 2, se muestra como la programación semanal de aulas para las clases de pregrado en la Universidad Tecnológica de Bolívar puede ser dividida en el conjunto de programaciones diarias (Ver ecuación 2). Esta concepción permite la asignación de aulas para un conjunto de clases que estén programadas para un día y que pertenezcan a un mismo tipo (Ver Figura 4).

Ecuación 2: la programación semanal de un tipo de clase es el conjunto de programaciones diarias para dicho tipo. Donde P_{tn} es la programación se aulas semanal para un tipo de clase, P_{dn} la programación diaria de aulas para cada tipo de clase de pregrado; t representa el tipo de clase y d_i el día de la semana en el que se realiza la asignación.

$$P_{tn} = [P_{tnd1}, P_{tnd2}, \dots P_{tnd6}]$$

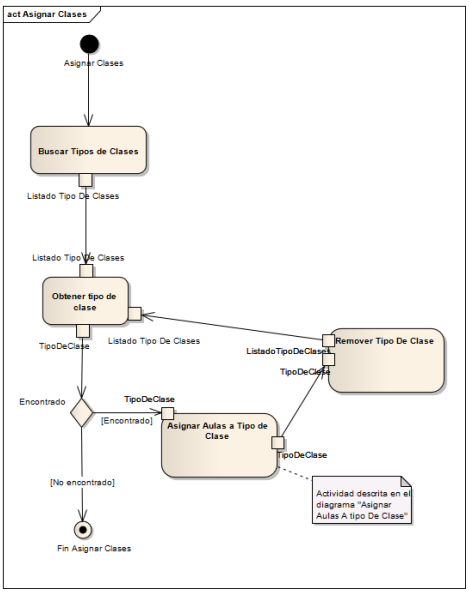
Fuente: Propia.

Ecuación 3: fusión de las ecuaciones 1 y 2.

$$P_G = [\{P_{t1d1}, P_{t1d2}, \dots P_{t1d6}\}, \{P_{t2d1}, P_{t2d2}, \dots P_{t2d6}\} \dots \{P_{tnd1}, P_{tnd2}, \dots P_{tnd6}\}]$$

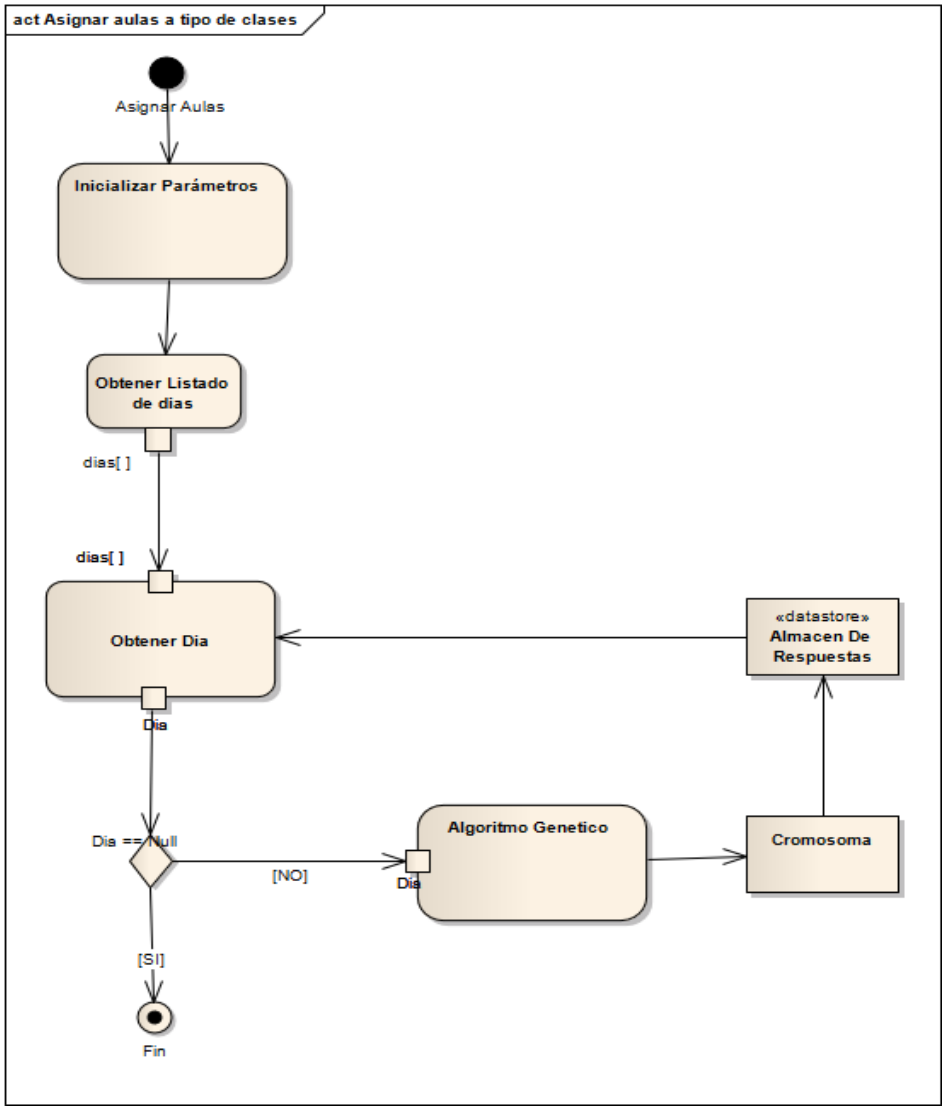
Fuente: propia.

Figura 4: Diagrama se secuencia que modela la reutilización de la asignación de aulas para cada tipo de clase.



Fuente: propia.

Figura 5: Muestra el diagrama de secuencia de la reutilización de un algoritmo genético de asignación de aulas para cada día de la semana académica.



Fuente: propia.

6.2. REPRESENTACIÓN DE LA SOLUCIÓN

Como se explicó anteriormente, un algoritmo genético es una técnica de inteligencia artificial que emplea metaheurísticas evolutivas para alcanzar el valor óptimo de una minimización o maximización, o en su defecto el mejor valor posible en un tiempo determinado (Libersman, 2001). El primer paso para la utilización de esta técnica es la representación de la solución, denominada “Cromosoma”, la cuál está firmemente ligada a la abstracción del modelo.

El algoritmo genético que ha sido diseñado para la asignación de las aulas de todos los cursos de pregrado, será reutilizado cada día de la semana y por cada tipo de clase. Esto permite concluir que este será ejecutado 6T veces, donde T representa el número de tipos de clases y la constante 6 los días de semana habilitados para impartir las clases de pregrado. El algoritmo tiene las siguientes entradas:

- Tipo de clase.
- Tipo de aula.
- Día de la semana.

Y tendrá como salida una codificación de la respuesta denominada Cromosoma, que será la mejor asignación posible en base a los parámetros de entrada. Dicha codificación está descrita por la formula 4 y su explicación gráfica, está descrita por la gráfica 6.

Ecuación 4: Representación matemática de un cromosoma.

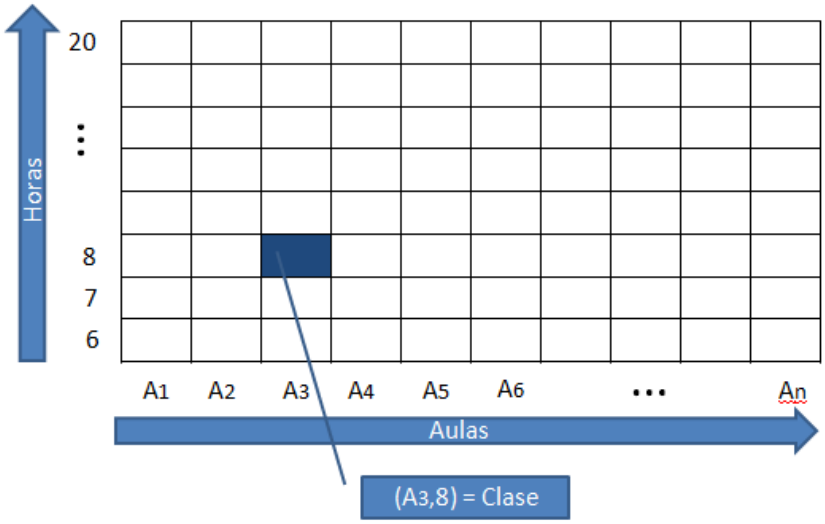
$$P_{tndn} = [G_1, G_2 \dots G_n] \text{ Donde } G_n = (\, idA, h)$$

P_{tndn} Representa la programación de las aulas en un día académico de un tipo de clase determinado, también denominado *Cromosoma*.

G_n: es un punto en un espacio bidimensional, que contiene la clase que ha sido asignada en el aula con identificación "*idA*" a la hora "*h*", también denominado *Gen* El número de genes del cromosoma está determinado por *15A*, donde la constante 15 representa las horas disponibles para asignar clases en un aula [6am - 8pm] y *A* representa el número de aulas disponibles. Un Gen puede contener información nula, lo cuál indica que no hay clase asignada.

Fuente: propia.

Figura 6: Representación gráfica de un cromosoma.



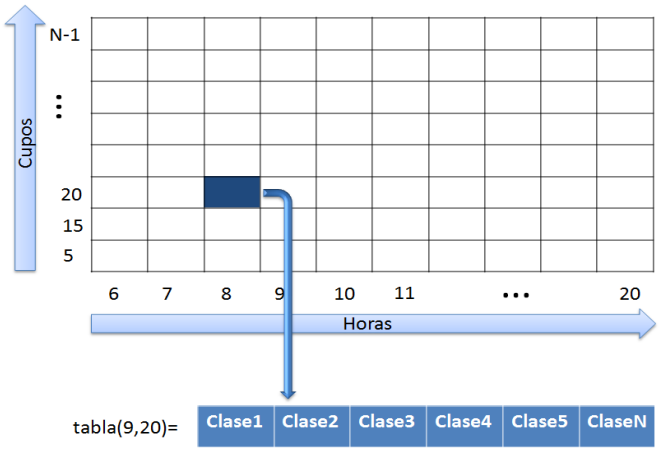
Fuente: Propia.

6.3. HEURÍSTICA GENERACIÓN DE POBLACIÓN INICIAL.

Para la generación de la población inicial, el algoritmo usa una estructura complementaria de apoyo que clasifica las clases en listas ubicadas de acuerdo al número de estudiantes matriculados y el cupo mínimo de aceptación. El *cupo mínimo de aceptación* se halla buscando el valor más pequeño de la *lista de capacidades* que sea mayor o igual al número de estudiantes que posee la clase. La lista de capacidades es el conjunto de cupos máximos en todas las aulas disponibles para el tipo de clase seleccionado.

Una vez se ha hecho la clasificación, el algoritmo podrá solicitar una clase aleatoria de la lista, que a su vez está ubicada en la coordenada (h,cm). Donde *h* es la hora de inicio de la clase y *cm* el cupo mínimo de aceptación (Ver gráfica 7).

Figura 7: representación gráfica de la estructura complementaria.

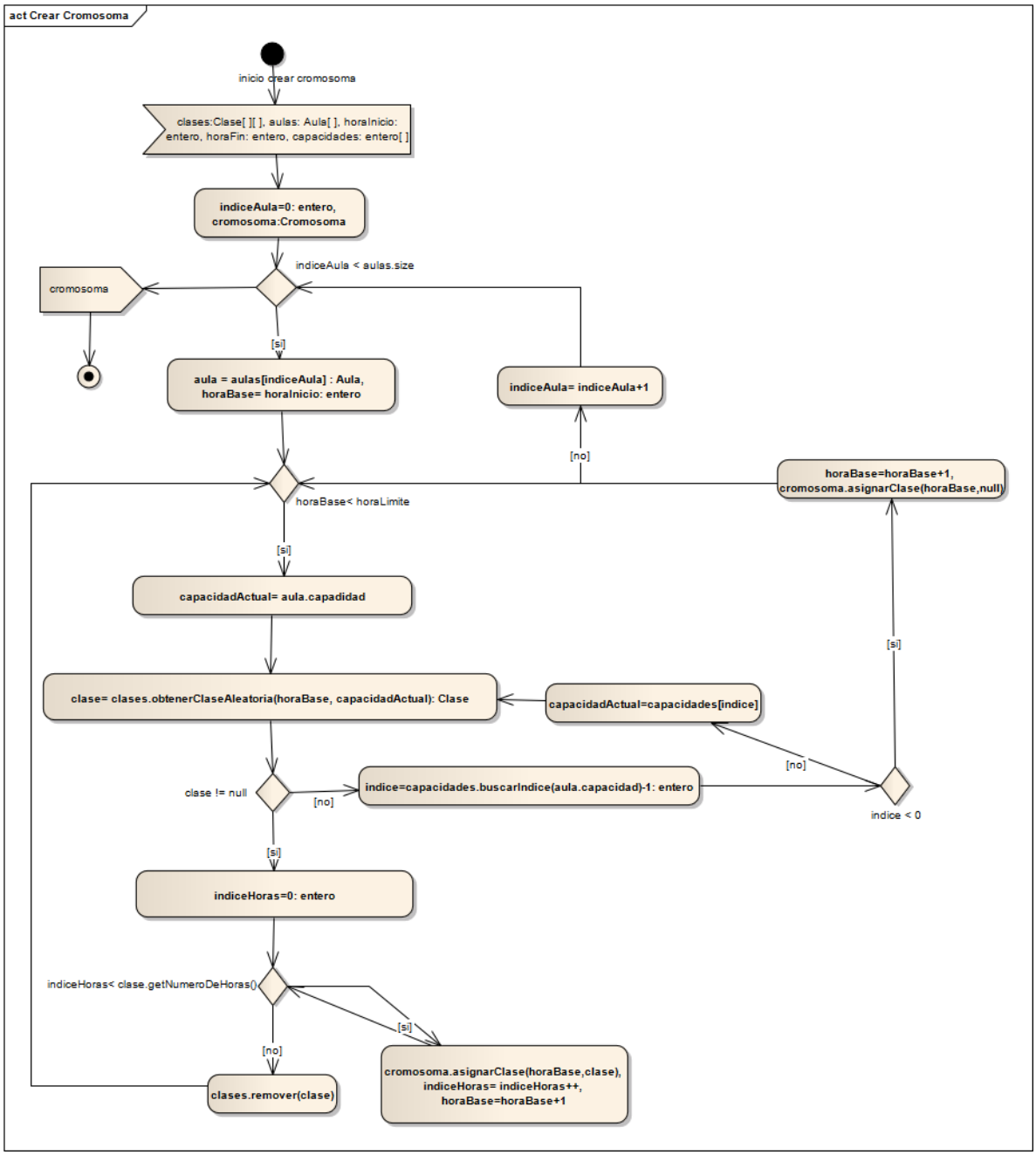


Fuente: propia.

Para la creación de la población inicial o generación inicial, el algoritmo crea el número de cromosomas que ha sido recibido como parámetro. Cada cromosoma es creado por medio de una heurística de asignación de clases que utiliza una copia de la estructura complementaria a la cual se le podrán hacer modificaciones y así poder reutilizar la estructura original en las siguientes creaciones.

La heurística de asignación de clases divide la disponibilidad de cada aula en bloques que representan las horas de clases. Una clase puede ocupar varios bloques seguidos en un aula, dependiendo el número de horas que posea; Al asignar una clase, el primer bloque ocupado corresponde a la hora de inicio de la misma. El objetivo principal de la rutina que programa las clases es asignar a cada bloque horario una clase aleatoria del conjunto de clases en el cual el desperdicio de cupos será mínimo. La copia de la estructura complementaria de clases se usa para facilitar la búsqueda de una clase factible por medio de la hora de inicio y el cupo máximo del aula (Ver figura 8). Este heurística es la fusión de las técnicas usadas en las publicaciones científicas de (Aldasht, Adi, & Alsaheb, 2010) y (Chaudhuri & Kajal, 2010), con ciertas modificaciones y reducciones, teniendo en cuenta la particularidad del modelo.

Figura 8: diagrama de flujo que explica gráficamente la heurística de crear un cromosoma.



Fuente: propia.

6.4. FUNCIÓN DE APTITUD.

La función que evalúa la aptitud de un cromosoma en al algoritmo genético está basada en el objetivo del mismo, el cuál es minimizar los cupos desperdiciados en la asignación de aulas. Este es un problema típico de “scheduling”, el cual puede ser minimizado por medio de la programación lineal entera mixta, la cuál usa teorías combinatorias para hallar la solución óptima (Peña & Zulmenzu, 2010). Siguiendo ese enfoque, el algoritmo considera cupo desperdiciado a la diferencia entre el cupo disponible en el aula menos el número de estudiantes inscritos a la clase en una hora. Ver ecuación 5.

Ecuación 5: función objetivo para minimizar el número de cupos desperdiciados en la asignación diaria de un tipo de clases.

$$ZD_{min} = \sum_{i=1}^{n} cd(G_i)$$

Donde

n= número de genes del cromosoma.

G_i= gen del cromosoma i-ésimo. Equivale a una asignación en un aula de una clase en un bloque de una hora.

cd(G_i)= función que recibe un gen del cromosoma y devuelve el número de cupos desperdiciados. El número de cupos desperdiciados es el resultado de la diferencia el cupo total del aula menos el número de estudiantes matriculados al curso.

Fuente: propia

La función descrita anteriormente corresponde a la minimización de un cromosoma que representa la asignación de aulas para un tipo de clase, en un día. La función objetivo que minimiza el número de cupos desperdiciados para la asignación semanal de un tipo de clases se encuentra en la ecuación 6.

Ecuación 6: función objetivo para minimizar el número de cupos desperdiciados en la asignación semanal de un tipo de clases.

$$ZS_{min} = \sum_{d=1}^{nd} \sum_{i=1}^{n} cd(G_{id})$$

Donde:

n= número de genes del cromosoma.

nd= número de días.

G_{id}= gen en la posición i del cromosoma que representa la solución del día d. Equivale a una asignación en un aula de una clase en un bloque de una hora en un día específico.

$cd(G_{id})$= función que recibe un gen del cromosoma y devuelve el número de cupos desperdiciados.

Fuente: propia.

La función que describe la minimización del desperdicio de cupos en la asignación de todos los tipos de clases está definida en la ecuación 7.

Ecuación 7: función objetivo para minimizar el número de cupos desperdiciados en la asignación semanal de todas las clases de pregrado.

$$ZG_{min} = \sum_{t=1}^{nt} \sum_{d=1}^{nd} \sum_{i=1}^{n} cd(G_{idt})$$

Donde:

n=número de genes totales

nd=número de días

i=gen.

d=día.

t= tipo de clase.

nt= número de tipos de clases.

G_{idt}= gen en la posición i del cromosoma que representa la solución del día d para el tipo de clase t.

$cd(G_{idt})$= función que recibe un gen del cromosoma y devuelve el número de cupos desperdiciados.

Fuente: propia.

Teniendo en cuenta que la mayoría de las técnicas genéticas están diseñadas específicamente para una función objetivo de maximización, el algoritmo genético reutilizable trabajará con la conversión de la función de minimización a maximización. (Libersman, 2001)

Ecuación 8: conversión de las funciones objetivos de minimización, a maximización.

$$ZD_{max} = \frac{1}{ZD_{min}}, ZS_{max} = \frac{1}{ZS_{min}}, ZG_{max} = \frac{1}{ZG_{min}}$$

Fuente: propia.

6.5. OPERADORES GENÉTICOS.

6.5.1. Operador de selección.

El operador de selección que ha sido usado en el algoritmo es la *selección universal aleatoria* (SUS). SUS es parecido a una ruleta con M punteros igualmente espaciados entre sí, de modo que con un solo lanzamiento se obtienen M ganadores. Este método no posee sesgo y su dispersión es la mínima posible (Davis, 1991). El algoritmo es:

```
Sum=0
Ptr= rand()  E [0,1]
Para (i=1) hasta M
       Sum = Sum+ Ne[i]
       Mientras(sum> ptr) hacer
                 seleccionarIndividuo[i]
                 ptr=ptr+1
       FIN MIENTRAS
  FIN PARA
```

Donde $N_e[i]$ es el número de copias para el individuo i-ésimo, definido como el producto de la aptitud del cromosoma entre la aptitud media de toda la población. (Estévez Valencia, 2011)

6.5.2. Operador de cruce.

El operador de cruce que se ha seleccionado para el algoritmo genético es el *cruce uniforme*, en el cuál cada gen de los padres tiene la posibilidad de pertenecer a los hijos, puesto que la implementación genera un número aleatorio y dependiendo un umbral se determina de qué padre se obtendrá el siguiente gen. Este operador es ampliamente usado puesto que contribuye a la preservación e identificación del conjunto de genes comunes, además que permite recombinar los genes no comunes (Sywerda, 1989). Una modificación que se le hizo a este procedimiento es darle un rango del 60% del umbral al cromosoma padre cuyo valor de aptitud sea mayor. Con esto se garantiza un mayor número de genes del mejor de los padres en el nuevo individuo (Falkenauer, 1999).

En el algoritmo genético planteado, el conjunto de genes pertenecientes al aula del padre seleccionado se copia a su posición equivalente en el nuevo cromosoma. Paralelamente al proceso de copia, se lleva un registro de las clases que han sido asignadas y en el nuevo cromosoma con el fin de no repetir asignaciones. Aquellas clases que no hayan podido ser asignadas, se reasignarán empleando una heurística similar a la aplicada en la generación de la población inicial. Al final del procedimiento se contrasta el número de clases totales a asignar y aquellas que han sido colocadas en la estructura. En caso de que la diferencia

no sea cero, el cromosoma hijo será descartado puesto que representaría una solución infactible.

6.5.3. Operador de mutación.

El operador de mutación que se ha planteado es la *mutación por doble intercambio (2-opt),* el cual es uno de los mejores operadores de mutación ampliamente usados debido a su eficacia (Gen & Cheng, 2000). Consiste en intercambiar la posición de dos genes seleccionados de manera aleatoria y teniendo en cuenta que no sean el mismo. En el modelo planteado se intercambian todas las clases programadas en 2 aulas seleccionadas aleatoriamente, bajo la restricción de que cualquier sobrecupo en un aula hará infactible el nuevo cromosoma.

6.5.4. Operador de reemplazo.

El operador de reemplazo aplicado en el algoritmo genético es de tipo *generacional*. Este tipo de reemplazo une el conjunto de cromosomas pertenecientes a la generación actual con el conjunto de cromosomas hijos, los ordena de forma descendente de acuerdo al valor de sus aptitudes y elimina los cromosomas en orden inverso hasta quedar con un numero de cromosomas igual al de la población inicial. En este procedimiento se ha creado una nueva generación, que ha remplazado a la anterior (Aldasht, Adi, & Alsaheb, 2010).

7. DISEÑO DEL GESTOR DE ASIGNACIÓN INTELIGENTE DE AULAS (GAIA).

7.1. REQUERIMIENTOS DEL SISTEMA.

7.1.1. Requerimientos Funcionales

1. Asignar Aulas.
2. Importar datos iniciales
3. Gestionar Algoritmo Genético
    a. Cambiar parámetros de algoritmo genético.
    b. Iniciar algoritmo genético
4. Mostrar Asignación.
5. Mostrar estadísticas.
6. Importar / Exportar asignación.
7. Comparar resultados.
8. Gestionar Usuarios.
9. Autenticar Usuario.

Los requerimientos funcionales del sistema serán descritos detalladamente en el anexo I del documento.

7.1.2. Requerimientos no funcionales

- RNF1: Exportar resultados con formatos compatibles con Microsoft Excel 2007 o superior.
- RNF2: El sistema debe permitir la implantación futura de asignación basada en preferencias.

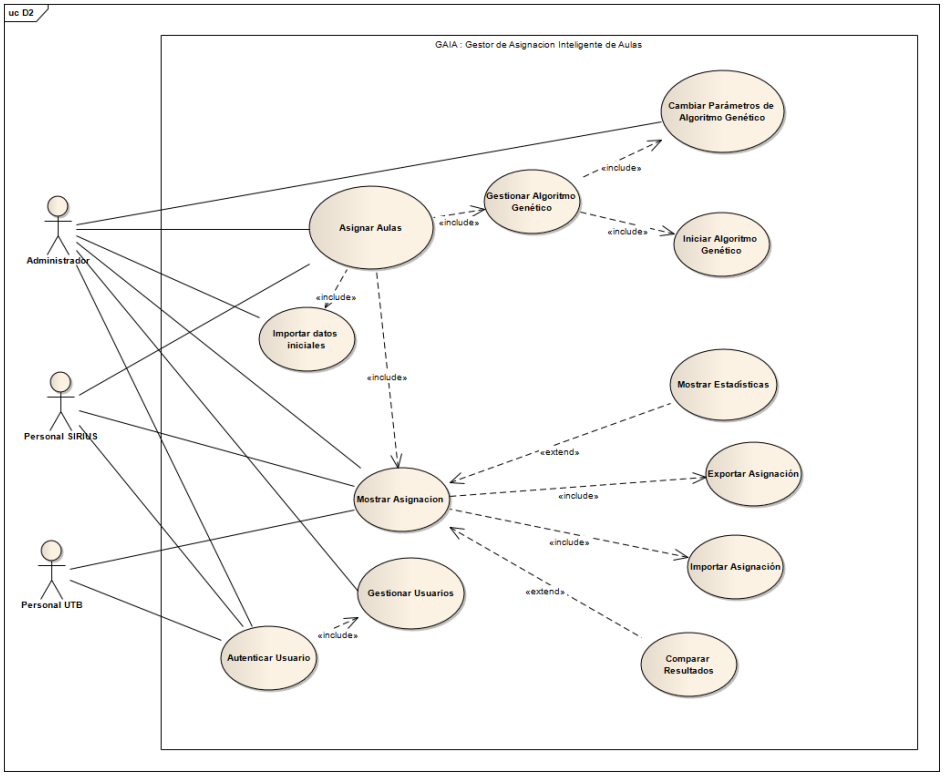
- RNF3: El sistema deberá soportar multi-threading con el fin de utilizar mejor los recursos de la máquina.

7.2. CASOS DE USO DEL SISTEMA

7.2.1. Diagrama de Casos de Uso.

En la figura 9 del presente documento, se ilustra el diagrama de casos de uso del sistema. Esté modelo ilustra las interacciones entre el usuario y el sistema, por lo cual indica las funcionalidades que deben ser implementadas y sus dependencias.

Figura 9: diagrama de casos de uso del sistema.



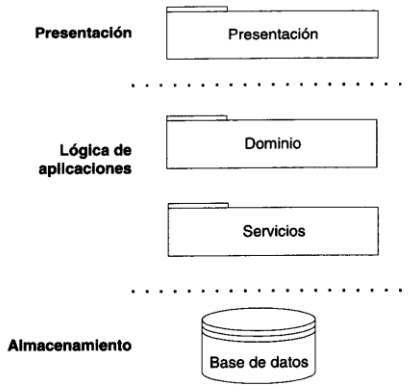
Fuente: personal.

La descripción detallada de los casos de uso del sistema se encuentra en el anexo II del documento.

7.3. ARQUITECTURA DEL SOFTWARE.

La arquitectura seleccionada es la *multicapas*, derivación del paradigma arquitectónico de tres capas, que consiste en usar la filosofía de paquetes con el fin de separar las operaciones del sistema en tres grandes grupos: el primer grupo es el encargado de soportar las vistas o la interface gráfica de usuario; el segundo grupo maneja la lógica del negocio y los servicios prestados, también es llamado "dominio"; y por última capa, la que contiene las bases de datos y/o los archivos. En la arquitectura multicapas, el dominio es separado en subcapas con el fin de separar la lógica del negocio (transacciones), de los servicios (funcionalidades del sistema) . (Larman, 2001)

Figura 9: Representación de una arquitectura de software multicapas.



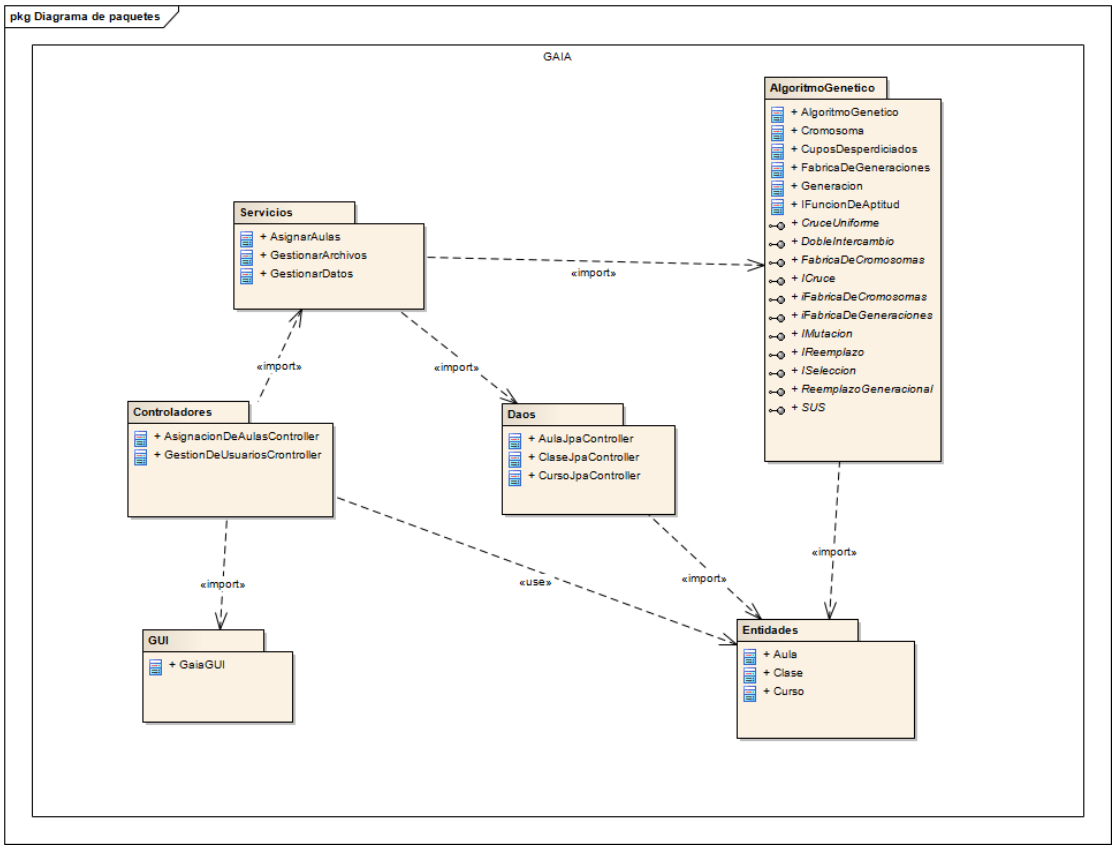
Fuente: (Larman, 2001)

7.4. DIAGRAMAS ESTÁTICOS.

7.4.1. Diagrama de Paquetes:

La figura 10 muestra el diagrama de paquetes, el cuál representa la agrupación lógica de las clases que describen el modelo del sistema desde un punto de vista estructural. La distribución de los paquetes se ha hecho en base a la arquitectura de capas planteada anteriormente.

Figura 10: Diagrama de paquetes.



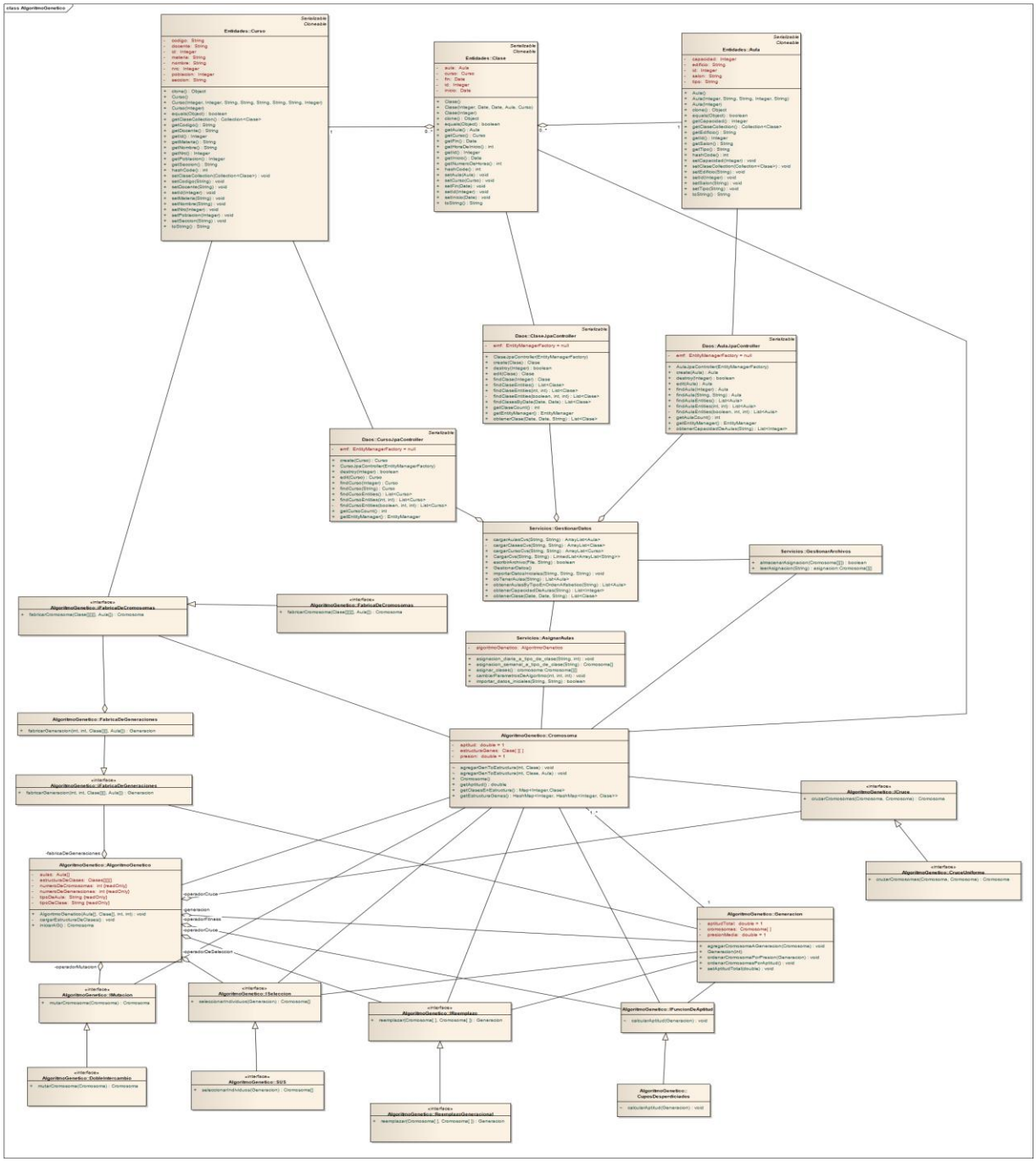
Fuente: propia.

Los paquetes en los que se divide el sistema son:

- Entidades: objetos básicos que almacenan la abstracción de los objetos básicos del sistema.
- Daos: objetos de comunicación entre la base de datos y el modelo del sistema.
- Algoritmo Genético: hace parte de la capa del dominio del sistema, establece el conjunto de clases que implementan el algoritmo diseñado en al capítulo anterior.
- Servicios: también hacen parte de la capa del dominio del sistema, implementan las funcionalidades especificadas en los casos de uso.
- Controladores: intermediarios entre la capa de visualización y la capa del dominio.
- GUI: capa de visualización del sistema.

7.4.2. Diagrama de Clases.

Figura 11: Diagrama de clases del sistema



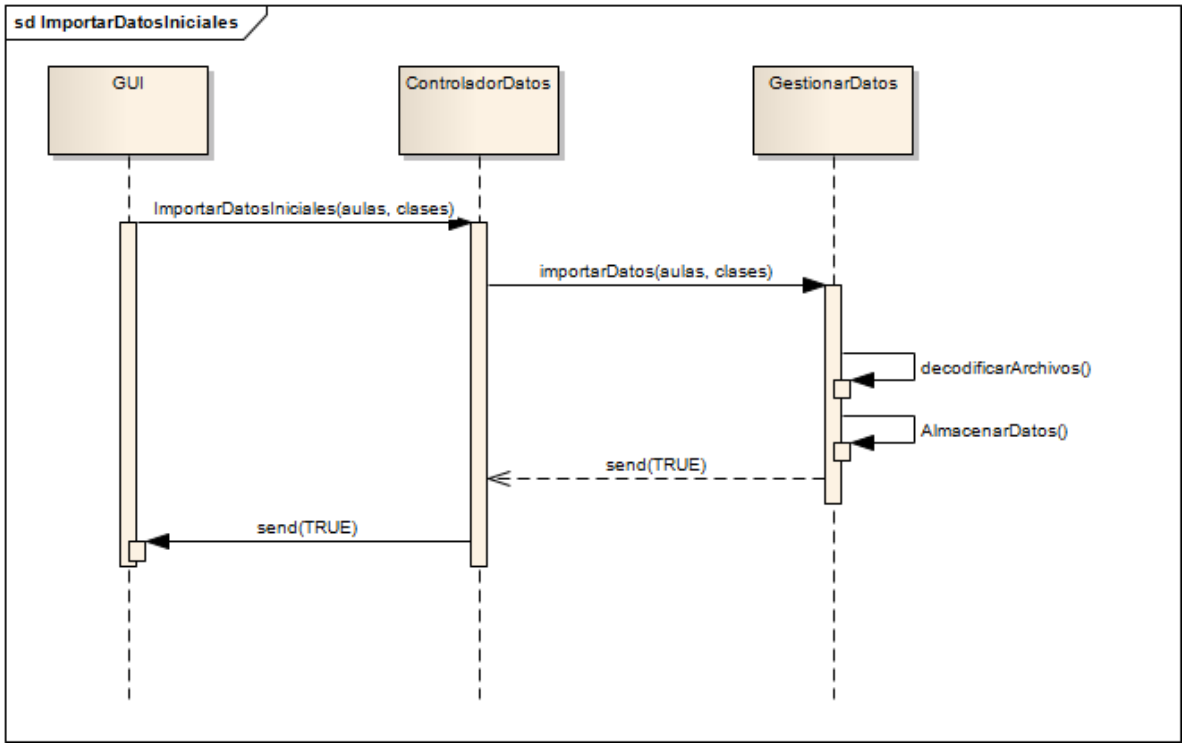
Fuente: propia.

La figura 11 muestra el diagrama de clases del sistema, el cuál es la representación gráfica de los bloques estructurales básicos del mismo.

7.5. Diagramas de secuencia.

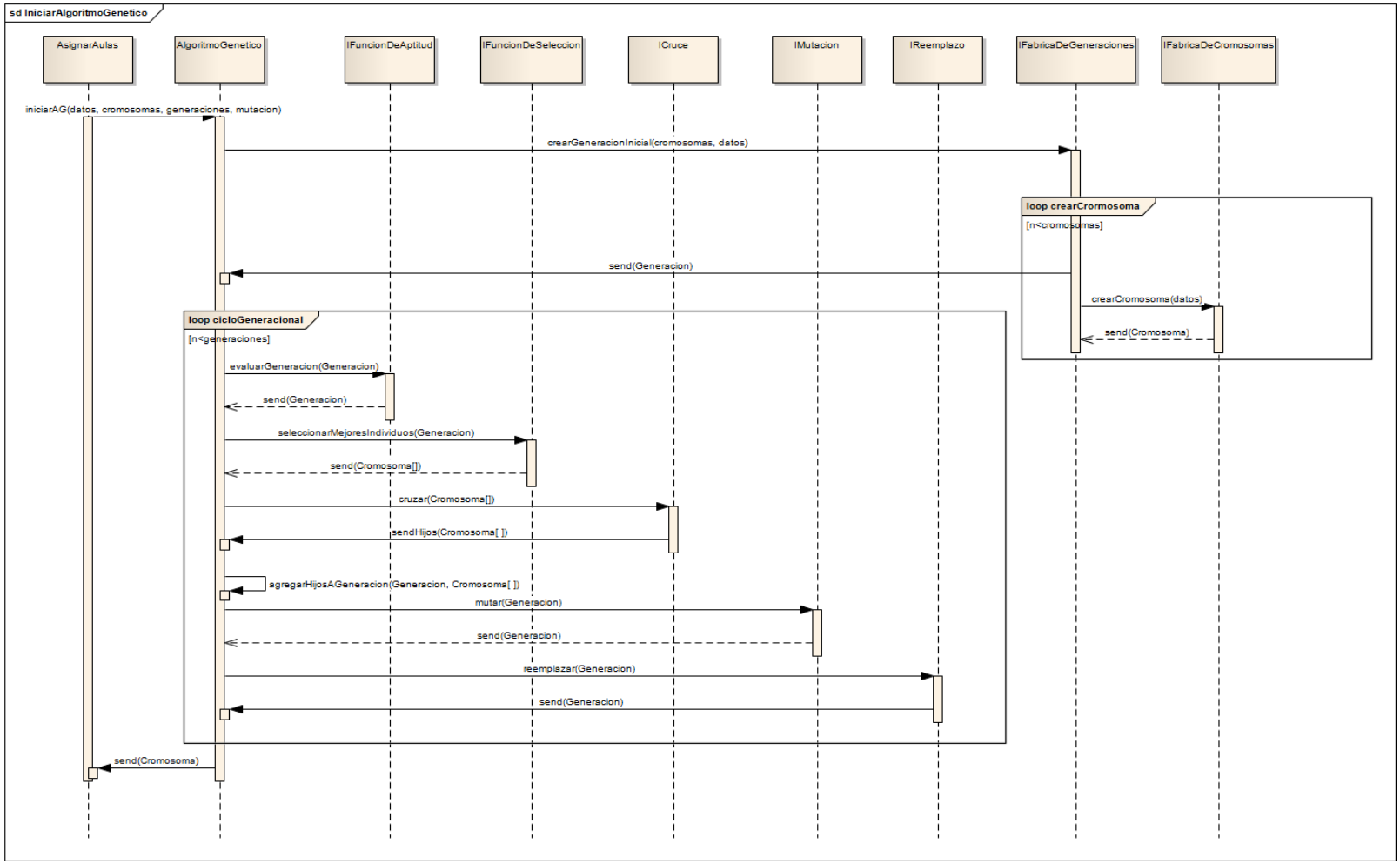
A continuación se muestran los diagramas de secuencias para los casos de usos más importantes y críticos en el sistema (figuras 12-14).

Figura 12: diagrama de secuencia del caso de uso “importar datos iniciales”.



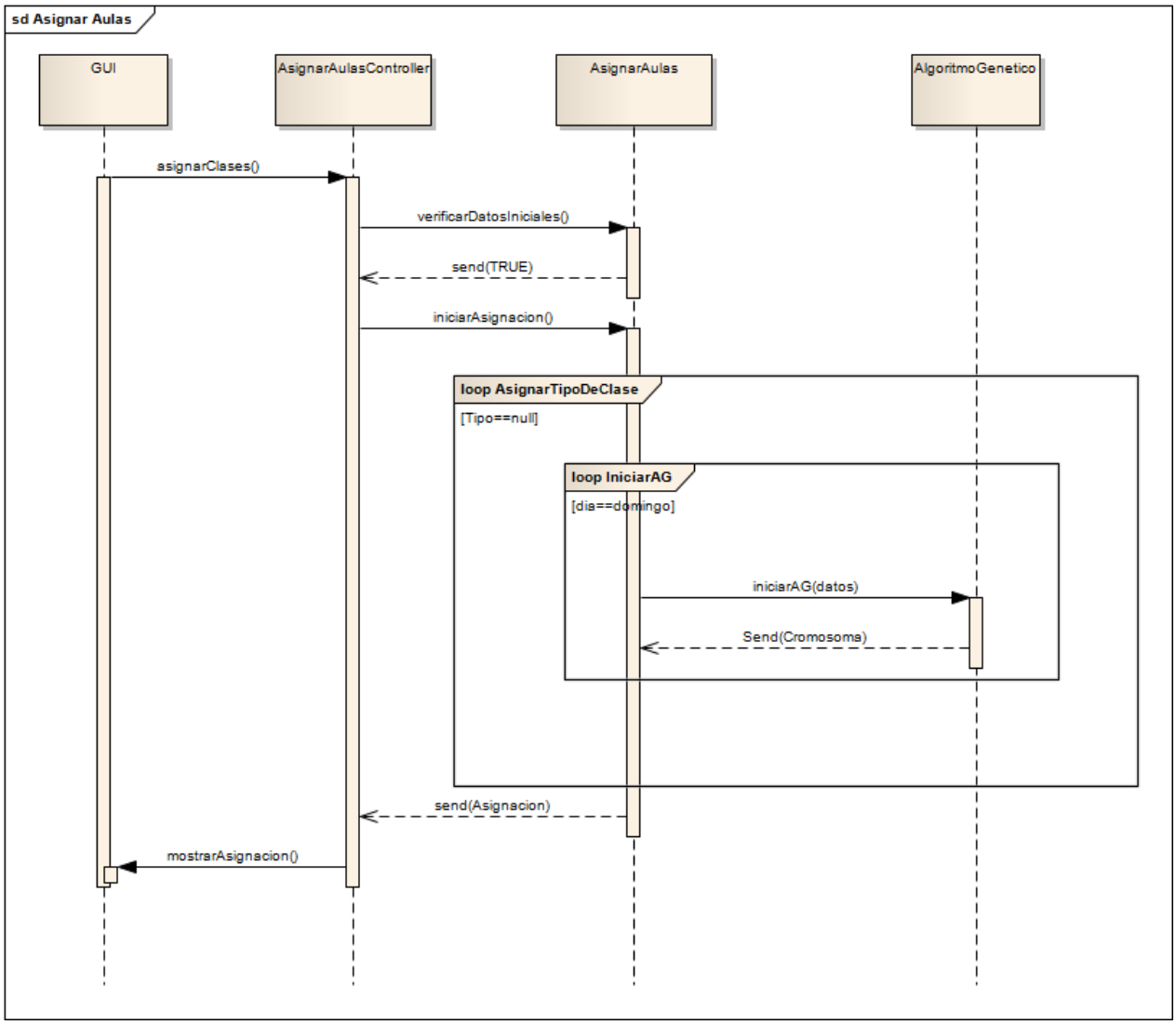
Fuente: propia.

Figura 13: diagrama de secuencia del caso de uso “iniciar algoritmo genético”.



Fuente: Propia.

Figura 14: Diagrama de secuencia del caso de uso “asignar aulas”.



Fuente: propia.

8. **IMPLEMENTACIÓN DEL PROTOTIPO**.

8.1. FUNCIONALIDADES A IMPLEMENTAR.

En el capítulo anterior se describió el diseño de todo el sistema de asignación de aulas para las clases de pregrado en la Universidad Tecnológica de Bolívar, basado en los requerimientos funcionales del mismo. El prototipo que ha sido construido, tiene como principal objetivo probar el algoritmo genético de asignación, por lo cual para su construcción solo se tendrán en cuenta aquellos requerimientos y casos de uso que sean pertinentes. Los casos de uso que serán implementados son:

- Gestionar algoritmo genético
- Iniciar algoritmo genético.
- Importar Datos iniciales.
- Cambiar parámetros de algoritmo genético.
- Mostrar Asignación.
- Exportar Asignación
- Mostrar estadísticas.

De acuerdo al listado anterior, solo serán implementados 6 de 12 requerimientos, lo cual implica que el prototipo posee el 50% de las funcionalidades planteadas en todo el sistema.

8.2. LENGUAJE DE PROGRAMACIÓN.

El lenguaje de programación en el que se ha implementado el prototipo del sistema GAIA es Java. Sus principales características son:

- Es una plataforma y lenguaje orientado a objetos.
- Diseñado originalmente por Sun Microsystems para electrodomésticos.
- Contiene una librería de clases base, llamada SDK.
- Usa una máquina virtual para la ejecución de un programa.
-

Los principales beneficios, por los que para la elaboración del prototipo se escogió este lenguaje fueron:

- Orientado a objetos: lo que permite abstraer de forma más natural el problema de dominio y usar las propiedades de la programación orientada a objetos, descritas en la sección 5.1.6.

- Interpretado e independiente de plataforma: puede ser ejecutado en la mayoría de sistemas operativos actuales. Solo es necesario la instalación de la máquina virtual de Java JRE (Java RunTime Enviroment).
- Dinámico y distribuido.
- Soporta la multi-tarea.
- Robusto y Seguro.

8.3. DESCRIPCIÓN DEL PROTOTIPO

La herramienta prototipo ha sido diseñada de una forma usable, reduciendo la cantidad de información y opciones mostradas en pantalla. Ésta consta de una barra de opciones superior, lista de aulas en la esquina lateral izquierda y la programación semanal de las clases para el aula seleccionada de la lista en la parte central (Ver Anexo III).

La barra de opciones, ubicada en la parte superior de la pantalla está dividida en 3 grupos de opciones:

1. Grupo de ejecución: en este se muestran las opciones que modifican la forma en la que son asignadas las aulas, estás son:
    a. Iniciar algoritmo: esta opción le permite al usuario iniciar la asignación de aulas para las clases teóricas de pregrado. Mientras

se ejecuta la asignación, una pantalla de espera será visualizada. (Ver anexo VIII).

b. Setup: esta opción le permite al usuario modificar los parámetros del algoritmo genético, los cuales son: número de cromosomas, número de generaciones del algoritmo y porcentaje de mutación. Ver anexo_.

2. Grupo de datos: en este se muestran las opciones que le permiten al usuario acceder a los datos y registros proporcionados por el sistema al asignar las aulas (Ver anexo IV). Las opciones de este grupo son:
    a. Exportar vista: exporta la programación del aula seleccionada en formato ".csv".
    b. Ver log: le permite al usuario ver el log de ejecución del algoritmo.
    c. Ver estadísticas: exporta un conjunto de datos de ejecución en formato ".csv", con el fin de permitir un análisis estadístico de la asignación.
    d. Exportar todo: exporta toda la asignación semanal de las clases de tipo teórico en formato ".csv".
3. Grupo de ayuda: en este grupo aparecen las opciones que brindan soporte al usuario. Ver anexo V.

9. ANÁLISIS Y DISCUSIÓN DE RESULTADOS.

El prototipo ha sido validado funcionalmente por medio de la comparación de los resultados obtenidos al evaluar con la función de aptitud la asignación de aulas para el primer periodo académico del año 2012, facilitado por Omer Salcedo, ingeniero del departamento SIRIUS en la universidad (Anexo I). Cabe resaltar, que del documento anteriormente mencionado se obtiene toda la información que necesita el prototipo para funcionar; por tal motivo, ha sido implementada una rutina que interpreta cada línea del formato .csv y lo transforma en un objeto tipo Clase, dependiendo las repeticiones semanales del curso.

La asignación de aulas para el primer periodo académico del 2012 (pregrado), aportada por SIRIUS es filtrada de acuerdo a los siguientes parámetros:

- Tipo de clase = TEO.
- Tipo de aula= AULA DE CLASE
- Campus= Campus Tecnológico

Como primera medida, se ha probado el algoritmo genético del prototipo con los resultados obtenidos al ejecutar la asignación de las clases del día miércoles (por ser el de mayor número de clases) con diferentes parámetros genéticos para hallar la mejor configuración. Las tablas 1, 2, 3 y 4 muestran los resultados

obtenidos al configurar los parámetros con un porcentaje de mutación del 0%, 1%, 2% y 3% respectivamente.

Tabla 1: Resultados de ejecutar el algoritmo genético para el día miércoles con porcentaje de mutación al 0%.

Cupos Desperdiciados (Mutación al 0%)			
Generaciones	**Número de Cromosomas**		
	5	**10**	**20**
10	3974	3960	3909
100	3963.66	3824,333	3835
1000	3955	3842	3729

Fuente: propia.

Tabla 2: Resultados de ejecutar el algoritmo genético para el día miércoles con porcentaje de mutación al 1%.

Cupos Desperdiciados (Mutación al 1%)			
Generaciones	**Número de Cromosomas**		
	5	**10**	**20**
10	3963	3944,66	3855
100	3922	3833	3816
1000	3933	3823	3719

Fuente: propia.

Tabla 3: Resultados de ejecutar el algoritmo genético para el día miércoles con porcentaje de mutación al 2%.

Cupos Desperdiciados (Mutación al 2%)			
Generaciones	**Número de Cromosomas**		
	5	**10**	**20**
10	3954	3880	3908
100	3975	3865	3807
1000	3899	3820	3899

Fuente: propia.

Tabla 4: Resultados de ejecutar el algoritmo genético para el día miércoles con porcentaje de mutación al 3%.

Cupos Desperdiciados (Mutación al 3%)			
Generaciones	**Número de Cromosomas**		
	5	**10**	**20**
10	3993	3919	3893
100	3911	3870	3803
1000	3947	3830	3746

Fuente: propia.

Los valores obtenidos y tabulados en la tablas 1, 2,3 y 4 provienen de la ejecución del algoritmo genético y cada valor resultante es la media de 10 resultados arrojados por el algoritmo para cada configuración. Como podemos observar al

analizar las tablas, la configuración más efectiva arrojada por la prueba está presente en la tabla 2. Esta configuración es:

- Número de cromosomas: 20
- Número de Generaciones: 1000.
- Porcentaje de mutación: 1%.

El mejor valor obtenido en las 10 iteraciones de la prueba bajo estos parámetros fue de 3709 cupos desperdiciados.

A continuación se procede a realizar la asignación de aulas para todas las clases teóricas de pregrado del primer período del 2012, configurando el algoritmo genético con los parámetros genéticos anteriormente hallados. El resultado está tabulado en la tabla 5.

Tabla 5: Comparación de cupos desperdiciados en la asignación realizada en la Universidad Tecnológica de Bolívar Vs la asignación realizada por el prototipo.

Cupos Desperdiciados Semanales			
Día	**Universidad**	**Prototipo**	**Diferencia**
Lunes	7450	3300	4150
Martes	7251	3736	3515
Miércoles	7696	3734	3962
Jueves	7045	2365	4680
Viernes	5536	2221	3315
Sábado	2540	805	1735
Total	37518	16161	21357

Fuente: propia.

Los resultados presentados por la tabla 5, muestra como la asignación de aulas para las clases teóricas de pregrado realizada por el prototipo es mejor que la asignación hecha en la Universidad Tecnológica de Bolívar, comparando solo el desperdicio de cupos. La programación de aulas hecha por el prototipo es 43% mejor, lo cual muestra su efectividad.

El tiempo de ejecución de la rutina de asignación fue de aproximadamente 4 minutos, en los que se asignó 57 aulas para 928 clases teóricas (Lunes-Sábado). Este resultado contrasta con los 3 días que demoraría hacer el proceso de asignación de aulas para las clases teóricas de forma manual, solo teniendo en cuenta el desperdicio de cupos (Entrevista José Barrios, No referenciada).

CONCLUSIONES

En la fase de la comprensión del problema se encontró que la asignación de aulas en la Universidad Tecnológica de Bolívar es un proceso de vital importancia, puesto que de éste depende la cantidad de clases que pueden ser ofertadas en un período académico. Actualmente este proceso demora alrededor de una semana de arduo trabajo, debido a que se lleva a cabo de forma manual y el sistema de apoyo no permite hacer un control del desperdicio de cupos, por lo cual no se puede optimizar esta programación.

Después de la comprensión del problema, en la fase de investigación se determinó que la asignación eficiente de aulas hace parte del problema de programación de cursos universitarios (*University Curse Timetable Problem*, UCTP), un problema de tipo *NP-Hard*, lo cual implica que el gasto computacional es elevado debido a la gran cantidad de combinaciones posibles para hallar una solución óptima. Este tipo de problemas son abordados por medio de metaheurísticas, dentro de las cuales se destacan los algoritmos genéticos. Estos últimos son considerados metaheurísticas evolutivas que usan la teoría neo-darwiniana de la supervivencia, con el fin de hallar una buena respuesta en un tiempo prudente y en el mejor de los casos hallar la solución óptima.

El resultado de la investigación arrojó la necesidad de diseñar un algoritmo genético capaz de minimizar los cupos desperdiciados al asignar una hora de clases en un aula. Para este algoritmo se realizó una heurística especial al generar la población inicial, basada primordialmente en agrupar las clases e impedir el sobrecupo. Las técnicas en las que se basaron los operadores genéticos se establecieron debido a su éxito en anteriores investigaciones similares, estas son:

- Selección: Selección estocástica universal (SUS).
- Cruce: Cruce uniforme.
- Mutación: Mutación por doble intercambio 2-opt.
- Reemplazo: Generacional.

El algoritmo resultante asigna eficientemente un conjunto de clases de un tipo determinado para un día académico, esta característica permite que sea reutilizado para hallar las asignaciones de todos los tipos de clase, para cada día de la semana.

Luego se diseñó un sistema de asignación de aulas que utiliza en su modelo de dominio el algoritmo genético diseñado previamente para signar eficientemente la programación diaria de cualquier tipo de clase. Esta herramienta también permite el análisis de estadísticas, la comparación entre asignaciones previas, la

visualización de una asignación y la persistencia de las programaciones realizadas. En la herramienta, el algoritmo genético fue abstraído y modelado por medio de patrones estructurales como es el patrón estrategia, con el fin de hacerlo extensible y permitir futuros cambios en los operadores genéticos.

Con el diseño realizado previamente se construyó una herramienta prototipo que posee el 70% de las funcionalidades del sistema completo. Este prototipo es capaz de realizar una asignación semanal de aulas para las clases de tipo teórico de pregrado en la Universidad Tecnológica De Bolívar, además permite la visualización de los resultados obtenidos, cambios de parámetros al algoritmo genético, visualización de estadísticas y exporta la programación en formato ".CSV".

El prototipo ha sido evaluado por medio de pruebas unitarias (Ver Anexos) y pruebas funcionales. En la prueba funcional se comparó el resultado obtenido al asignar las clases teóricas del primer semestre del 2012, teniendo como referencia el número de cupos desperdiciados en la asignación hecha por la universidad. Los resultados obtenidos fueron satisfactorios, el resultado de la prueba es (43%) mejor que el de la universidad.

Esta investigación puede ser considerada como un aporte para la automatización de todo el sistema de asignación de cursos de pregrado de la Universidad Tecnológica de Bolívar. Para trabajos futuros sería recomendable extender la

funcionalidad para poder asignar eficientemente los cursos de acuerdo a la oferta académica de los programas de pregrado. El prototipo construido es totalmente funcional y sus asignaciones podrían ser tenidas en cuenta por el departamento SIRIUS de la Universidad en el momento en el que se realice la programación para un nuevo período académico.

REFERENCIAS BIBLIOGRÁFICAS

- A. Hoffman. (1989). *Arguments on Evolution: A palenteologist's Perspective.* New York: Oxford University Press.
- Aldasht, M., Adi, S., & Alsaheb, M. (2010). *University Course Scheduling Using Evolutionary Algorithms.* Palestine Polytechnic University. Palestine Polytechnic University.
- Bremermann, H. J. (1962). *Optimization through evolution and recombination.* Wsashington D.C.: Spartan Books.
- Bremermann, H. J. (1958). *The evolution of intelligence. The nervous system as a model of its enviroment.* University of Washington, Seattle.
- Bremermann, H., & Rogson, M. (1964). *An evolution-type search method for convex sets.* ONR, California.
- Chaudhuri, A., & Kajal, D. (2010). Fuzzy Genetic Heuristic for University Course Timetable Problem. (I. Publication, Ed.) *nt. J. Advance. Soft Comput. , 2* (1).
- D. B. Fogel. (1995). *Evolucionary Computation. Toward a New Philosophy of Machine Intelligence.* New York: The institute of Electrical and Electronic Engineers.
- Davis, L. (1991). *Handbook of Genetic Algorithms.* Van Nostrand Reinhold.

- Duarte Muñoz, A. (2007). Metaheurísticas. (Dykinson, Ed.) *Ciencias experimentales y tecnología , 22*.
- Estévez Valencia, P. (2011). Optimización mediante algoritmos genéticos. *Anales del Instituto de Ingenieros de Chile* , 83-92.
- Falkenauer, E. (1999). The worth or uniform crossover. *Congress on evolutionary algorithms*, (pág. 783). USA.
- Friedman, G. J. (1956). *Selective Feedback Computers for Engineering Syntesis and Nervous System Analogy.* University of California, Los Ángeles.
- Gen, M., & Cheng, R. (2000). *Genetic Algoritms and Engineering Optimization.* New York: Jhon Wiley and Sons, Inc.
- Hernandez, S., & Fernandez, C. (2006). *Metodología de la investigación.* Mexico D.F.: Mc Graw Hill.
- Jong, K. D. (1975). *Analysis of the Behavior of a Class of Genetic Adaptive Systems.* University of Michigan, Michigan.
- Joyanes Aguilar, L. (1999). *Programación Orientada a Objetos.* Madrid, España: McGraw-Hill.
- L. J. Fogel, A. J. Owens, & M. J. Walsh. (1966). *Artificial Intelligence throught Simulated Evolution.* New York: Jhon Wiley & Sons, Inc.
- Larman, C. (2001). *UML y patrones.* Ciudad de Mexico: Pearson.

- Libersman, H. (2001). Metaheurísticas. In *Introducción a la invetsigación de operaciones* (pp. 617-655). McGraw Hill.
- Michalewicz, Z. (1996). *Evolucionary computation: practical issues.* New York: Springer-Verlag.
- Object Management Group. (n.d.). *OMG*. Retrieved 2012 12-Abril from http://www.omg.org/technology/readingroom/UML.htm
- Peña, V., & Zulmenzu, L. (2010). Estado del Job Shop Problem. *Universidad Técnica Federico Santa María* .
- Santana Quintero, L., & Coello Coello, C. (Diciembre de 2006). Una introducción a la computación evolutiva y algunas aplicaciones en economía y finanzas. (U. P. Olavide, Ed.) *Revista de métodos cuantitativos para la economía y la empresa* , 3-26.
- Sanz Montemayor, A. I Seminario sobre Sistemas Inteligentes 2006. En 2. Librería-Editorial Dykinson (Ed.), *Volumen 10 de Colección Actas*, *10.*
- Sigl, B., Golub, M., & Mornar, V. (2011). *Solving Timetable Scheduling Problem by Using Genetic Algorithms.* Zagreb (Cracia): University of Zagreb.
- Sommervile, I. (2005). *Ingeniería de Software.* Madrid: Pearson.
- Sparx Systems. (n.d.). *Sparx Systems*. Retrieved 2012 йил 10-Mayo from http://www.sparxsystems.com/resources/uml2_tutorial/

- Sywerda, G. (1989). Uniform Crossover in genetic algorithms. *3th International conference on genetic algorithms*, (págs. 5-8). USA.
- T. Back. (1996). *Evolutionary algorithms in Theory and Practice.* New York: Oxford University Press.
- Turing, A. M. (1950). *Computing Machinery and Intelligence.* Mind.

ANEXO I

Requerimientos funcionales del sistema.

ID	RF1
Descripción	Asignar Aulas
Descripción detallada	El sistema permitirá a usuarios con privilegios iniciar la asignación eficiente de aulas. Para tal fin, se ejecutará el algoritmo genético (RF3.1) para la asignación de todos los días para todos los tipos de clases.
Entradas	Fecha y hora de inicio.
Salidas	• Mensaje de información que indica que se ha realizado la programación. • Programación de aulas. • Mostrar Asignación (RF4).
Excepciones	• En caso de no poderse llevar a cabo la asignación, el sistema notificará al usuario del error y le pedirá intentar de nuevo.
Prioridad	Alta.

Fuente: propia.

ID	RF2
Descripción	Importar datos iniciales
Descripción detallada	El sistema permitirá a usuarios con privilegios importar la información de aulas y clases disponibles en la universidad.
Entradas	Aulas y clases en la universidad.
Salidas	• Mensaje de información que le informa al usuario que la importación ha sido exitosa.
Excepciones	• En caso de no poderse llevar a cabo la asignación, el sistema notificará al usuario del error y le pedirá intentar de nuevo.
Prioridad	Alta.

Fuente: propia.

ID	RF3
Descripción	Gestionar Algoritmo Genético
Descripción detallada	El sistema cuenta con un algoritmo genético capaz de realizar las asignaciones que minimicen el desperdicio de cupos en las clases programadas en la universidad.
Prioridad	Alta.

Fuente: propia.

ID	RF3.1
Descripción	Cambiar parámetros
Descripción detallada	La herramienta le permitirá al administrador del sistema el cambio de parámetros en el algoritmo genético, con el fin de modificar su comportamiento de búsqueda de acuerdo a las necesidades.
Entradas	Numero de cromosomas, número de generaciones, porcentaje de mutación.
Salidas	• Mensaje de información que le informa al usuario que el cambio de parámetros ha sido exitoso.
Excepciones	• En caso de no poderse llevar a cabo la asignación, el sistema notificará al usuario del error y le pedirá intentar de nuevo.
Prioridad	Media.

Fuente: propia.

ID	RF3.2
Descripción	Iniciar Algoritmo genético.
Descripción	El sistema iniciará el algoritmo genético que buscará la mejor

detallada	asignación posible de acuerdo a los parámetros establecidos con el fin de minimizar los cupos desperdiciados en la asignación de las clases de un tipo en particular, para un día determinado.
Entradas	Numero de cromosomas, número de generaciones, porcentaje de mutación, tipo de clase, tipo de aula asociada, día de la semana.
Salidas	• El algoritmo genético devolverá el primer cromosoma de la última generación, el cuál es una representación estructurada de la mejor solución encontrada.
Excepciones	• En caso de no poderse llevar a cabo la ejecución del algoritmo, el sistema notificará al usuario del error y le pedirá intentar de nuevo.
Prioridad	Media.

Fuente: propia.

ID	RF4
Descripción	Mostrar Asignación
Descripción	El sistema mostrará al usuario una representación de la

detallada	solución obtenida por la rutina de asignación que esté actualmente en memoria o haya sido cargada desde un archivo.
Entradas	Asignación
Salidas	• Representación gráfica del cromosoma.
Excepciones	• En caso de no poderse llevar a cabo la ejecución del algoritmo, el sistema notificará al usuario del error y le pedirá intentar de nuevo.
Prioridad	Media.

Fuente: propia.

ID	RF5
Descripción	Mostrar Estadísticas
Descripción detallada	El sistema permitirá que un usuario avanzado y con privilegios observe las estadísticas de ejecución del algoritmo genético, como lo son: • Tiempo de ejecución. • Numero de cupos sobrantes.

	• Aulas menos usadas. • Aulas más usadas.
Entradas	Asignación
Salidas	Estadísticas de ejecución.
Excepciones	Ninguna.
Prioridad	Media.

Fuente: propia.

ID	RF6
Descripción	Importar / Exportar asignación
Descripción detallada	El sistema permitirá que un usuario con privilegios importe o exporte una asignación creada anteriormente.
Entradas	Asignación
Salidas	• Representación gráfica de la asignación.
Excepciones	• En caso de no encontrar el archivo a importar, notificar el error al usuario. • Si el archivo seleccionado para la lectura no es un cromosoma válido, notificar al usuario de la inconsistencia.
Prioridad	Media.

Fuente: propia.

ID	RF7
Descripción	Comparar Resultados
Descripción detallada	El sistema permitirá que un usuario pueda comprar en una tabla las estadísticas de dos o más cromosomas.
Entradas	Asignación 1, Asignación 2
Salidas	• Comparación de las asignaciones.
Excepciones	• En caso de no poderse llevar a cabo la ejecución del algoritmo, el sistema notificará al usuario del error y le pedirá intentar de nuevo.
Prioridad	Media.

Fuente: propia.

ID	RF8
Descripción	Gestionar usuarios
Descripción detallada	La gestión de usuarios del sistema permitirá: • Añadir un nuevo usuario. • Editar un usuario existente. • Eliminar un usuario. • Mostrar información de usuario (s).
Entradas	Usuario
Salidas	• Comparación de las asignaciones.
Excepciones	• En caso de solicitar realizar una operación con un usuario inexistente, notificar del error
Prioridad	Media.

Fuente: propia

ID	RF9
Descripción	Autenticar Usuario

Descripción detallada	Se cotejará el nombre del usuario con la contraseña.
Entradas	Nombre de usuario, contraseña.
Salidas	Usuario autenticado con éxito.
Excepciones	• En caso de solicitar realizar una operación con un usuario inexistente, notificar del error. • Si no cotejan el usuario y la contraseña se notificará del error y se impedirá el accedo a la aplicación.
Prioridad	Media.

Fuente: propia

ANEXO II

Casos de Uso del sistema.

<table>
<tr><td>Nombre</td><td>Asignar Aulas</td></tr>
<tr><td>Actores</td><td>Personal SIRIUS, administrador</td></tr>
<tr><td>Propósito</td><td>Crear la programación de aulas para todos los cursos de pregrado de la universidad tecnológica de bolívar.</td></tr>
<tr><td>Resumen</td><td>Un usuario con privilegios selecciona la opción asignar aulas y el sistema recopila las asignaciones individuales de cada día de la semana para cada tipo de clases. Luego el sistema mostrará los resultados.</td></tr>
<tr><td>Tipo</td><td>Primario</td></tr>
<tr><td>R Cruzadas</td><td>RF3, RF4,</td></tr>
<tr><td>Eventos</td><td>
<table>
<tr><td>Acción del actor</td><td>Acción del sistema</td></tr>
<tr><td>1-Un Usuario con privilegios selecciona la opción “iniciar asignación”.</td><td>2-El sistema verifica la existencia de los datos iniciales.

3- El sistema obtiene un listado de los tipos de clases cargados</td></tr>
</table>
</td></tr>
</table>

			y los tipos de aulas asociados a dichas clases. 4- El sistema ejecuta un algoritmo genético para cada tipo de clase, cada día de la semana y almacena temporalmente en memoria cada resultado. 5-Llamar al caso de uso “Mostrar Asignación”, para hacer una representación de la solución.
Curso Alterno	**Línea 2:** Si no están cargados los datos iniciales, el sistema solicitará su carga.		

Fuente: propio.

Nombre	Importar datos iniciales
Actores	Administrador
Propósito	Cargar en el sistema la información de aulas y clases programadas en la universidad.
Resumen	Un usuario administrador, selecciona la opción insertar datos iniciales e indica la ruta en la cual se encuentran los archivos que contienen la información de aulas.
Tipo	Primario
R Cruzadas	RF2
Eventos	

Acción del actor	Acción del sistema
1-Un Usuario administrador selecciona la opción “importar datos iniciales”.	2-El sistema verifica la existencia de los datos iniciales. 3-El sistema requiere la dirección del archivo de aulas y

		el archivo de clases.
	4- El usuario ingresa la ruta de acceso a los archivos de aulas y clases.	5- Leer el archivo de aulas. 7- Validar el archivo de aulas. 8- Importar aulas. 9- Leer archivo de clases. 10-Validar archivo de clases. 11-Importar clases. 12-Notificar importación exitosa
Curso Alterno	**Línea 2:** Si existen datos iniciales, el sistema le preguntará al usuario si desea que sean reemplazados.	
	Línea 5: Si el archivo de aulas no existe, abortar el proceso y notificar.	
	Línea 7: Si la estructura del archivo de aulas no corresponde con el formato establecido, abortar el proceso y notificar.	
	Línea 9: Si el archivo de clases no existe, abortar el proceso y notificar.	

	Línea 10: Si la estructura del archivo de clases no corresponde con el formato establecido, abortar el proceso y notificar.

Fuente: propio.

Nombre	Cambiar parámetros de algoritmo genético
Actores	Administrador
Propósito	Cambiar los parámetros del algoritmo genético.
Resumen	Un usuario administrador, selecciona la opción “cambiar parámetros” y modifica el comportamiento del algoritmo genético.
Tipo	Primario
R Cruzadas	RF3, RF3.1
Eventos	

Acción del actor	Acción del sistema
1-Un Usuario administrador selecciona la opción “cambiar parámetros”.	2-El sistema pide los nuevos parámetros.
4- El usuario ingresa al sistema los siguientes parámetros:	5- El sistema valida los parámetros.

<table>
<tr><td></td><td><table><tr><td>• Numero de cromosomas.
• Numero de generaciones.
• Porcentaje de mutación.</td><td>6- Preguntar al usuario si confirma los nuevos parámetros.
7- Cambiar los parámetros de ejecución.
8-Notificar al usuario del cambio.</td></tr></table></td></tr>
<tr><td>Curso Alterno</td><td><table><tr><td>Línea 5: Si los parámetros no son válidos, se indica al usuario que deben ser corregidos.</td></tr><tr><td>Línea 6: Si el usuario no confirma los nuevos parámetros, abortar el proceso.</td></tr></table></td></tr>
</table>

Fuente: propio.

<table>
<tr><td>Nombre</td><td>Iniciar Algoritmo Genético</td></tr>
<tr><td>Actores</td><td>Administrador, Personal SIRIUS</td></tr>
<tr><td>Propósito</td><td>Ejecutar el algoritmo genético y hallar la mejor asignación para un tipo específico de clase en un día de la semana.</td></tr>
<tr><td>Resumen</td><td>El sistema ejecuta un algoritmo genético que halla la mejor asignación posible para un tipo de clase, que usa a su vez un tipo de aula y ha sido programada en un día específico de la semana.</td></tr>
<tr><td>Tipo</td><td>Primario</td></tr>
<tr><td>R Cruzadas</td><td>RF3, RF3.1, RF1</td></tr>
<tr><td>Eventos</td><td>
<table>
<tr><td>Acción del actor</td><td>Acción del sistema</td></tr>
<tr><td>1-Un Usuario administrador selecciona la opción “iniciar algoritmo genético”.</td><td>2-el sistema verifica que existan datos iniciales.
3-Pedir tipo de clase, tipo de aula, día de la semana.</td></tr>
<tr><td>4- El usuario envía el tipo de clase, tipo de aula y día de la</td><td>5- El sistema valida los parámetros.</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><td></td><td><table><tr><td>semana.</td><td>6- Obtener aulas
6- Iniciar población inicial.
7- Construir generaciones.
8-Eviar mejor cromosoma</td></tr></table></td></tr>
<tr><td>Curso Alterno</td><td><table><tr><td>Línea 2: Si no están cargados los datos iniciales, el sistema solicitará su carga.</td></tr><tr><td>Línea 5: Si los parámetros no son válidos, se indica al usuario que deben ser corregidos.</td></tr></table></td></tr>
</table>

Fuente: propio.

<table>
<tr><td>Nombre</td><td>Mostrar asignación</td></tr>
<tr><td>Actores</td><td>Administrador, Personal SIRIUS, Personal UTB</td></tr>
<tr><td>Propósito</td><td>Mostrar en representación tabular una asignación</td></tr>
<tr><td>Resumen</td><td>El sistema mostrará de forma tabular la representación de una asignación hecha previamente.</td></tr>
<tr><td>Tipo</td><td>Primario</td></tr>
<tr><td>R Cruzadas</td><td>RF4</td></tr>
<tr><td>Eventos</td><td>
<table>
<tr><td>Acción del actor</td><td>Acción del sistema</td></tr>
<tr><td>1-Un Usuario administrador selecciona la opción “ver asignación”.</td><td>2-El sistema le pregunta que asignación desea ver.</td></tr>
<tr><td>3- El usuario selecciona la asignación</td><td>5- Representar en forma tabular la asignación.</td></tr>
</table>
</td></tr>
</table>

Curso **Alterno**	

Fuente: propio.

Nombre	Mostrar estadísticas
Actores	Administrador, Personal SIRIUS, Personal UTB
Propósito	Mostrar los datos estadísticos al generar una asignación.
Resumen	El sistema mostrará las estadísticas de la última asignación realizada. Por lo cual se visualizarán las siguientes variables: • Cupos desperdiciados. • Numero de Generaciones • Numero de Datos. • Porcentaje de mutación • Tiempo de ejecución
Tipo	Secundario
R Cruzadas	RF5
Eventos	

Acción del actor	Acción del sistema
1-Un Usuario administrador selecciona la opción “ver estadísticas”.	2-El sistema le pregunta que asignación desea ver.
3- El usuario selecciona la asignación	5- Mostrar las estadísticas de la solución.

Curso Alterno	

Fuente: propio.

Nombre	Comparar resultados
Actores	Administrador, Personal SIRIUS, Personal UTB
Propósito	Comparar un conjunto de soluciones creadas con anterioridad
Resumen	El sistema mostrará una tabla comparativa con los resultados de las siguientes variables: • Cupos desperdiciados. • Numero de Generaciones • Numero de Datos. • Porcentaje de mutación • Tiempo de ejecución
Tipo	Secundario
R Cruzadas	RF7
Eventos	

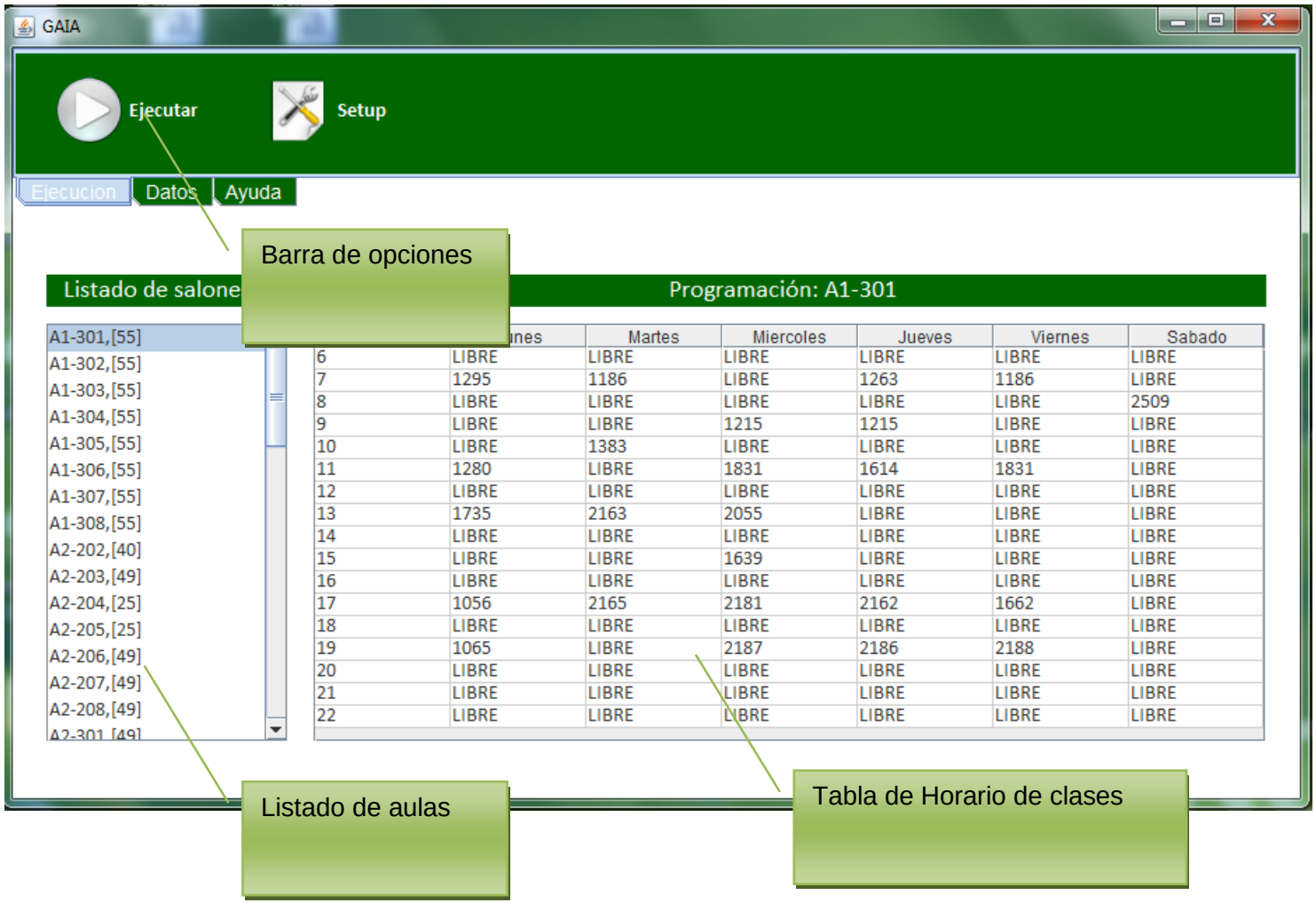
Acción del actor	Acción del sistema
1-Un Usuario administrador selecciona la opción “comparar soluciones”.	2-El sistema le pregunta las asignaciones que desea comparar.
3- El usuario envía las asignaciones que desea comparar.	4-Procesar las asignaciones seleccionadas. 5- Representar en forma tabular la asignación.

Curso Alterno	

Fuente: propio.

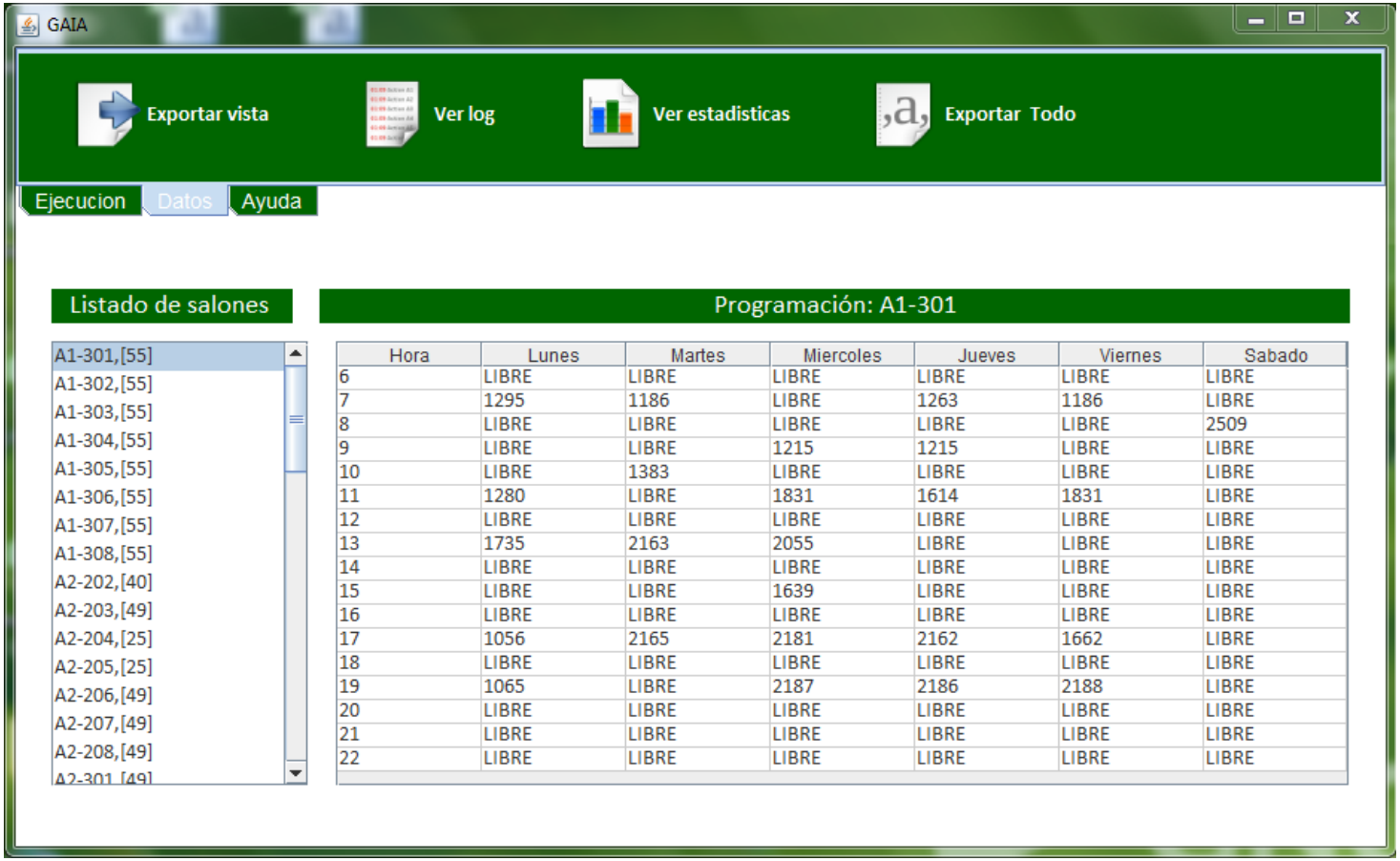
ANEXO III

División de la pantalla inicial del prototipo.



ANEXO IV

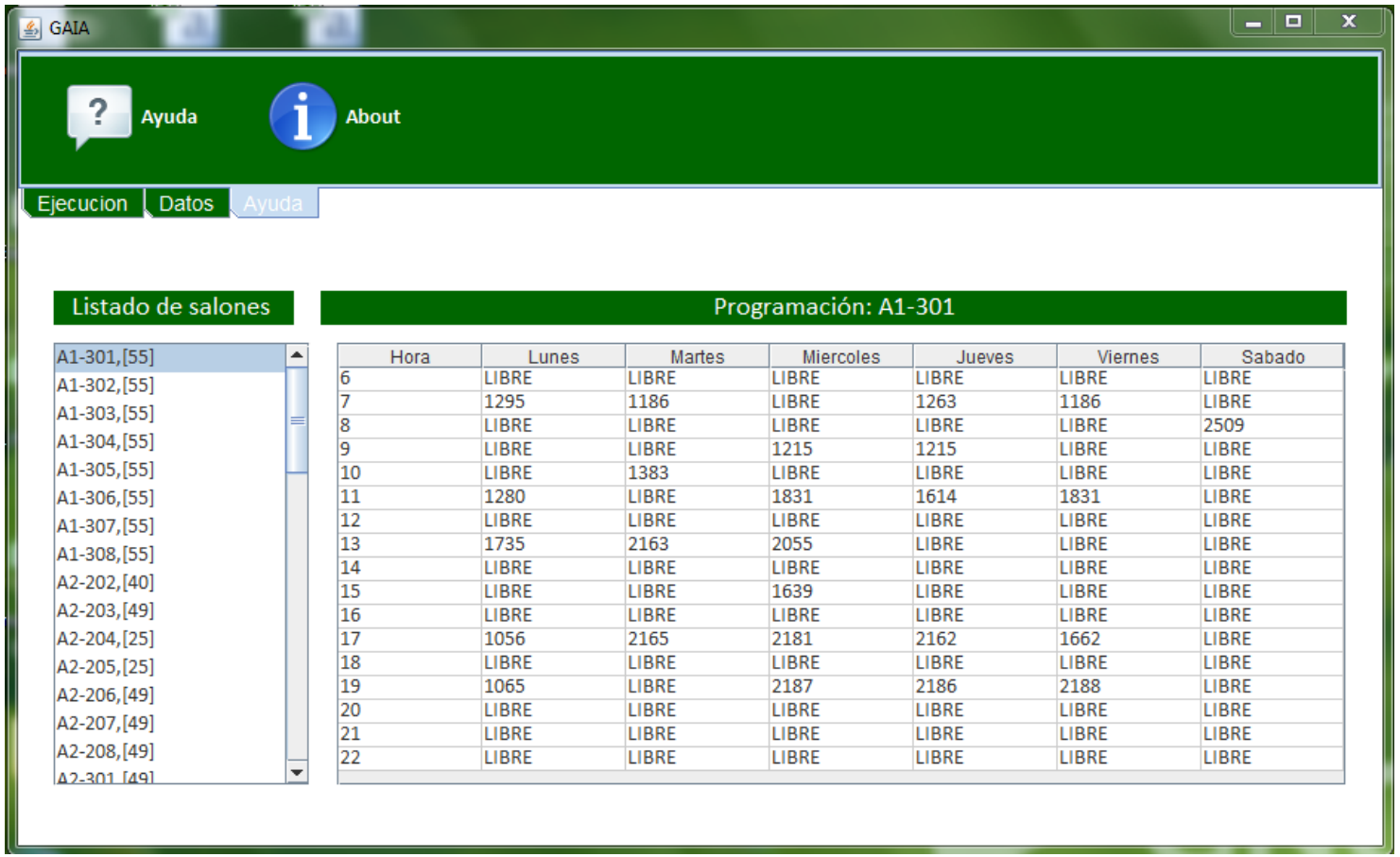
Opciones de manejo de datos en el prototipo



Hora	Lunes	Martes	Miercoles	Jueves	Viernes	Sabado
6	LIBRE	LIBRE	LIBRE	LIBRE	LIBRE	LIBRE
7	1295	1186	LIBRE	1263	1186	LIBRE
8	LIBRE	LIBRE	LIBRE	LIBRE	LIBRE	2509
9	LIBRE	LIBRE	1215	1215	LIBRE	LIBRE
10	LIBRE	1383	LIBRE	LIBRE	LIBRE	LIBRE
11	1280	LIBRE	1831	1614	1831	LIBRE
12	LIBRE	LIBRE	LIBRE	LIBRE	LIBRE	LIBRE
13	1735	2163	2055	LIBRE	LIBRE	LIBRE
14	LIBRE	LIBRE	LIBRE	LIBRE	LIBRE	LIBRE
15	LIBRE	LIBRE	1639	LIBRE	LIBRE	LIBRE
16	LIBRE	LIBRE	LIBRE	LIBRE	LIBRE	LIBRE
17	1056	2165	2181	2162	1662	LIBRE
18	LIBRE	LIBRE	LIBRE	LIBRE	LIBRE	LIBRE
19	1065	LIBRE	2187	2186	2188	LIBRE
20	LIBRE	LIBRE	LIBRE	LIBRE	LIBRE	LIBRE
21	LIBRE	LIBRE	LIBRE	LIBRE	LIBRE	LIBRE
22	LIBRE	LIBRE	LIBRE	LIBRE	LIBRE	LIBRE

ANEXO V

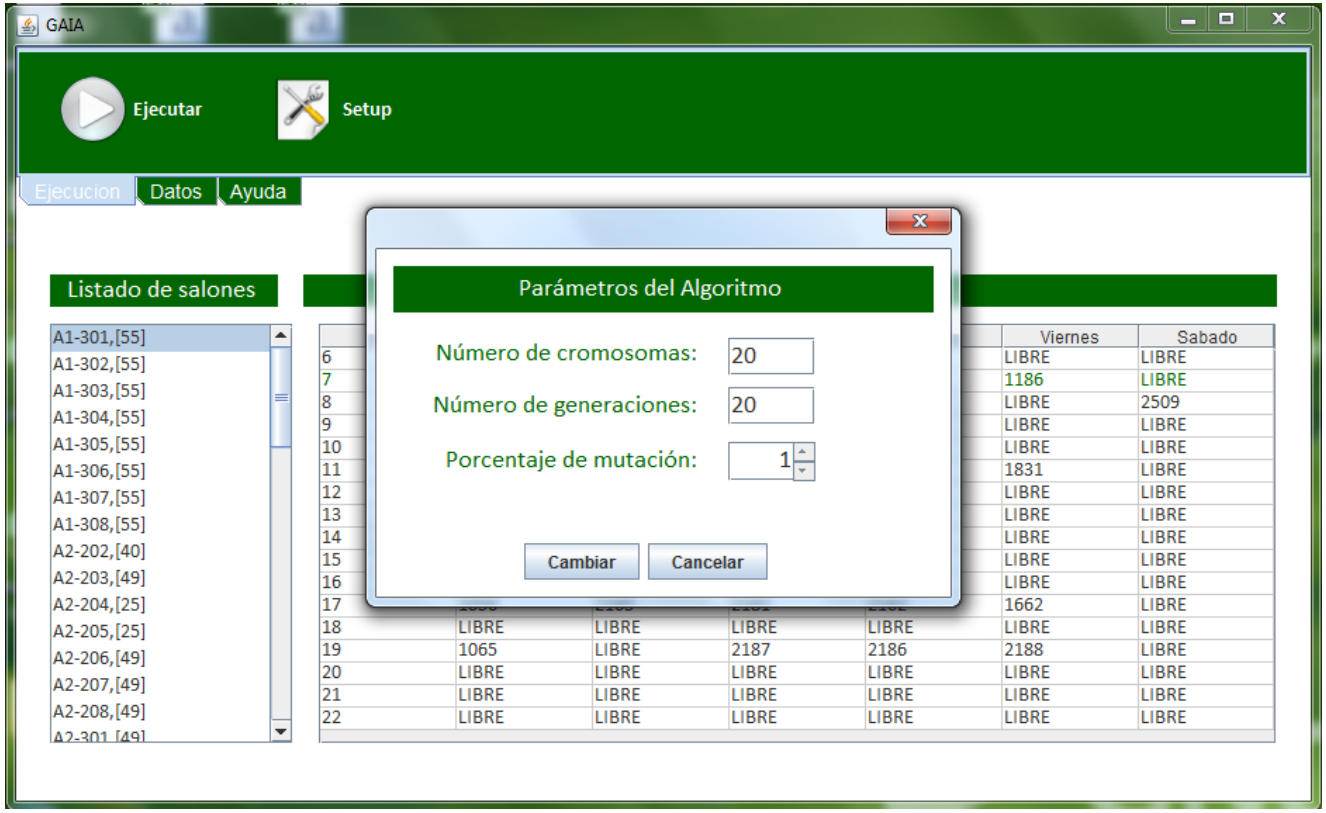
Opciones de ayuda en el prototipo.



Hora	Lunes	Martes	Miercoles	Jueves	Viernes	Sabado
6	LIBRE	LIBRE	LIBRE	LIBRE	LIBRE	LIBRE
7	1295	1186	LIBRE	1263	1186	LIBRE
8	LIBRE	LIBRE	LIBRE	LIBRE	LIBRE	2509
9	LIBRE	LIBRE	1215	1215	LIBRE	LIBRE
10	LIBRE	1383	LIBRE	LIBRE	LIBRE	LIBRE
11	1280	LIBRE	1831	1614	1831	LIBRE
12	LIBRE	LIBRE	LIBRE	LIBRE	LIBRE	LIBRE
13	1735	2163	2055	LIBRE	LIBRE	LIBRE
14	LIBRE	LIBRE	LIBRE	LIBRE	LIBRE	LIBRE
15	LIBRE	LIBRE	1639	LIBRE	LIBRE	LIBRE
16	LIBRE	LIBRE	LIBRE	LIBRE	LIBRE	LIBRE
17	1056	2165	2181	2162	1662	LIBRE
18	LIBRE	LIBRE	LIBRE	LIBRE	LIBRE	LIBRE
19	1065	LIBRE	2187	2186	2188	LIBRE
20	LIBRE	LIBRE	LIBRE	LIBRE	LIBRE	LIBRE
21	LIBRE	LIBRE	LIBRE	LIBRE	LIBRE	LIBRE
22	LIBRE	LIBRE	LIBRE	LIBRE	LIBRE	LIBRE

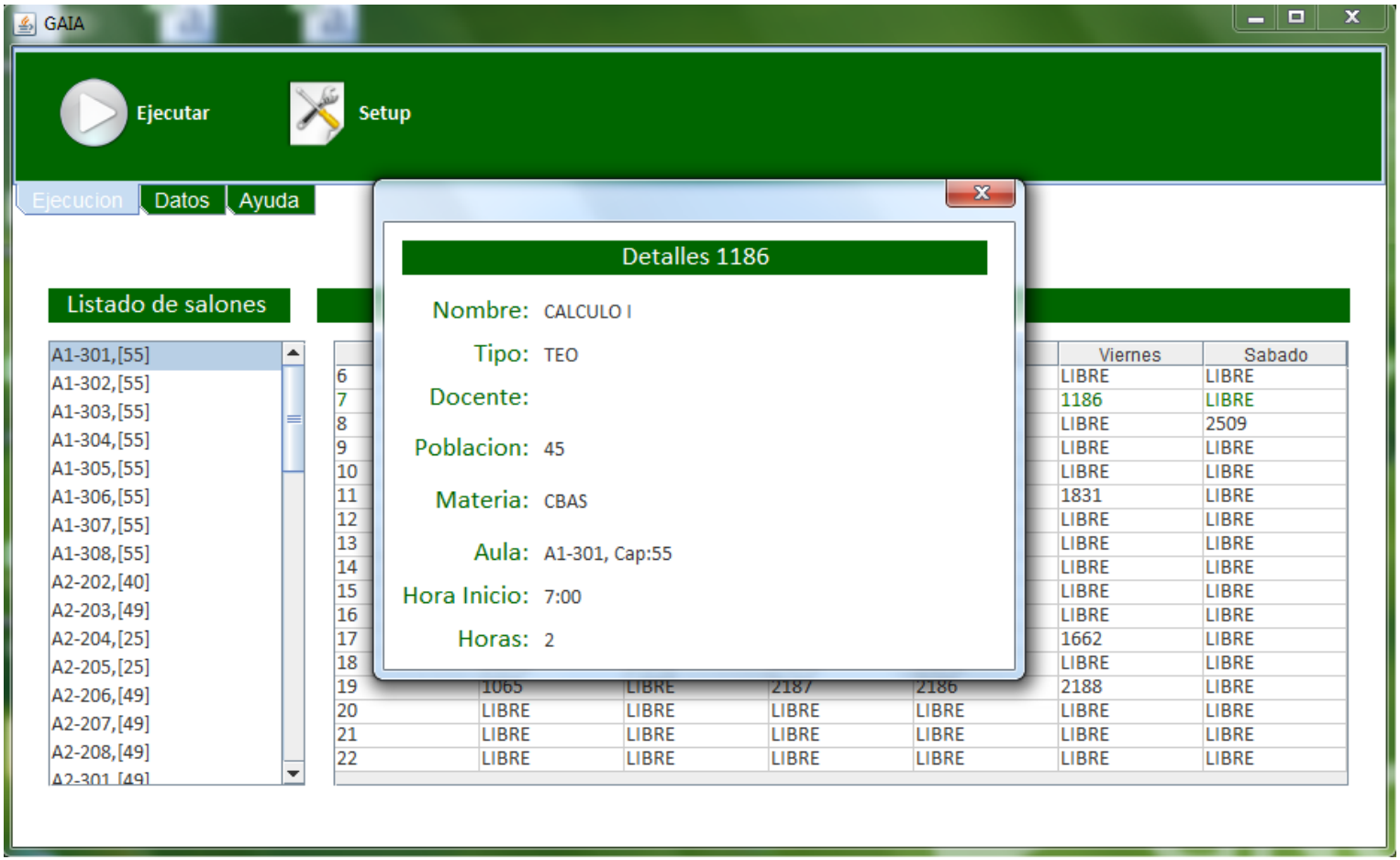
ANEXO VI

Pantalla “Cambiar parámetros de algoritmo genético” en el prototipo.



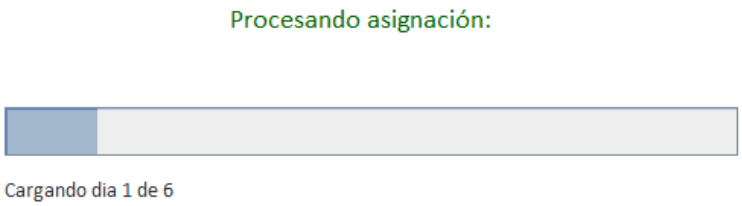
ANEXO VII

Pantalla “Información de clase” mostrada en el prototipo al seleccionar un NRC.



ANEXO VIII

Ventana de espera mostrada en el prototipo al ejecutar una asignación.



ANEXO IX

Resultados adjuntos.

Como documento adjunto se encuentra el archivo AnálisisResultadosGAIA.xslx, el cuál contiene la comparación de la asignación realizada por la Universidad Tecnológica de Bolívar y la asignación realizada por el prototipo.

ANEXO X

Orientaciones técnicas.

A continuación se describen los detalles técnicos del proyecto.

- IDE: Netbeans 7.1, gratuito, disponible en http://www.netbeans.org.
- Base De Datos: MySQL 5.1.44, libre, disponible en http://www.mysql.org
- Manejo de persistencia: Java Persistence Api (EclipseLink 2.0).
- Para la ejecución del prototipo es necesario crear una base de datos llamanda “gaia” y un usuario llamado “gaia_admin” con contraseña “4dm1n” e importar la información inicial que se encuentra en el archivo datos_iniciales.sql, luego ejecutar el .jar de la aplicación.