UNIVERSIDAD NACIONAL DEL CALLAO

FACULTAD DE INGENIERIA INDUSTRIAL Y DE SISTEMAS

ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS

TESIS
"ALGORITMO GENÉTICO APLICADO EN LA PROGRAMACIÓN ACADÉMICA DE UNA ESCUELA PROFESIONAL EN UNA FACULTAD UNIVERSITARIA"

PARA OPTAR EL TÍTULO DE INGENIERO DE SISTEMAS

AUTOR:

HENRY CASTRO POCCO
BACH. INGENIERÍA DE SISTEMAS

CALLAO, JULIO 2016
PERÚ

UNIVERSIDAD NACIONAL DEL CALLAO

Facultad de Ingeniería Industrial y de Sistemas

Av. Juan Pablo II N° 306 Bellavista-Callao

ACTA DE SUSTENTACIÓN

En la ciudad del Callao a los veintiocho días del mes de mayo del 2016, se reunieron en el local del Auditórium de la Facultad de Ingeniería Industrial y de Sistemas de la Universidad Nacional del Callao, la Mg. Erika Zevallos Vera en calidad de presidente del jurado, el Ing. José Farfán Aguilar en calidad de secretario del jurado y del Mg. Osmart Morales Chalco en calidad de vocal del jurado, con el objeto de calificar la sustentación de la tesis titulada:

"ALGORITMO GENÉTICO APLICADO EN LA PROGRAMACIÓN ACADÉMICA DE UNA ESCUELA PROFESIONAL EN UNA FACULTAD UNIVERSITARIA".

Presentado por el bachiller en ingeniería de sistemas: HENRY CASTRO POCCO
Para optar el título profesional de: INGENIERO DE SISTEMAS
Obteniendo el calificativo de:

16 (DIECISÉIS)

En fe de lo cual, se suscribió la presente acta firmada por el señor presidente del jurado y los miembros integrantes del mismo.

Mg. Erika Zevallos Vera
Presidente

Ing. José Farfán Aguilar
Secretario

Mg. Osmart Morales Chalco
Vocal

DEDICATORIA

A mis padres Jorge y Luz que siempre cultivaron en mi persona el deseo de llegar a ser un profesional de éxito

AGRADECIMIENTO

Un especial reconocimiento a mi alma mater la Universidad Nacional del Callao, a mis queridos docentes de la Escuela Profesional de Ingeniería de Sistemas por sus sabias enseñanzas depositadas en mi persona

INDICE

INDICE DE CUADROS

INDICE DE FIGURAS

RESUMEN

Actualmente en el país existen ciento cuarenta idos universidades, de las cuales treinta idos son públicas y el resto son privadas, el sistema de universidad peruano cuenta con alrededor de 1136 facultades y 2272 escuelas profesionales.

La Facultad de Ingeniería Industrial y de Sistemas de la Universidad Nacional del Callao, cuenta con dos escuelas profesionales: La Escuela Profesional de Ingeniería Industrial y la Escuela. Profesional de Ingeniería de Sistemas, en estas escuelas se desarrollan al año dos semestres académicos que implican el desarrollo programaciones horarias semestrales para cada escuela profesional. Mediante la programación horaria se busca optimizar el uso de los recursos de aulas, docentes y tiempo utilizados en estos procesos y de esta forma optimizar el proceso de matrícula de los estudiantes.

Desarrollar una programación horaria demanda de recursos y tiempo, el modelamiento de una programación horaria puede ser resuelto mediante la aplicación de algoritmos genéticos que son tipo timetabling de la forma NP-completo. Los algoritmos genéticos son técnicas de optimización que postulan la supervivencia del más apto, que se sustentan en la teoría de la evolución de las especies postulada por Charles Darwing.

En el presente estudio utilizamos algoritmos genéticos para optimizar la programación horaria semestral de la Escuela Profesional de una Facultad Universitaria en la Universidad Nacional del Callao, ello implica un mejor servicio al estudiante al haber menos conflictos de choques o cruces horarios.

Finalmente mediante el algoritmo genético de asignación de aulas y de docentes se obtiene mejores resultados que los costos alcanzados por las personas encargadas de realizar la misma tarea, llegándose a alcanzar una mejora porcentual con respecto al experto de 73.7% y 91.67% en la asignación de docentes y aulas respectivamente.

Al automatizar la asignación de aulas y docentes se reduce notablemente el tiempo, la misma tarea antes era realizada en un promedio 5 a 6 días, mientras que con el algoritmo genético ya se están obteniendo resultados comparados con los del experto antes de los 5 primeros minutos en promedio.

ABSTRACT

Currently the country has gone hundred forty universities, of which thirty are gone public and the rest are private, the Peruvian university system has about 1136 schools and 2272 vocational schools.

School of Industrial and Systems Engineering of the National University of Callao, has two professional schools: the Professional School of Industrial Engineering and the School. Professional Systems Engineering in these schools develop two academic semesters per year involving the development semiannual time schedules for each professional school. By the time schedule seeks to optimize the use of resources of classrooms, teachers and time used in these processes and thus optimize the process of student enrollment.

Develop a scheduling demand for resources and time, modeling a time schedule can be solved through the application of genetic algorithms that are timetabling type NP-complete form. Genetic algorithms are optimization techniques that posit the survival of the fittest, which are based on the theory of evolution of species by Charles Darwin postulated.

In the present study we use genetic algorithms to optimize the biannual scheduling of the Professional School of University Faculty, of the National University of Callao, this means a better service to the student to have fewer conflicts of shock or crosses schedules.

Finally, through genetic algorithm allocation of classrooms and teachers better results are obtained than the costs made by the persons responsible for

performing the same task, getting itself to reach a percentage improvement over the expert 73.7% and 91.67% in the allocation of teachers and classrooms respectively.

By automating the allocation of classrooms and teachers greatly reduces the time, the same task was previously carried out on average five to six days, while the genetic algorithm are already getting results compared with those of the expert before the first 5 minutes on average.

INTRODUCCIÓN

La programación académica horaria que se realiza en las escuelas profesionales de las facultades de una Universidad es una actividad compleja y que requiere de un tiempo de entre 5 a 6 días para la persona encarga de esta actividad

Mediante la aplicación de la técnica de algoritmos genéticos es posible optimar el proceso de programación académica y de esta forma mejorar el uso de aulas y la asignación de docentes al dictado de los diferentes cursos programados.

Los Algoritmos Genéticos son una metodología de La Inteligencia Artificial que ha sido muy utilizada para resolver diferentes problemas de optimización. El presente trabajo utiliza un algoritmo genético para afrontar el problema de la generación de programas académicos por lo que es necesario ahondar en este tema.

La posibilidad de que los AGs resuelvan un amplio rango de problemas se debe principalmente a que la solución está representada como una población.

Es así que los AGs pueden ser aplicados a muchos problemas y esto solo está limitado a la habilidad del desarrollador para representar eficientemente la solución

.

Se dice que los AGS son una colección variada de algoritmos que pueden ser utilizados para evolucionar soluciones en un amplio rango de problemas debido a que hay una gran variedad de AGs que pueden ser utilizados y esto va depender del tipo de problema a ser resuelto.

La presente aplicación se desarrolla en la Escuela Profesional de Ingeniería Industrial de la Facultad de Ingeniería Industrial y de Sistemas de la Universidad Nacional del Callao.

El Capítulo I, trata sobre el planteamiento del problema a investigar, los objetivos, justificación, alcances y limitaciones de la investigación. En el Capítulo II sobre el marco teórico se describen las principales teorías y conceptos sobre algoritmos genéticos. El Capítulo III, trata sobre las variables de estudio y su operacionalización. En el Capítulo IV sobre metodología se expone el planteamiento metodológico aplicado, así como los medios de software y hardware requeridos para la aplicación. El Capítulo V trata sobre los resultados de la investigación, donde se desarrollan los principales algoritmos genéticos para asignación de aulas y docentes a las programaciones académicas. El Capítulo VI, trata sobre la discusión de los resultados de los experimentos sobre asignación de docentes y aulas y finalmente el Capítulo VII, se dan las principales conclusiones y recomendaciones de la presente investigación.

CAPÍTULO I. PLANTEAMIENTO DE LA INVESTIGACIÓN

1.- PLANTEAMIENTO DEL PROBLEMA DE INVESTIGACION

1.1. Determinación del problema

El sistema de la Universidad Peruana, actualmente lo conforma 142 universidades, de las cuales 91 son universidades privadas y 51 son universidades nacionales, ellas en conjunto cuentan con alrededor de 1136 facultades y 2272 escuelas profesionales, dedicadas a la formación profesional de sus estudiantes.

La Facultad, según la Ley Universitaria N° 30220 son las unidades de formación académica, profesional y de gestión. Están integradas por docentes y estudiantes.

La Escuela Profesional, según la Ley Universitaria N° 30220 es la organización encargada del diseño y actualización curricular de una carrera profesional, así como dirigir su aplicación, para la formación y capacitación pertinente hasta la obtención del grado académico y título profesional correspondiente.

Uno de los procesos más importantes de la Escuela Profesional, es la programación académica, mediante el cual estructura los cursos a dictarse en el semestre académico, los docentes a cargo del dictado de estas asignaturas, así como las aulas y horarios de los cursos programados.

El presente trabajo de investigación se desarrolla en la Universidad Nacional del Callao, en la Facultad de Ingeniería Industrial y de Sistemas la cual tiene dos

escuelas profesionales la Escuela Profesional de Ingeniería Industrial y la Escuela Profesional de Ingeniería de Sistemas , la facultad cuenta actualmente con 55 docentes ordinarios, 20 docentes contratados y alrededor de 1600 estudiantes, el mismo que incluye también a los estudiantes de la Sede Cañete, donde también se imparte enseñanza en las carreras profesionales de la facultad. La aplicación del algoritmo genético se centra en optimizar la programación horaria y académica en una Escuela Profesional de una Facultad Universitaria con sede principal en el Callao.

La programación académica que se desarrolla en una escuela profesional de una facultad universitaria, implica asignar a los distintos cursos programados, docentes, aulas y horario. Como consecuencia de ello sin programación académica, no se puede efectuar el proceso de matrícula, no se puede asignar docentes, aulas y horarios, por tanto no hay alumnos ni clases.

La programación académica para su organización tiene en cuenta como actores importantes a los cursos, docentes, y aulas, para poder satisfacer de manera adecuada las necesidades de formación de los estudiantes. En el caso de la variable curso se debe de considerar la demanda de alumnos por el curso, para el caso docentes se debe de considerar su disponibilidad horaria, su categoría y experiencia académica y profesional y para las aulas su capacidad de albergue de alumnos y disponibilidad horaria.

La programación de cursos a dictarse en un semestre académico es aprobada por el Director de Escuela y luego por el Consejo de Facultad, luego esta programación de cursos permite desarrollar la programación académica semestral, la misma que es desarrollada por un docente de la Escuela Profesional, en coordinación con la Dirección de Escuela y Departamento Académico, finalmente esta programación académica es aprobada por el Consejo de Facultad.

La programación académica busca optimizar la demanda de alumnos por un determinado curso respetando la disponibilidad horaria de los docentes, evitar los cruces de horario y la capacidad de albergue de estudiantes por aula.

1.2. Formulación del problema

En línea frente a estos problemas, pretendemos averiguar: ¿Es posible desarrollar un sistema informático que use Algoritmos Genéticos para optimizar la asignación de aulas y docentes en la programación académica en la Escuela Profesional de una Facultad Universitaria?

1.3. Objetivos de la investigación

1.3.1. Objetivo General

Desarrollar un Sistema Inteligente basado en Algoritmos Genéticos que permita mejorar y automatizar la asignación de aulas y docentes en la programación académica de la Escuelas Profesional de Ingeniería Industrial de la Facultad.

1.3.2. Objetivos Específicos.

Se desprende del objetivo general, los siguientes objetivos específicos:

- Obtener la mayor información de los expertos humanos encargados de la tarea de creación de programas académicos para poder determinar las variables que intervienen en dicha programación así como de sus respectivas restricciones.
- Diseñar y desarrollar un algoritmo genético ad hoc para mejorar y automatizar la asignación de aulas y docentes en programas académicos, teniendo como base las restricciones propias del problema.
- Evaluar los resultados obtenidos por el Sistema Inteligente basado en Algoritmos Genéticos comparándolos con los obtenidos por los expertos humanos, esperándose un grado de error en el rango de más menos 20 % de lo obtenido por los expertos humanos.

1.4. Justificación

Como se mencionó anteriormente la creación de programas académicos es un proceso fundamental en toda institución educativa, no solo desde el punto de vista académico, en la cual se trata de dar el mejor producto académico a los alumnos, sino también desde el punto de vista económico en el cual se trata de desarrollar un producto de calidad de la manera más eficiente posible.

La implementación de un Sistema Inteligente basado en Algoritmos Genéticos, desde el punto de vista académico, ayudará a desarrollar un mejor producto académico para los alumnos. Un mejor producto académico para los alumnos consiste en ofrecerles una adecuada infraestructura física y un servicio de enseñanza de calidad. Lo primero implica tener aulas apropiadas para el dictado de los cursos y cuya capacidad satisfaga al número de inscritos en ellos. Un servicio de enseñanza de calidad debe considerar varias aspectos, entre otros, que la persona encargada de dictar un curso sea la más apropiada de acuerdo a su nivel de experiencia y que haya menos alumnos insatisfechos porque no lograron matricularse en un curso debido a que no hay la suficiente cantidad de cursos aperturados. Es decir el desarrollo de un mejor producto académico para los alumnos está estrechamente relacionado a la creación de programas académicos.

Un Sistema Inteligente que automatice la creación de programas académicos ayudará a minimizar los errores que pueden ocasionarse al desarrollarse esta tarea. Toda esta complejidad resultante de la combinación de variables y de sus respectivas restricciones, presentes en este problema no es tan simple de resolver por un experto humano, ni por un software convencional.

Desde el punto de vista económico un Sistema Inteligente basado en Algoritmos Genéticos automatizará en buena medida la tarea de creación de programas académicos, con menos inversión en tiempo y en recursos humanos. Dichos recursos humanos tomarán como base la solución proporcionada por el Sistema

Inteligente a fin de complementarlo obteniendo así la versión definitiva de la programación académica.

Resolver el problema de la creación de programas académicos mediante el uso de Sistemas Inteligente también se justifica debido a que posteriormente éste pueda ser adaptado a otras instituciones educativas no solo especializadas en la enseñanza Superior Universitaria, sino también en Escuelas de Enseñanza, Institutos, etc.

1.5. Alcances y limitaciones

Entre los alcances de la presente tesis tenemos las siguientes:

- Un análisis y levantamiento de información de los expertos humanos relacionados a la creación de programas académicos.
- En base a la información recolectada de los expertos humanos realizar el diseño y programación del Sistema Inteligente basado en Algoritmos Genéticos.
- Evaluar los resultados obtenidos con el Sistema Inteligente basado en Algoritmos Genéticos de acuerdo a los factores de tiempo y errores en la creación de los programas académicos.

Entre las limitaciones de la presente tesis tenemos:

- El sistema no tendrá una interfaz de manejo de usuarios, roles y accesos, debido a que este sistema sólo será manejado en la Intranet de la Escuela Profesional Universitaria y los accesos serán restringidos a través de esta intranet.
- El Sistema Inteligente basado en Algoritmos Genéticos será ajustado a la realidad de la organización de estudio.
- El Sistema no cubre la proyección de cursos a dictar y toma esta información como entrada para la asignación óptima de docentes y aulas.
- El algoritmo genético presentado en este trabajo de investigación está sujeto a las restricciones más importantes recogidas por los usuarios por lo que está sujeto a mantenimientos por si se diera el caso de manejar restricciones adicionales o quitar algunas de ellas.

CAPÍTULO II. MARCO TEÓRICO

2.1. Antecedentes del estudio

Para entender el presente trabajo en este apartado haremos algunas definiciones concernientes a los Algoritmos Genéticos y de cada uno de sus componentes y parámetros. Así mismo describimos la tarea de creación de programas académicos dentro de ella.

Timetabling y Schedulin

En la asignación de horarios académicos, existen dos conceptos claves en la literatura que agrupan las investigaciones de este tipo de problemas. Los conceptos de timetabling y scheduling.

Un problema de scheduling es aquel que se trata de la asignación de recursos en el tiempo para llevar a cabo un conjunto de tareas (Baker 1974) y tiene como objetivo minimizar el costo total de los recursos asignados (Wren 1996)

Se define un timetabling el problema de asignar ciertos recursos, sujeto a restricciones, en un número limitado de horarios y lugares físicos con el objeto de satisfacer una serie de objetivos en el mayor grado posible (Wren 1996).

Los problemas de timetabling contienen restricciones fuertes y restricciones débiles (Larrosa 2003). Las restricciones fuertes o duras son aquellas que deben ser satisfechas de forma mandatorio y su no cumplimiento inhabilita la solución. Las restricciones débiles o suaves son aquellas que es deseable que se cumplan, pero que su incumplimiento no inhabilita la solución aunque la hace de menor calidad. Generalmente son restricciones de preferencia o de priorización, y muchas veces son estas restricciones las que se desean maximizar (o minimizar según sea el caso) para buscar acercarse a la solución óptima.

El objetivo del timetabling es minimizar el incumplimiento de las restricciones suaves (Tallabó 1999)

Generalmente, los problemas de asignación de horarios académicos corresponden a problemas de programación de clases y profesores y programación de clases y salas. Por ende, la mayoría de los problemas de asignación de horarios tienen componentes de timetabling y de scheduling.

Los Algoritmos Genéticos

Los Algoritmos Genéticos son una metodología de La Inteligencia Artificial que ha sido muy utilizada para resolver diferentes problemas de optimización. El presente trabajo utiliza un algoritmo genético para afrontar el problema de la generación de programas académicos por lo que es necesario ahondar en este tema. En primer lugar daré algunas definiciones básicas de un Algoritmo

Genético, y posteriormente pasaré a describir su flujo de funcionamiento y cada uno de sus componentes.

Conceptos Básicos

Los Algoritmos Genéticos (AGs) forman parte de lo que se llama los algoritmos evolutivos, los cuales están representados por tres escuelas fundamentales: Los algoritmos genéticos de J. H. Holland en USA, las estrategias evolutivas desarrolladas en Alemania por I. Rechenberg y H. P. Schwefel y la programación evolutiva. Cada una de ellas son propuestas diferentes pero todas ellas está inspiradas en la evolución natural [Pohlheim06].

Según la "Teoría de Evolución Natural" de Darwin, las especies naturales van evolucionando para adaptarse al medio en que viven. Aquellos individuos que mejor se adapten tendrán mayor probabilidad de procrear y sobrevivir hasta la edad adulta, haciendo así que sus características genéticas pasen de generación en generación [Villa07].

Los AGs no son un simple algoritmo, pero si una colección de algoritmos y técnicas que pueden ser utilizados para resolver una gran variedad de problemas en diferentes dominios. Por ejemplo, se puede considerar los AGs como una técnica para la optimización numérica, así como también pueden ser utilizados para muchas más aplicaciones [Tim08].

La posibilidad de que los AGs resuelvan un amplio rango de problemas se debe principalmente a que la solución está representada como una población. Es así que los AGs pueden ser aplicados a muchos problemas y esto solo está limitado a la habilidad del desarrollador para representar eficientemente la solución [Tim08]. Se dice que los AGS son una colección variada de algoritmos que pueden ser utilizados para evolucionar soluciones en un amplio rango de problemas debido a que hay una gran variedad de AGs que pueden ser utilizados y esto va depender del tipo de problema a ser resuelto[Tim08].

Se dice también que un AG es una técnica basada en poblaciones porque en lugar de operar en una simple solución potencial, los AGs utilizan una población de soluciones potenciales. Cuanto más grande sea la población, más grande va ser la diversidad de miembros de la población, y más grande el área de búsqueda de la población [Tim08].

Flujo Básico de un Algoritmo Genético

Ahora que ya conocemos más acerca lo que es un Algoritmo Genético, es necesario también saber su mecanismo de funcionamiento, En [Villa07] encontramos el flujo de un algoritmo genético, el cual está compuesto de la siguiente manera como se muestra en la Figura 2.1.

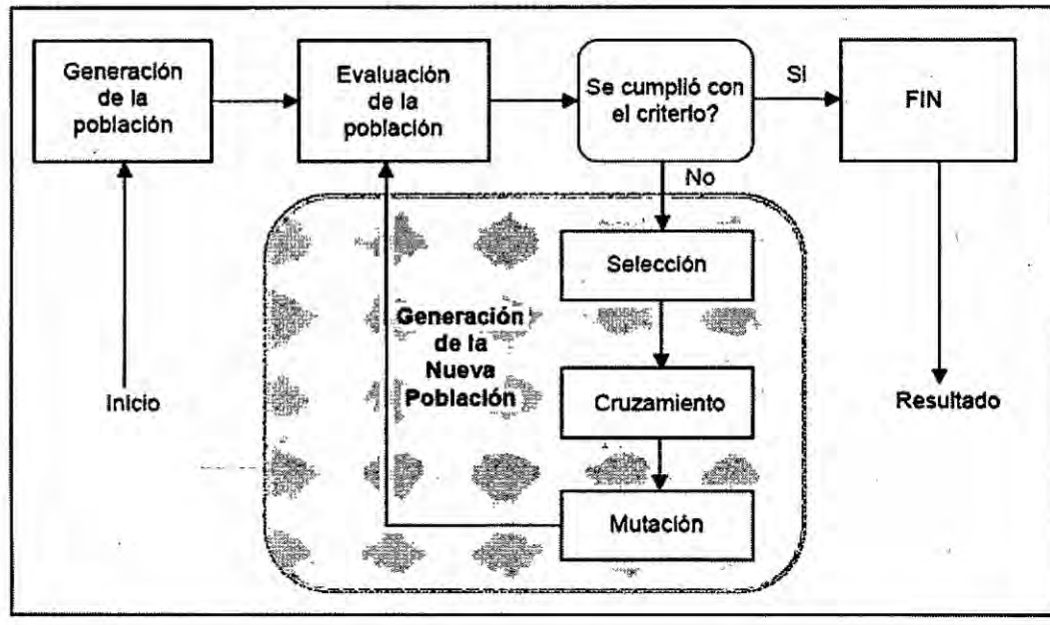

Figura 2.1 Flujo Básico de un Algoritmo Genético

Generación de la Población Inicial

En primer lugar se genera un conjunto de soluciones potenciales aleatoriamente. Este conjunto de soluciones sirve como la primera generación. Para conseguir mejores resultados, este conjunto de soluciones debería tener una adecuada diversidad (con miembros que sean diferentes y no similares) [Tim08].

Evaluación de la Población

Se calcula el fitness de cada individuo, el cual mide cuan bien cada solución potencial resuelve el problema. Cuanto más alto es el fitness, mejor solución se tendrá con respecto a las otras [Tim08].

De acuerdo con esta función de aptitud o fitness se evalúan los individuos y si cumplen o no con el criterio de parada. Si no cumple con el criterio de parada se procederá a generar una nueva población (una nueva generación de soluciones), de lo contrario el algoritmo se detendrá y arrojará un resultado. El criterio de parada se puede cumplir o bien por qué se cumplió con solucionar el problema es decir se obtuvo la solución con el mejor fitness, o porque se llegó a la última generación establecida como parámetro, o porque el usuario canceló el proceso [Pohlheim06].

Otro criterio de término es cuando hay una falta de diversidad en las soluciones que arroja el algoritmo, es decir que los miembros de la población se vuelven similares y por consiguiente causa una pérdida en la habilidad de buscar. Es entonces que necesitamos terminar el algoritmo detectando si el fitness promedio de la población está muy cerca al mejor fitness de cualquiera de los miembros de la población. Una vez que la población se hace muy similar y los miembros están en áreas similares del espacio de búsqueda el AG se vuelve incapaz de ir a otras áreas en las que se pueda buscar un mejor fitness [Tim08].

Generación de la Nueva Población

En la generación de la nueva población se siguen los procesos de selección, recombinación y mutación, a continuación describiremos cada uno de ellos.

Proceso de Selección

Este proceso determina que individuos son escogidos para ser recombinados. Estos individuos son los llamados padres de la siguiente generación de soluciones y son seleccionados de acuerdo a su fitness. Según [Tim08] son dos los algoritmos más básicos para la selección de padres: la selección tipo ruleta y la selección elitista, pero también encontramos otros algoritmos de selección como los que se señalan en [Pohlheim06] que son también importantes como es el caso de la selección por torneo. A continuación describiremos cada uno de los algoritmos de selección.

Selección Tipo Ruleta

La selección tipo ruleta es un algoritmo probabilístico que selecciona los miembros de la población en proporción a su fitness (el que se acomoda mejor, es el más probable de ser seleccionado).

Como podemos ver en la Figura 2.2 [Tim08] al utilizar la selección por ruleta, un resultado de selección probable seria que 2 miembros de A y un miembro de B y C sean seleccionados. Debido a que A es el miembro con más alto fitness que los otros miembros, este tiene el privilegio de propagar más cromosomas a la siguiente población.

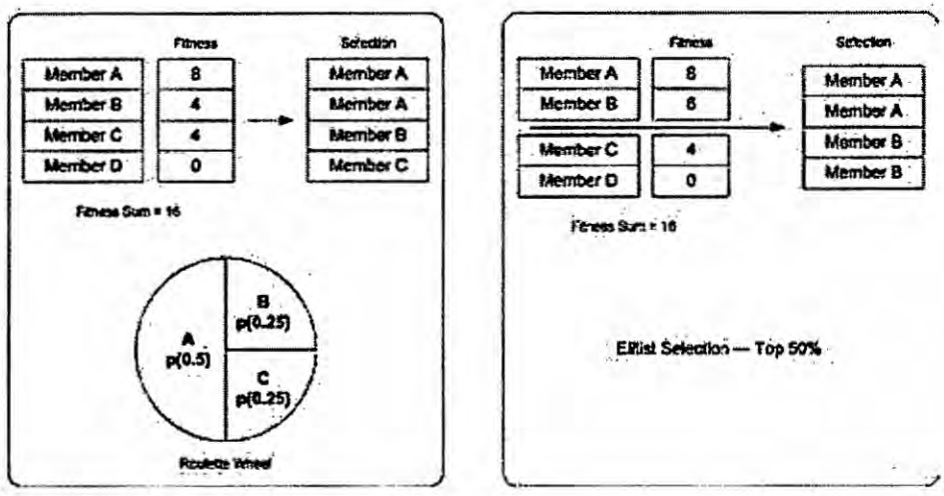

Figura 2.2 Selección Tipo Ruleta y Selección Elitista

Selección Elitista

En la selección elitista los miembros con mejor fitness son los seleccionados, forzando a que los demás miembros mueran.

En el caso de la Figura 2.2 el algoritmo elitista toma el 50% de miembros de la población con mejor fitness y luego distribuye a la siguiente generación.

Selección por Torneo

En esta selección un número t de individuos son seleccionados aleatoriamente de la población y el mejor individuo de este grupo es seleccionado como padre. Este proceso es repetido tantas veces como individuos son necesarios para completar la

población. Como podemos ver existe un parámetro t para manejar el tamaño del torneo, esta variable debe ser un número que varía entre 2 y el número de elementos de la población.

Proceso de Recombinación

A estas alturas tenemos un número de miembros que tiene el derecho de propagar su material genético a la siguiente población. El siguiente paso es recombinar este material genético para formar la siguiente generación. Comúnmente los padres son seleccionados dos a la vez del conjunto de individuos que están permitidos a propagarse (del proceso de selección). Dados los dos padres, dos hijos son creados en la siguiente generación con cierta cantidad de alteración producto del proceso de recombinación. Este proceso de recombinación se realiza mediante el uso de operadores genéticos, estos pueden ser: Crossover, Mutación e Inversión [Tim08].

En el operador Crossover como se observa en la Figura 2.3, los padres son combinados seleccionando un punto de cruce y luego se intercambian las colas de los padres para producir dos hijos. Otra variante del Crossover crea dos puntos de cruce, intercambiando los materiales genéticos en 2 lugares [Tim08].

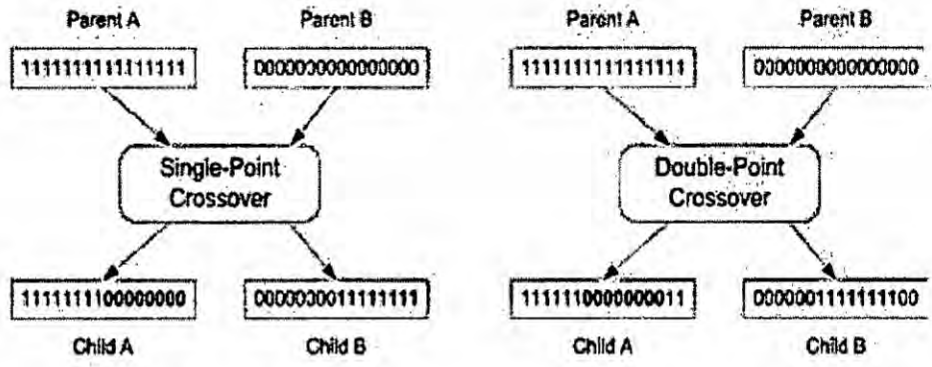

Figura 2.3 Operación Tipo Crossover

La operación de Mutación y la de Inversión son dos de las operaciones iniciales del trabajo original de Holland y podemos verlas en la Figura 2.4. El operador de mutación sólo muta o cambia un bit. Mientras que en el operador de inversión se toma un pedazo del cromosoma y se lo invierte. En este caso un rango de bits es cambiado [Tim08].

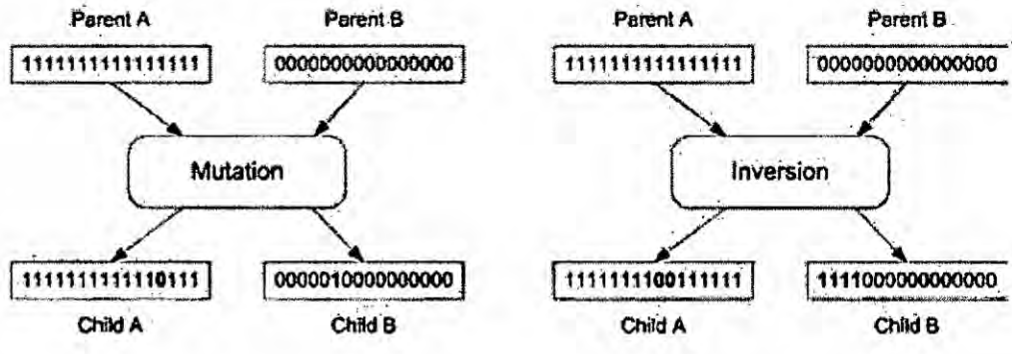

Figura 2.4 Operación de Mutación e Inversión

Parámetros Genéticos

Como hemos visto hasta el momento los AGs utilizan diferentes parámetros. Estos parámetros van a influenciar en el comportamiento del algoritmo genético y especialmente en los resultados obtenidos por éste. Es por eso que a continuación vamos a hacer una breve descripción de cada uno de ellos.

Tamaño de la Población

Este parámetro va a medir el número de individuos que habrá en la población. Según [Villa07] El tamaño de la población afecta el desempeño global y la eficiencia de los AGs; de esta manera con una población pequeña el desempeño puede caer debido a que la población ofrece una pequeña cobertura del espacio de búsqueda del problema. Una población grande generalmente ofrece una cobertura representativa del dominio del problema, además de prevenir convergencias prematuras para soluciones locales en vez de globales. Asimismo cuando se trabaja con grandes poblaciones, los requisitos computacionales son mayores y puede tener una convergencia del resultado más lenta

Tamaño del Torneo

Este parámetro mide el tamaño de la competencia en el algoritmo de selección por torneo para seleccionar los padres de la siguiente generación. Según [Pohlheim06] cuando el tamaño del torneo es mayor hay una mayor intensidad de selección y también mayor pérdida de la diversidad de la población

Tasa de Mutación

La tasa de mutación nos va medir el número de intercambios que van a realizarse en el proceso de mutación. En [Villa07] se señala que una baja tasa de mutación previene que una posición del cromosoma se quede estancada en un valor, además de posibilitar que se llegue en cualquier punto del espacio de búsqueda. Con una tasa muy alta la búsqueda se torna esencialmente aleatoria.

Número de Generaciones

Este parámetro nos indica el número de iteraciones que hará el algoritmo genético hasta detenerse. Cuanto mayor sea éste número de generaciones mayor va a ser el tiempo de búsqueda del algoritmo y puede ser que también se obtengan mejores resultados.

La Creación de Programas Académicos en Escuela Profesional Universitaria

El núcleo del negocio del EPU es ofrecer el servicio de enseñanza de formación en Ingeniería Industrial e Ingeniería de Sistemas a sus alumnos. Los programas de enseñanza está compuesto por un conjunto de cursos que tienen que ser llevados por los alumnos durante el semestre académico, el semestre académico tiene una duración de diecisiete semanas.

El Director de Escuela en coordinación con el Docente Programador Académico, elaboran la Programación Académica de acuerdo a distintos factores.

En primer lugar se hace una proyección de los cursos que van a dictarse durante el semestre académico, para ello se toma en cuenta la matrícula semestral anterior y la matrícula anterior a esta, tomando en consideración que la programación académica es semestral y los cursos varían de semestre en semestre. Se programa desde el primer ciclo hasta el décimo ciclo, teniendo en consideración que la Universidad Nacional del Callao efectúa dos procesos de admisión al año, uno en diciembre cuyos ingresantes comienzan sus clases generalmente a fines de marzo o primeros días de abril y el otro examen de admisión es el mes de julio, cuyos ingresantes comienzan sus clases en el mes de agosto. Partiendo de esta consideración se tiene alumnos casi a plena capacidad para los distintos ciclos de estudio.

2.2. Aportes teóricos

MARTÍN DEL BRIO, Bonifacio; SANZ, Alfredo. "Redes Neuronales y Sistemas Borroso". Editorial Rama. España 2006

Los sistemas borrosos de control FLC, su campo aplicación más importante es a nivel industrial, lejos de ser entes estáticos, estos pueden ser entrenados para optimizar su funcionamiento. Los FLC son aproximaciones funcionales genéricos, es decir, dado un cierto nivel de error, se puede encontrar un FLC que aproxime cualquier función con un error menor que el fijado y para ello se puede hacer uso

de diversas técnicas, algunas procedentes del campo de las redes neuronales, otras de los métodos estadísticos, así como de los algoritmos genéticos.

Los algoritmos genéticos se inspiran en la forma como la naturaleza hace la evolución de los seres vivientes, adaptándolos a diferentes entornos o hábitats, en la aplicación del más fuerte, el más fuerte es el que sobrevive. La técnica empleada por la naturaleza consiste principalmente en la codificación de las características de los seres vivos en el genoma y su evolución a través de la reproducción sexual y las mutaciones. Desde esta perspectiva los algoritmos genéticos se presentan como una forma de optimización de sistemas.

En el marco de los algoritmos genéticos se define:

Genoma: Todos los parámetros que definen a todos y cada uno de los individuos de la población.

Genotipo: La parte del genoma que describe a un individuo concreto. En esta interpretación son los genes que describen un controlador concreto.

Fenotipo: La expresión de un genotipo. En esta interpretación es un controlador concreto.

Gen: Cada uno de los parámetros que describen a un individuo. En esta interpretación los genes codifican parámetros de un controlador concreto en este

caso de un FLC pueden ser por ejemplo, la ganancia del escalado, la forma o el tipo de función de pertenencia de una entrada, salida, regla, etc.

Qué optimizar?

Al respecto caben cuatro opciones básicas:

a) Optimización de coeficientes.

- Ganancias de entrada y salida.
- Partición de las variables de entrada.
- Partición de las variables de salida.

b) Optimización de reglas.

c) Optimización de la estructura de la base de reglas.

d) Optimización completa.

- Codificación desordenada.

Complejidad del problema

La complejidad computacional estudia los recursos que necesita un algoritmo para resolver un problema. Los recursos objeto de estudio son el tiempo y el espacio. El tiempo se traduce en la cantidad de pasos de ejecución de un algoritmo para resolver el problema y el espacio se puede definir como la cantidad de memoria necesaria. Desde ese punto de vista los problemas se clasifican de muchas clases, de ellos definiremos: P, NP y NP-Completo (Garey 1983).

La clase de complejidad P es el conjunto de problemas de decisión que pueden ser resueltos en una máquina de (de Turing) determinista en tiempo polinómico, lo que corresponde intuitivamente a problemas que pueden ser resueltos aún en el peor de los casos. La clase de complejidad NP es el conjunto de los problemas de decisión que pueden ser resueltos en el tiempo polinómico por una máquina de Turing no- determinista.

La clase NP es importante porque contiene muchos problemas de búsqueda y optimización para los que se desea saber si existe cierta solución o si existe una mejor solución de las conocidas. Todos los problemas de esta clase tienen la propiedad de que sin embargo, cualquier solución suya puede ser verificada.

La clase de complejidad NP-completo es el subconjunto de los problemas de decisión en NP tal que todo problema en NP-completo se puede transformar poligonalmente en cada uno de los problemas de NP-completo. Una

transformación polinomial es un algoritmo determinista que transforma instrucciones de un problema en instrucciones del otro.

Para el problema de time tabling, al ser un problema de complejidad NP-completo, se hace imposible intentar obtener la mejor solución mediante el descubrimiento de todas las posibles soluciones para los casos en que el tamaño del problema exceda una cierta dimensión, normalmente bastante limitada, por lo que deben ser abordado usando alguno de los siguientes enfoque (Cormen 2001):

- Aproximación: Un algoritmo que rápidamente encuentra una solución no necesariamente óptima, pero dentro de un cierto rango de error. En algunos casos, encontrar una buena aproximación es suficiente para resolver el problema, pero no todos los problemas NP-completos tienen buenos algoritmos de aproximación.
- Probabilidad: Un algoritmo probabilístico obtiene en promedio una buena solución al problema planteado, para una distribución de los datos de entrada dada.
- Heurística: Son algoritmos que no entregan soluciones exactas y que se basan en una estrategia de búsqueda para llegar más rápido a una solución.

El problema de la generación de horarios académicos es un problema tipo NP-completo, lo cual lo convierte en objeto de estudio por su dificultad y sus múltiples variantes.

El problema de creación de horarios en su temática general es universal a todas las instituciones de educación superior, incluyendo en ellas a las universidades, pero las restricciones particulares hacen que cada instancia requiera una solución propia.

Soluciones Meta-heurísticas

Las metodologías meta-heurísticas son métodos probabilísticos que parten de una solución inicial y a medida que el algoritmo se va ejecutando va mejorando la solución. Este tipo de soluciones están basadas en un principio de "aumento sucesivo" e intentan evitar la generación se soluciones pobres de calidad. Una de las desventajas que presentan es el alto costo de procesamiento que requieren para llegar a una solución de calidad.

Dentro de estos métodos meta-heurísticos tenemos:

- Búsqueda tabú (Tabu Search)
- Algoritmos genéticos.
- Colonia de hormigas (ACO Ant Colony Optimización)

- Recocido simulado o enfriamiento lento simulado (simulated annealing)

RODRÍGUEZ, Piedad. "Introducción a los algoritmos genéticos y sus aplicaciones"

El origen de lo que se conoce como computación evolutiva hay que buscarlo en su razón de ser, los conocimientos sobre evolución se pueden aplicar en la resolución de problemas de optimización. La idea era "evolucionar" una población de candidatos a ser solución. El operador de selección escoge, entre los cromosomas de la población aquellos con capacidad de reproducción y entre estos, los que sean más "compatibles", producirán más descendientes que el resto. El cruce extrae partes de dos cromosomas, imitando la combinación biológica de dos cromosomas aislados (gemelos). La mutación se encarga de cambiar de modo aleatorio, los valores del alelo en algunas localizaciones del cromosoma y, por último, la inversión, invierte el orden de una sección contigua del cromosoma, recolocando por tanto el orden en que se almacenan los genes.

En opinión de John Holland y sus colegas de la Universidad de Michigan los objetivos de los algoritmos genéticos son: (1) abstraer y explicar rigurosamente el proceso adaptativo de los sistemas naturales, y (2) diseñar sistemas artificiales que retuvieran los mecanismos más importantes de los sistemas naturales.

Los algoritmos genéticos, son algoritmos de búsqueda basados en los mecanismos de selección natural y genética natural. Combinan la supervivencia de los más

compatibles entre las estructuras de cadenas, con una estructura de información ya aleatorizada, intercambiada para construir un algoritmo de búsqueda con algunas de las capacidades de innovación de la búsqueda humana.

Todos los organismos que conocemos están compuestos por una o más células, cada una de las cuales contiene a su vez uno o más cromosomas (esto es cadenas de ADN), que tienen la función de ser una especie de "anteproyecto" del organismo que forman parte. Un cromosoma se puede dividir conceptualmente en genes, bloques funcionales de ADN que codifican una determinada proteína. Solemos pensar en los genes, aunque en una visión muy superficial, como los responsables de determinar los rasgos del individuo, tales como color de los ojos, cabellos, piel. Las diferentes posibilidades de escoger un rasgo (ojos azules, cabello negro, piel trigueña) reciben el nombre de alelos. Cada gene está localizado en una determinada posición (lugar) dentro del cromosoma que integra.

CAPÍTULO III. VARIABLES DE ESTUDIO

3.1.- VARIABLES.

3.1.1.- Variable Dependiente.

Programación Académica en la Escuela Profesional de una Facultad Universitaria.

3.1.2.- Variables Independientes.

- Aplicación de Algoritmos Genéticos (factores claves a modelar aulas, docentes, costos, tiempo).

3.2.- Definición y operacionalización de variables

Figura 3.1 Definición y operaciones con variables

Variables	Definición Conceptual	Definición Operacional	Indicadores	Medición
Variable Independiente				
Algoritmos genéticos	Son algoritmos de búsqueda basados en los mecanismos de selección natural y genética natural. Combinan la supervivencia de los más compatibles entre las estructuras de cadenas, con una estructura de información ya aleatorizada, intercambiada para construir un algoritmo de búsqueda con algunas capacidades de innovación de la búsqueda humana. La idea del procedimiento algorítmico es la supervivencia del más más apto	El algoritmo genético a través de los operadores de selección, cruce y mutación, en primera instancias escoge los cromosomas entre la población para efectuar la reproducción, luego elige el lugar y cambia las secuencias antes y después de esa posición entre dos cromosomas para crear una nueva descendencia, para finalmente producir variaciones de modo aleatorio en el cromosoma. .	Indicador de aulas utilizadas, Indicador de docentes. Indicador de tiempos. Indicador de costos.	Número aulas utilizadas. Número docentes. Tiempos. Soles.
Variable Dependiente				
Programación académica en la escuela profesional de la facultad.	Sistema de planificación y coordinación de actividades humanas que permite conseguir los objetivos de instituciones de manera efectiva.	El Sistema facilita la toma de decisiones a partir de las interrelaciones sistémicas de sus distintos componentes.	Cumplimiento de la programación académica. Sistema facilita toma de decisiones a partir uso de algoritmos genéticos.	%Cumplimiento de la programación académica. Facilidad sistema toma decisiones(rapidez, eficacia, exactitud)

CAPÍTULO IV. METODOLOGÍA

4.1. Tipo de investigación

Es un tipo de investigación descriptivo, correlacional.

4.2. Diseño de la investigación

El diseño corresponde a una investigación de tipo mixto, utilizando el enfoque cualitativo y cuantitativo para el acopio y tratamiento de los datos.

4.3. Población y muestra

Población y muestra se considera la Programación Académica que efectúa la Escuela Profesional de Ingeniería Industrial e Ingeniería de Sistemas de la Universidad Nacional del Callao

4.4. Técnicas e instrumentos de recolección de datos

Para la recolección de información se utilizará:

- Cuestionario.
- Revisión documental.
- Visitas de campo.

Los instrumentos que se utilizarán son:

- Cuestionarios
- Guías.

4.5. Plan de análisis estadístico de datos

Técnicas Estadísticas y Descriptivas a utilizar:

Prueba de hipótesis.

Se usarán índices promedios ponderados y de desviación de las distintas variables.

Se usará AGs para el análisis de la situación problema y propuesta de solución.

CAPÍTULO V. RESULTADOS UTILIZANDO ALGORITMOS GENÉTICOS

En el presente capítulo vamos a presentar la solución propuesta para resolver el problema de la asignación de aulas y docentes en la generación de programas académicos mediante el uso de algoritmos genéticos. Para ello en primer lugar justificaremos el uso de los algoritmos genéticos para resolver este problema y posteriormente describiremos la adaptación de la metodología en la resolución del problema.

5.1. Uso de los Algoritmos Genéticos.

En primer lugar nombraremos las técnicas actuales existentes para resolver el problema de la programación académica, luego señalaremos los criterios que nos servirán para evaluar las técnicas mencionadas y finalmente presentaremos un cuadro comparativo en el cual justificaremos porque el uso de los algoritmos genéticos es el más apropiado en la solución de este problema.

5.1.1. Técnicas Utilizadas

Entre las técnicas mencionadas para la optimización en la asignación de aulas y docentes vemos que la única comparable con los algoritmos genéticos es la de la Búsqueda Tabú. Descartamos la primera solución debido a que es un sistema determinista que sólo sirve para asistir en la creación de programas académicos. Descartamos así mismo el modelo basado en agentes porque solo nos proporciona

un diseño pero no nos presenta en detalle el algoritmo a utilizarse en cada uno de los agentes componentes del diseño, este algoritmo podría ser un algoritmo de optimización o no.

5.1.2. Criterios de Evaluación

Entre los criterios de evaluación podemos señalar los expuestos en [Raghavjee08]. Aquí se comparan el uso de los algoritmos genéticos y la búsqueda Tabú para el problema de la Programación de Clases en base al mejor costo y al costo promedio obtenido por cada metodología.

Es preciso también señalar que el problema de la programación afrontado en este caso de estudio está sujeto a la realidad correspondiente, pero de todos modos nos sirve como referencia para poder optar por una metodología en lugar de la otra.

5.1.3. Comparación de las Técnicas

A continuación mostraremos los resultados obtenidos en el trabajo desarrollado en [Raghavjee+2008], en el cual se comparan el uso de los algoritmos genéticos con la Búsqueda Tabú en base al costo promedio final y el mejor costo obtenido por cada uno de los algoritmos de optimización sobre un conjunto de cuatro problemas presentados con diferentes números de restricciones.

Estos requerimientos están dados por el número de clases, número de profesores, número de localizaciones (venues) o aulas y número de periodos, como se puede observar la siguiente Figura 5.1.

Problem	No. of Classes	No. of Teachers	No. of Venues	No. of Periods	No. of Reqs
hdtt4	4	4	4	30	120
hdtt5	5	5	5	30	150
hdtt6	6	6	6	30	180
hdtt7	7	7	7	30	210
hdtt8	8	8	8	30	240

Figura 5.1 Características de los Problemas a ser Resueltos.

Posteriormente en la Figura 5.2 podemos observar los resultados obtenidos tanto por la Búsqueda Tabú (TS) como por los Algoritmos Genéticos (GA) para este conjunto de problemas.

Method	*hdtt4*	*hdtt5*	*hdtt6*	*hdtt7*	*hdtt8*
TS	BC: 0 AC: 0.2	BC: 0 AC: 2.2	BC: 3 AC: 5.6	BC: 4 AC: 10.9	BC: 13 AC: 17.2
GA	BC: 0 AC: 0	BC: 0 AC: 0	BC: 0 AC: 0.1	BC: 0 AC: 0.5	BC: 0 AC: 0.73

Figura 5.2 Cuadro Comparativo de los Resultados Obtenidos por la Búsqueda Tabú (TS) y los Algoritmos Genéticos (GA)

El valor BC representa el mejor costo y el valor AC el costo promedio en cada una de los problemas presentados. Como podemos notar los mejores costos son obtenidos por el Algoritmo Genético, justificando de esta forma la elección de esta metodología para resolver el problema de la Programación Académica en una escuela profesional de la UNAC.

5.2. Adaptación de los Algoritmos Genéticos en el Problema.

En este apartado vamos a presentar la adaptación del algoritmo genético para la solución del problema de la programación académica tomando como referencia la realidad de la UNAC, el algoritmo genético básicamente busca optimizar la asignación tanto de aulas como de docentes haciéndolo de manera automática, de esta forma el usuario final trabajará sobre él para hacerle los ajustes finales y lo habilitará para un proceso de matrícula.

A continuación empezaremos con el planteamiento del problema

5.2.1. Planteamiento Del Problema

El Problema está dividido en dos etapas:

a) Asignar el total de aulas disponibles de una escuela profesional de la UNAC a todos los grupos de cursos en cada horario que ésta ofrece para la enseñanza en un semestre determinado.

b) Asignar el total de profesores disponibles de una escuela profesional de la UNAC a todos los grupos de cursos en cada horario que ésta ofrece para la enseñanza en un semestre determinado.

Estas dos etapas pueden hacerse paralelamente o una después de otra en cualquier orden, además es necesario recalcar que para la realización de esta asignación se toma como precondición que los cursos ya estén proyectados para el semestre

indicado, es decir que ya deberían estar cargados todos los grupos de cursos con sus respectivos horarios en la Base de Datos listos para sólo actualizarlos con las aulas y docentes requeridos.

Cada asignación tanto de docentes como de aulas está sujeta a un conjunto de restricciones que deben cumplirse.

En el caso de la asignación de aulas se tienen las siguientes restricciones:

R1. Un aula sólo debe ser asignada a un curso si está disponible, de esta forma se debe evitar que un aula sea programada más de una vez en un mismo horario.

R2. El 75% de los cursos del primer ciclo proyectados en un horario y en una sede determinada deben ser asignados con las aulas que tienen la primera mejor capacidad, y el 25 % restante, con aulas que tienen la segunda mejor capacidad, Ejemplos:

Dado un conjunto de cursos del primer ciclo para el Horario de 8:00 a.m. a 9:40 a.m. en donde la primera y segunda mejor capacidad de las aulas es de 100 y 80 alumnos respectivamente, una asignación correcta sería la siguiente:

Grupo 1, Aula 406, Capacidad =100

Grupo 2, Aula 504, Capacidad = 100

Grupo 3, Aula 601, Capacidad = 100

Grupo 4, Aula 410, Capacidad = 80

Mientras que una incorrecta asignación sería la siguiente:

Grupo 1, Aula 201, Capacidad = 50

Grupo 2, Aula 301, Capacidad = 50

Grupo 3, Aula 405, Capacidad = 50

Grupo 4, Aula 410, Capacidad = 80

R3: Todos los demás cursos deben ser asignados en un aula de tal manera que el número de alumnos proyectados en cada curso sea menor o igual a la capacidad del aula, Ejemplo.

Dado el Grupo 1 del curso Matemática II en el horario de 8:00 a.m. a 9:40 a.m. que tiene una proyección de alumnos de 50, entonces un aula correctamente asignada sería la siguiente:

Aula Asignada: 306, Capacidad = 60

Mientras que un aula incorrectamente asignada sería la siguiente

Aula Asignada: 302, Capacidad = 40

R4: Los cursos que tienen horas de tipo laboratorio deben ser asignados en las aulas de tipo laboratorio en un determinado horario.

Y en el caso de la asignación de docentes se tiene las siguientes restricciones:

R1: Asignar a los profesores (principales, asociados y/o auxiliares) de tal manera que cumplan con todas las horas obligatorias según su dedicación:

Dedicación exclusiva: como mínimo 10 horas de clases

Tiempo completo: como mínimo 10 horas de clases

Tiempo parcial 20 horas: como mínimo 20 horas de clases.

Tiempo parcial 10 horas: como mínimo 10 horas de clases.

R2: Asignar a los profesores extraordinarios entre 10 y 13 horas.

R3: Asignar a los jefes de práctica sólo a horarios de práctica o laboratorio, no se les (debe asignar a horas de teoría).

R4: Un profesor solo puede dictar en otras horas que hayan sido indicadas en su disponibilidad horaria semestral.

R5: Se deben asignar profesores a cursos que estén habilitados para dictar (normalmente son los cursos en que ya ha dictado históricamente).

R6: No se puede programar a un mismo profesor en un horario determinado para más de un curso.

R7. No se puede programar más de un profesor en un mismo rango de horarios para un mismo curso y aula determinados.

5.2.2. Representación de La Solución

Como hemos podido ver en el planteamiento del problema hay dos subproblemas que tienen que ser resueltos, uno es la asignación de aulas y el otro es la asignación de docentes cada uno con sus respectivas restricciones. Debido a esto se ha visto por conveniente diseñar dos algoritmos genéticos uno para la asignación de docentes y otro para la asignación de aulas. A continuación se hará la descripción de la solución para lo cual empezaremos por identificar las entradas al problema, posteriormente se mostrará las salidas las cuales están representados por dos cromosomas una para representar la solución de la asignación de aulas y otra para la asignación de docentes. Luego mostraremos

como se define la función objetivo de cada uno de los algoritmos genéticos y su respectiva función fitness. Dado esto presentaremos el código de los algoritmos genéticos con cada uno de sus componentes como son el algoritmo de selección de los padres de la nueva generación y el algoritmo de mutación.

5.2.2.1. Entradas.

A continuación de acuerdo a los datos proporcionados en el planteamiento del problema vamos a definir las entradas que tomará el Algoritmo Genético tanto para la asignación de aulas como para la asignación de docentes:

Semestre: semestre actual de la programación académica.

Año: año actual de la programación académica.

Escuela: escuela donde se realiza la programación académica.

Hs: Arreglo de horarios ofrecidos, donde:

$$Hi \in Hs = \{H_0, H_1, \ldots, H_h\},$$

h: número de horarios ofrecidos.

Aus: Arreglo de aulas disponibles, donde:

$$Ai \in Aus = \{A_0, A_1, \ldots, A_a\},$$

a: número de aulas disponibles.

Profes: Arreglo de profesores disponibles, donde:

$$Pi \in Profes = \{P_0, P_1, \ldots, P_p\},$$

p: número de profesores disponibles.

Programas: Arreglo de Programas Académicos proyectados, donde:

$$PAi \in Programas = \{PA_0, PA_1, \ldots, PA_{pa}\},$$

pa: número de programas académicos proyectados el semestre y año señalado

Cada uno de los elementos de horarios, aulas, profesores y programas académicos proyectados, etc. será representado a través de clases UML, las cuales nos van a permitir acceder con mayor facilidad a los atributos de cada una de éstas clases UML. Esto se puede apreciar en la Figura 5.3 que nos muestra el Diagrama de Clases UML.

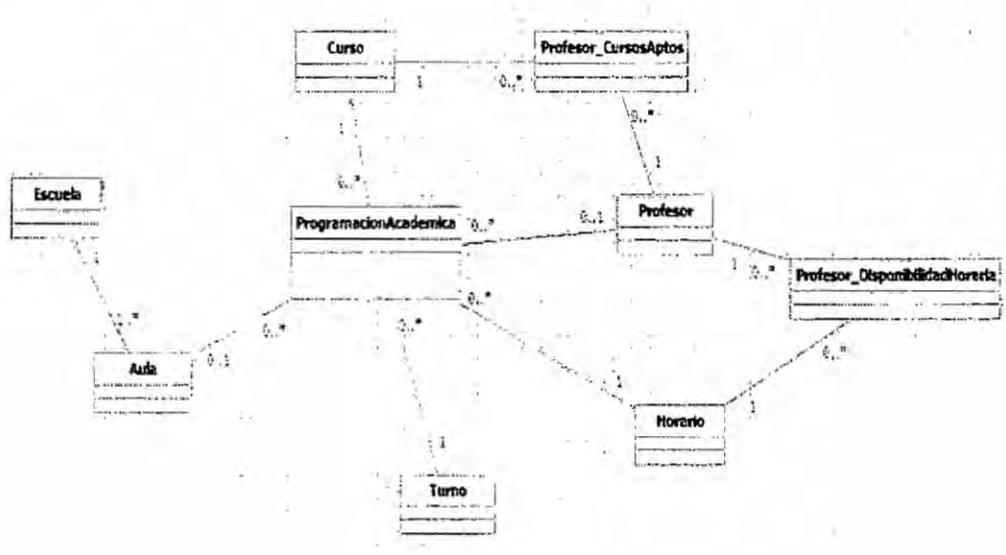

Figura 5.3 Diagrama de Clases del Sistema

Como se puede apreciar en la Figura 5.3 existen otras clases adicionales como los Curso, Escuela, Profesor_DisponibilidadHoraria, Turno, Profesor_cursosAptos que no se han considerado dentro de las entradas, estos por facilidad de diseño se encuentran como atributos dentro de otras clases debido a la relación de dependencia entre ellas. De esta forma las clases UML cursos están dentro de las

clases UML Programas Académicos y las clases UML Profesor_DisponibilidadHoraria, Profesor_cursosAptos están dentro de la clase UML Profesor. Por lo que implícitamente ya son entradas del Sistema.

Otras de las entradas para la solución del problema mediante el uso de Algoritmos Genéticos son los parámetros genéticos, para éste problema se utilizarán los siguientes:

TAMAÑO_POBLACION: Representa con cuantas soluciones (cromosomas o individuos) trabajará el algoritmo genético en cada generación (iteración)

NUMERO_GENERACIONES: Representa el número de iteraciones que dará el algoritmo genético.

TAMAÑO_TORNEO: Representa la cantidad de soluciones que se seleccionarán aleatoriamente para competir por reproducir su material genético en el proceso de mutación.

TAMAÑO_MUTACION: Representa el número de intercambios que se hará a una solución durante el proceso de mutación.

5.2.2.2. Salidas.

Las salidas van a depender de cada etapa del problema a solucionar, es así que se va a tener un cromosoma que represente la solución al problema de la asignación de aulas y otro cromosoma que representa al problema de la asignación de docentes de la siguiente forma:

Q: Matriz Solución de Programas Académicos para la Asignación de Aulas, donde:

$$Qij \in Q = \{Q_{00}, Q_{01}, \ldots, Q_{ha}\}$$

a: número de aulas disponibles y

h: número de horarios disponibles.

R: Matriz Solución de Programas Académicos para la Asignación de Profesores, donde:

$$Rij \in R = \{R_{00}, R_{01}, \ldots, R_{hp}\},$$

p: número de profesores disponibles y

h: número de horarios disponibles.

Como podemos ver cada cromosoma es una matriz de programas académicos. Tanto en el primero como en el segundo caso cada fila representa un Horario, mientras que las columnas representan las aulas y los docentes respectivamente para cada caso. Una de las ventajas de esta representación es que nos ayuda a poder evitar los cruces de horarios tanto para las aulas, como para los profesores. Finalmente lo que está en la intersección entre las filas y columnas son los programas académicos proyectados inicialmente, es decir en cada uno de ellos hay información sobre el curso y el horario en el que han sido proyectados y cuando el valor de esta intersección sea "Nothing" es porque no hay ningún programa académico para el aula o profesor y horario determinado. Lo que hará el algoritmo genético es hacer movimientos entre estos programas académicos entre columnas solamente y no entre filas ya que no se pretende alterar los horarios a los que pertenecen cada uno de los programas académicos sino intercambiar aulas y profesores hasta tratar de ubicarlos en el aula y docente más apropiados de

acuerdo a las restricciones establecidas. A continuación en la Figura 5.4 y Figura 5.5 vemos gráficamente la representación de cada uno de los cromosomas:

	A_1	...	A_a
H_1	PA_1		PA_5
. . .			
H_h	Nothing		PA_{10}

Figura 5.4 Representación Cromosómica de la Asignación de Aulas

	P_1	...	P_p
H_1	PA_5		Nothing
. . .			
H_h	PA_{10}		PA_{11}

Figura 5.5 Representación Cromosómica de la Asignación de Docentes

Otra de las salidas del algoritmo genético es el costo final que alcanza a obtener el algoritmo genético, el número de la iteración o generación en que consiguió este costo, el tiempo que demoró en hacer toda esta operación y la variación del costo del Algoritmo Genético respecto a la asignación hecha por los encargados de esta tarea manualmente.

5.3. Descripción de la Solución Tecnológica

En este capítulo vamos a describir en detalle el sistema desarrollado para la resolución del problema de la asignación de docentes y aulas, para ello presentaremos el modelado del negocio, los diagramas de actividad, el diagramas de casos de uso y sus respectivas especificaciones, el diagrama de secuencia, el diagrama de clases, algunas consideraciones sobre el ambiente de desarrollo, requerimientos mínimos de hardware y software, el modelo de datos, los prototipos del sistema y finalmente la codificación de los algoritmos genéticos.

5.3.1. El Modelado del Negocio

Como podemos observar en La Figura 5.6 el modelo de negocio está compuesto por dos procesos la asignación de docentes y la asignación de aulas. El trabajador de negocio viene a ser la secretaria de la escuela profesional quien es la persona encargada de realizar esta tarea. Un actor de negocio vendría a ser el Sistema de Matrícula, el cual tomará la información cargada en los programas académicos para poder realizar los procesos de matrícula. Es preciso señalar de que existe también otro proceso que se realiza dentro del marco de la creación de programas académicos, este es la proyección de cursos a aperturar en un mes determinado, esto no ha sido considerado debido a que el algoritmo diseñado en este trabajo de investigación está orientado a resolver el problema de la asignación de aulas y docentes específicamente.

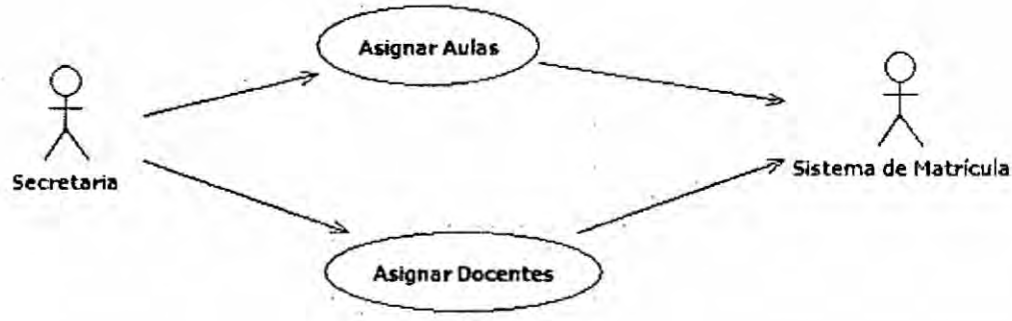

Figura 5.36 Modelo de Negocio

5.3.2. Diagramas de Actividad

A continuación en la Figura 5.7 y 5.8 presentamos los diagramas de actividad de cada uno de los casos de uso de negocio. Como podemos observar en cada uno de los procesos se empieza tomando la información a partir de los programas académicos proyectados. Cada programa académico proyectado contiene información acerca del horario y el curso que se dicta dentro de este horario. También se tiene la información de la demanda de alumnado en cada uno de los cursos. Por otro lado se cuenta con la información de las aulas disponibles, los docentes disponibles, las horas obligatorias en las que debe dictar un docente, la disponibilidad para dictar y los cursos aptos para dictar, todo esto representan las restricciones para hacer una asignación de docentes y de aulas de manera óptima. Finalmente toda esta información debe ser actualizada en el Sistema de Matrícula para que se puedan iniciar los procesos de matrícula.

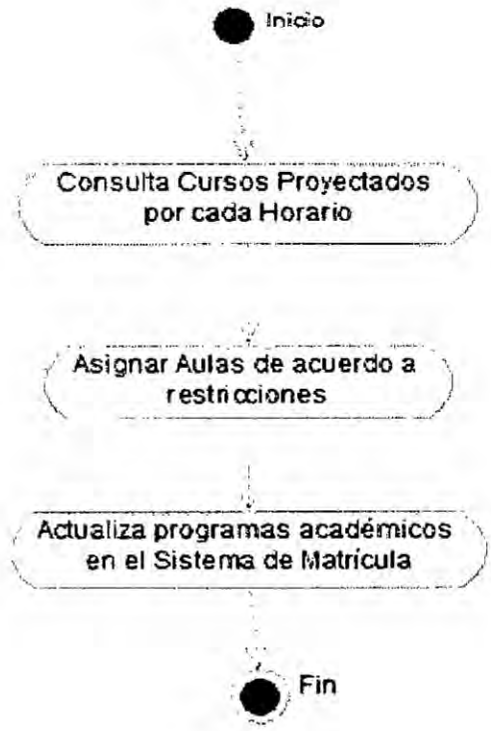

Figura 5.37 Diagrama de Actividad de la Asignación de aulas

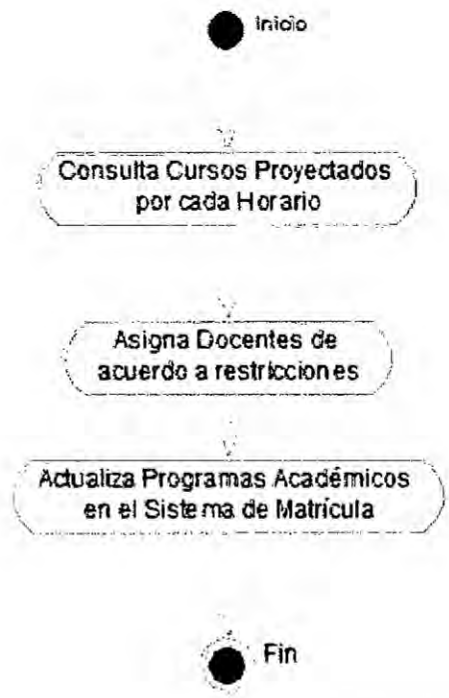

Figura 5.38 Diagrama de Actividad de la Asignación de Docentes

5.3.3. Diagrama de Casos de Uso

Como podemos observar en la Figura 5.9 se tienen 4 casos de uso para el sistema, en el primer caso de uso se establecerán los parámetros de los algoritmos genéticos a utilizar, luego existe otros dos casos donde se implementará los algoritmos genéticos. Y finalmente existe otro caso de uso para poder habilitar o aprobar los programas académicos generados por el algoritmo genético y de esta forma puedan ser utilizados por el sistema de matrícula.

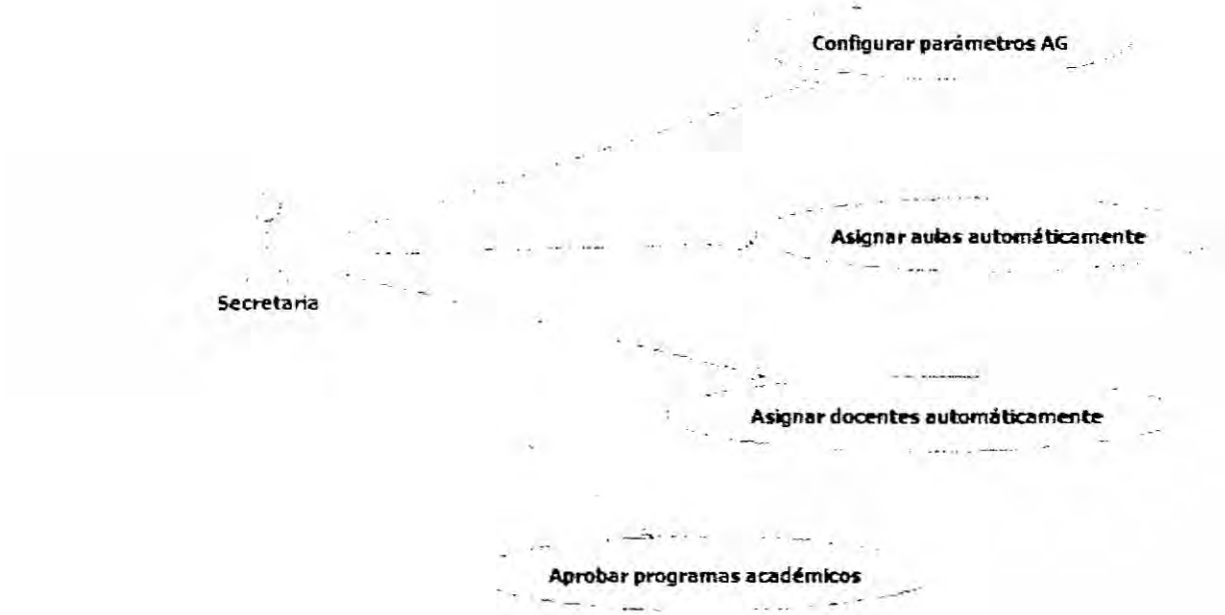

Figura 5.39 Diagrama de Casos de Uso del Sistema.

5.4. Especificación de los Casos de Uso

A continuación describiremos cada caso de uso mediante sus especificaciones correspondientes y sus diagramas de secuencia. Cada especificación de caso de uso está compuesta por una breve descripción, luego se mencionará que actores intervienen en la realización del caso de uso, luego describiremos el flujo de eventos tanto el flujo principal como el flujo alternativo y finalmente se señalarán las precondiciones y postcondiciones. Por otro lado en los diagramas de secuencia se describirá el flujo principal de cada caso de uso.

5.4.1. Caso de Uso Configurar Parámetros AG

5.4.1.1. Especificación

A continuación en la Tabla 5.1 presentamos la especificación del Caso de Uso Configurar Parámetros del Algoritmo Genético (AG).

Descripción	En esta opción del sistema se establecerán los parámetros del Algoritmo Genético, como son el tamaño de la población, el número de generaciones, el tamaño de la tasa de mutación, el tamaño del torneo para hacerlos persistir en la Base de Datos y posteriormente ser utilizados.
Actores	Secretaria (SA), Sistema (S)
Flujo de Eventos	Flujo Principal: - SA: Ingresar el tamaño de la población - SA: Ingresar el número de generaciones - SA: Ingresar el tamaño de la tase de mutación - SA: Ingresar el tamaño del torneo - SA: Grabar los Datos - S: Graba los datos en la Base de Datos Flujo Alternativo: S: Si todos los parámetros no han sido ingresados mostrar mensaje de error.
Precondiciones	Ninguna
Postcondiciones	El Sistema se cargará con parámetros para que puedan ser usados por el algoritmo genético.

Tabla 5.31 Especificación Caso de Uso Configurar Parámetros del AG

5.4.1.2. Diagrama de Secuencia

A continuación en la Figura 5.10 mostramos el diagrama de secuencia del flujo principal del Caso de Uso Configurar Parámetros del AG.

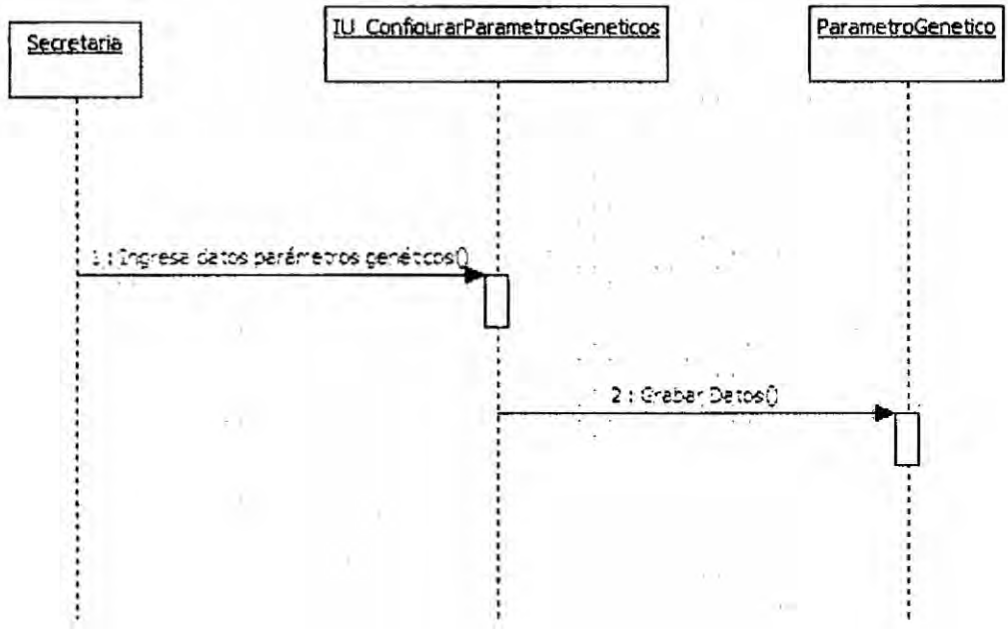

Figura 5.10 Diagrama de Secuencia Caso de Uso Configurar Parámetros del AG

5.4.2. Caso de Uso Asignar Aulas Automáticamente

5.4.2.1. Especificación

A continuación en la Tabla 5.2 presentamos la especificación del Caso de Uso Asignar Aulas Automáticamente.

Descripción	Esta opción del sistema se encargará de hacer la asignación automática de aulas para todos los programas académicos

	proyectados en un semestre determinado a través del uso de algoritmos genéticos.
Actores	Secretaria (SA), Sistema (S).
Flujo de Eventos	Flujo Principal: - SA: Seleccionar una configuración de parámetros genéticos existentes y un periodo de matrícula (semestre y año). - SA: Generar la población inicial de soluciones. - S: Generar la población inicial para la asignación de aulas y enviar confirmación. - SA: Ejecutar el Algoritmo Genético de Asignación de Aulas. - S: Ejecutar el algoritmo genético de asignación de aulas y mostrar los resultados obtenidos (tiempo, costo de la mejor solución, número de generación final, variación con respecto al encargado) - S: Mostrar las asignaciones hechas. - SA: Grabar las asignaciones en los programas académicos. - S: Grabar las asignaciones de aulas en los programas académicos. Flujo Alternativo: - Si el usuario cancela la ejecución del algoritmo genético este mostrará la última mejor solución recorrida. - Si el usuario no selecciona los parámetros genéticos ni el periodo de matrícula no se podrá crear la

	población inicial ni se ejecutará el algoritmo genético de asignación de aulas. - El sistema no podrá ejecutar el algoritmo genético si no se ha generado antes una población inicial de soluciones.
Precondiciones	- Debe haber configuraciones de parámetros genéticos guardados en la Base de Datos. - Debe haber programas académicos proyectados guardados en la Base de Datos, es decir los cursos con sus respectivos horarios ya cargados en el periodo de matrícula indicado.
Postcondiciones	Se tienen los programas académicos con aula asignada listos para ser habilitados a matrícula.

Tabla 5.32 Especificación Caso de Uso Asignar Aulas Automáticamente

5.4.2.2. Diagrama de Secuencia

A continuación en la Figura 5.11 mostramos el diagrama de secuencia del flujo principal del Caso de Uso Asignar Aulas Automáticamente.

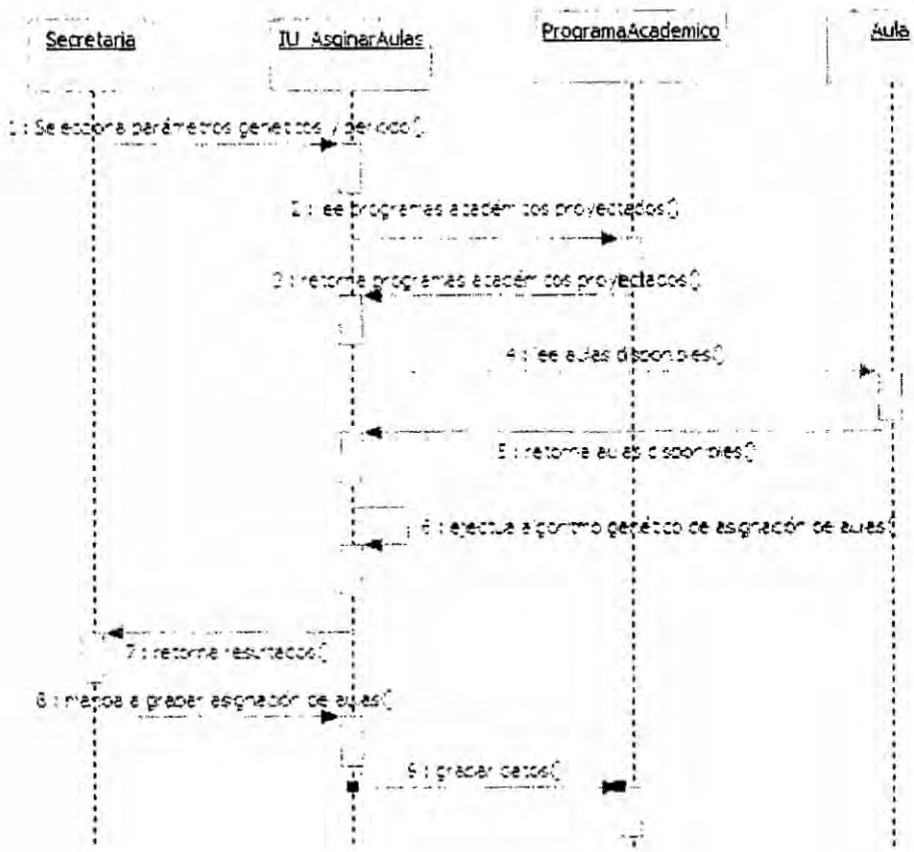

Figura 5.11 Diagrama de Secuencia Caso de Uso Asignar Aulas Automáticamente

5.4.3. Caso de Uso Asignar Docentes Automáticamente

5.4.3.1. Especificación

A continuación en la Tabla 5.3 presentamos la especificación del Caso de Uso Asignar Docentes Automáticamente.

Descripción	Esta opción del sistema se encargará de hacer la asignación automática de aulas de docentes para todos los programas académicos proyectados en un semestre determinado a través

	del uso de algoritmos genéticos.
Actores	Secretaria (SA), Sistema (S).
Flujo de Eventos	Flujo Principal: - SA: Seleccionar una configuración de parámetros genéticos existentes y un periodo de matrícula (semestre y año). - SA: Generar la población inicial de soluciones. - S: Generar la población inicial para la asignación de docentes y enviar confirmación. - SA: Ejecutar el Algoritmo Genético de Asignación de Docentes. - S: Ejecutar el algoritmo genético de asignación de docentes y mostrar los resultados obtenidos (tiempo, costo de la mejor solución, número de generación final, variación con respecto al encargado) - S: Mostrar las asignaciones hechas. - SA: Grabar las asignaciones en los programas académicos. - S: Grabar las asignaciones de docentes en los programas académicos. Flujo Alternativo: - Si el usuario cancela la ejecución del algoritmo genético este mostrará la última mejor solución recorrida. - Si el usuario no selecciona los parámetros genéticos ni el periodo de matrícula no se podrá crear la población inicial ni se ejecutará el algoritmo genético de asignación de docentes.

	- El sistema no podrá ejecutar el algoritmo genético si no se ha generado antes una población inicial de soluciones.
Precondiciones	- Debe haber configuraciones de parámetros genéticos guardados en la Base de Datos. - Debe haber programas académicos proyectados guardados en la Base de Datos, es decir los cursos con sus respectivos horarios ya cargados en el periodo de matrícula indicado.
Postcondiciones	Se tienen los programas académicos con docentes asignados listos para ser habilitados a matrícula.

Tabla 5.3 Especificación Caso de Uso Asignar Docentes Automáticamente

5.4.3.2. Diagrama de Secuencia

A continuación en la Figura 5.12 mostramos el diagrama de secuencia del flujo principal del Caso de Uso Asignar Docentes Automáticamente.

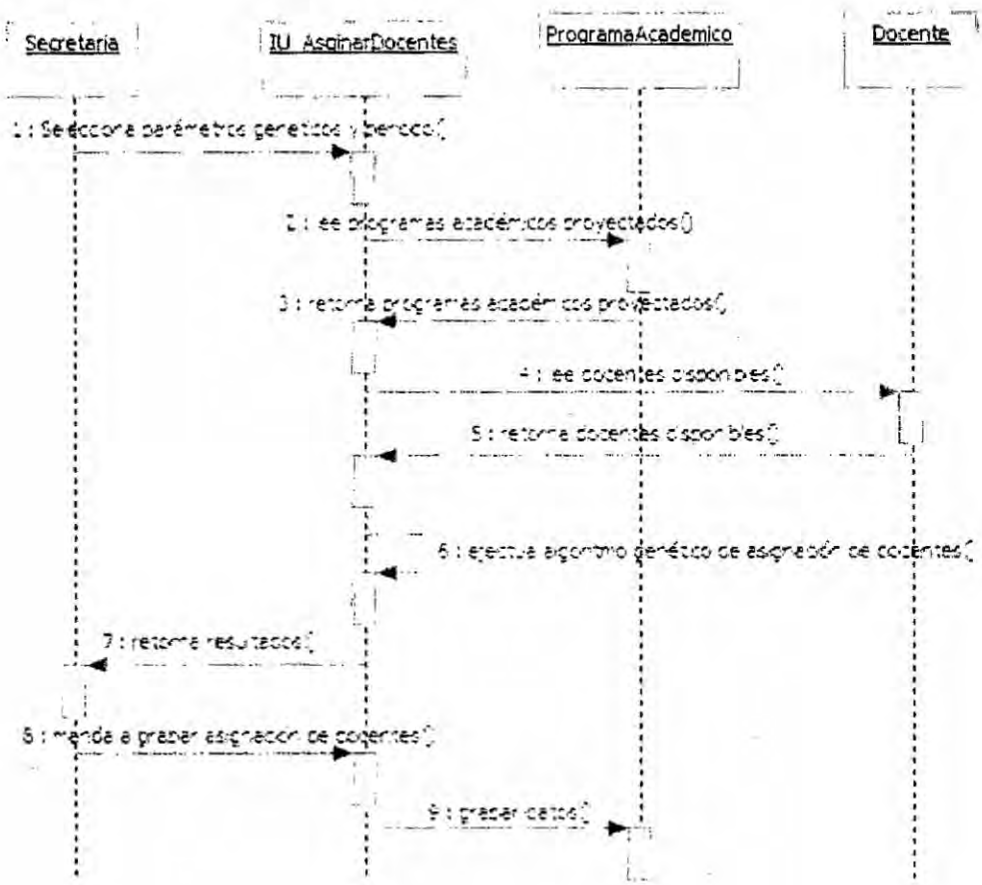

Figura 5.312 Diagrama de Secuencia Caso de Uso Asignar Docentes Automáticamente

5.4.4. Caso de Uso Aprobar Programas Académicos

5.4.4.1. Especificación

A continuación en la Tabla 5.4 presentamos la especificación del Caso de Uso Aprobar Programas Académicos.

Descripción	Esta opción del sistema sirve para aprobar los programas académicos y así poder ser utilizados por el sistema de matrícula.

Actores	Secretaria (SA), Sistema (S).
Flujo de Eventos	Flujo Principal: SA: Seleccionar un periodo de matrícula S: Mostrar los programas académicos del periodo de matrícula seleccionado por el usuario. SA: Aprobar los programas académicos que pasan a matrícula.
Precondiciones	Haber hecho la asignación de aulas y docentes
Postcondiciones	Los Programas Académicos ya se encuentran habilitados para matrícula.

Tabla 5.3 Especificación Caso de Uso Aprobar Programas Académicos

5.4.4.2. Diagrama de Secuencia

A continuación en la Figura 5.13 mostramos el diagrama de secuencia del flujo principal del Caso de Uso Aprobar Programas Académicos.

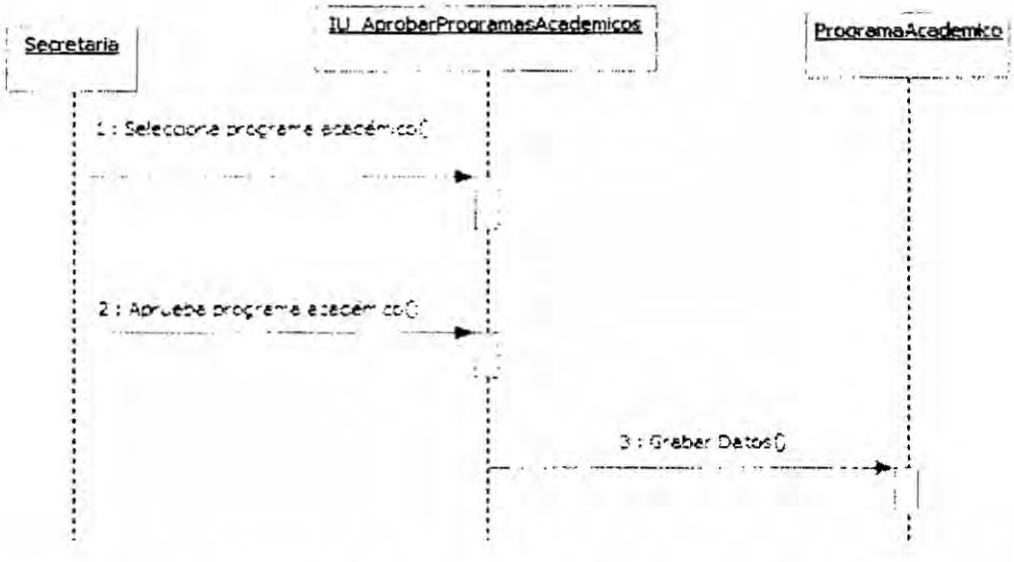

Figura 5.13 Diagrama de Secuencia Caso de Uso Aprobar Programas Académicos

5.5. Diagrama de Clases

A continuación en la Figura 5.14 presentamos el Diagrama de Clases detallado, con todas las entidades, atributos y relaciones de las clases utilizadas por el sistema.

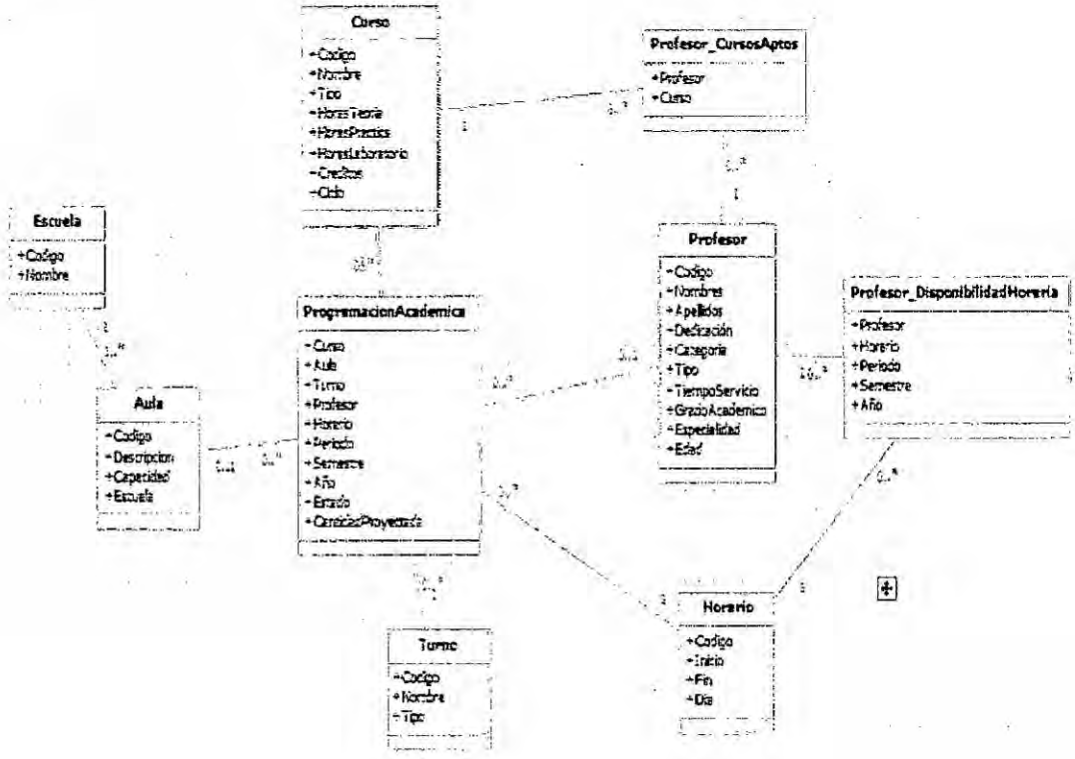

Figura 5.14 Diagrama de Clases Detallado

Entre las entidades que tenemos están: ProgramaAcadémico, Horario, Aula, Profesor, Curso, Profesor_DisponibilidadHoraria, Profesor_CursosAptos, Turno, Escuela; las cuales vamos a describir a continuación.

ProgramaAcademico

Representa el producto final de programar en un horario determinado un curso, un profesor y un aula para el dictado de clases.

Curso

Es el término utilizado para la materia en la cual se imparte el servicio de enseñanza.

Horario

Representa un periodo de tiempo desde una hora inicial hasta una hora final en la cual se dicta un curso un determinado día.

Aula

Representa un ambiente físico o lugar donde se hace el dictado del curso.

Profesor

Es la persona encargada de dictar el curso

Profesor_CursosAptos

Representa los cursos que puede dictar un profesor, según su especialidad y/o experiencia de dictado de cursos.

Profesor_DisponibilidadHoraria

Representa las horas en las que un profesor posee tiempo para dictar una clase en un periodo determinado de matrícula.

Escuela

Representan la escuela profesional para la cual están registrados los alumnos, y de la cual obtendrán un título profesional, siempre y cuando hayan completado todos los cursos obligatorios de la malla curricular o plan de estudios aprobado.

Turno

Representa las opciones en la cual se matriculan los alumnos, un alumno se puede matricular un en turno para un determinado curso y horario con un determinado profesor.

5.6. Modelo de Datos

Dadas las entidades mostradas en el Diagrama de Clases descrito en la sección anterior, presentamos el Diagrama Físico de la Base de Datos del Sistema, este diagrama como se aprecia en Anexo 1: Modelo de Datos de la Solución, nos muestra todos los campos y relaciones entre las entidades de la Base de Datos a nivel físico.

5.7. Arquitectura de la Solución

El Sistema como se puede observar en el Diagrama de Componentes de la Figura 5.15 estará divido en cuatro capas: la capa de presentación, la capa de lógica de negocio, la capa de acceso a datos y la capa de la Base de Datos. En la capa de presentación tendremos todas las interfaces que servirán para el ingreso de los datos y así mismo mostrar los resultados del sistema, en la capa de lógica de negocio estará el algoritmo genético utilizado para la asignación de aulas y de docentes, en la capa de acceso a datos estarán todos los controles necesarios para poder obtener los datos de la Base de Datos y finalmente en la Base de Datos Académica residirá toda la información necesitada.

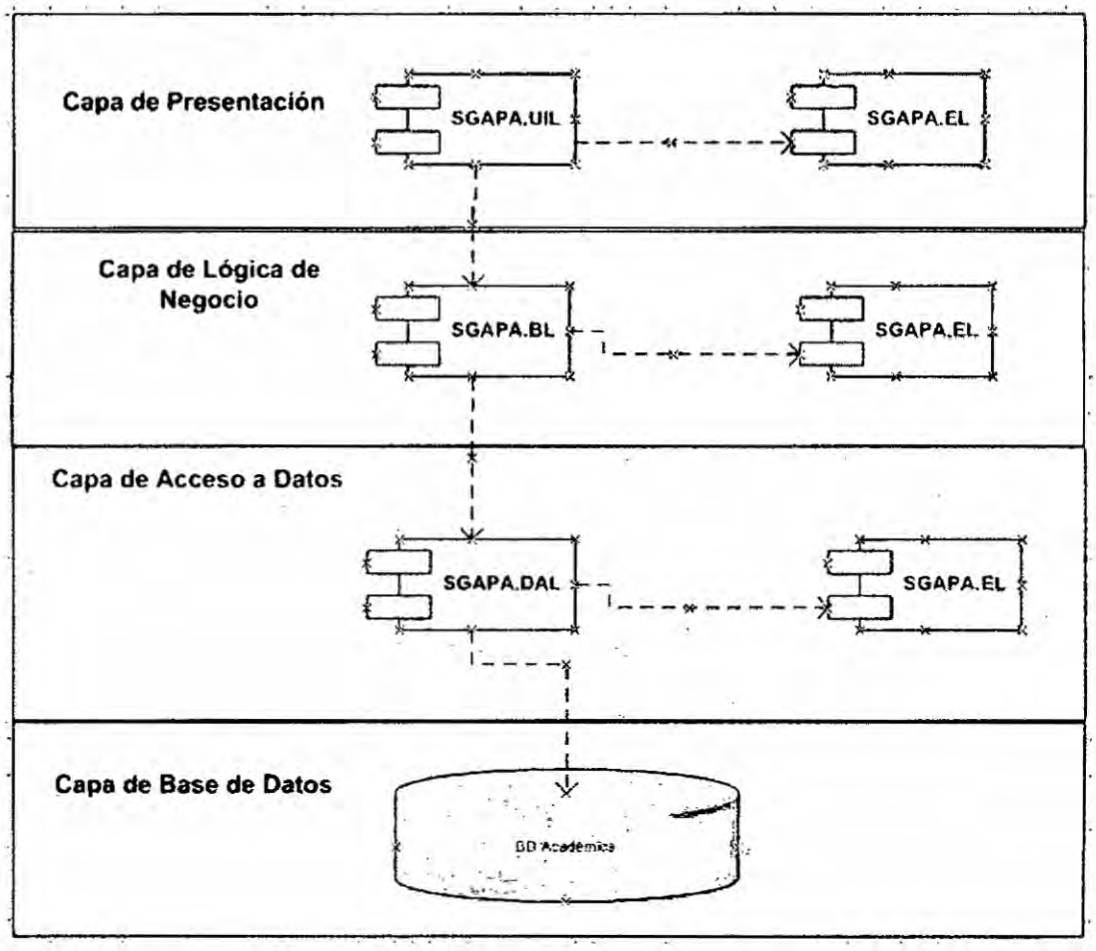

Figura 5.15 Arquitectura de la Solución

Así mismo es preciso recalcar que el sistema es una aplicación Windows Cliente Servidor que está desarrollado sobre la plataforma de Microsoft .Net, por lo que también son necesarios los componentes del Microsoft NetFramework 4.0.

5.8. Despliegue de la Solución

A continuación en la Figura 5.16 presentamos el diagrama de despliegue de la solución en la cual podemos apreciar la forma como todos los componentes de la

aplicación se van a desplegar tanto en el Servidor de Base de Datos como en el cliente.

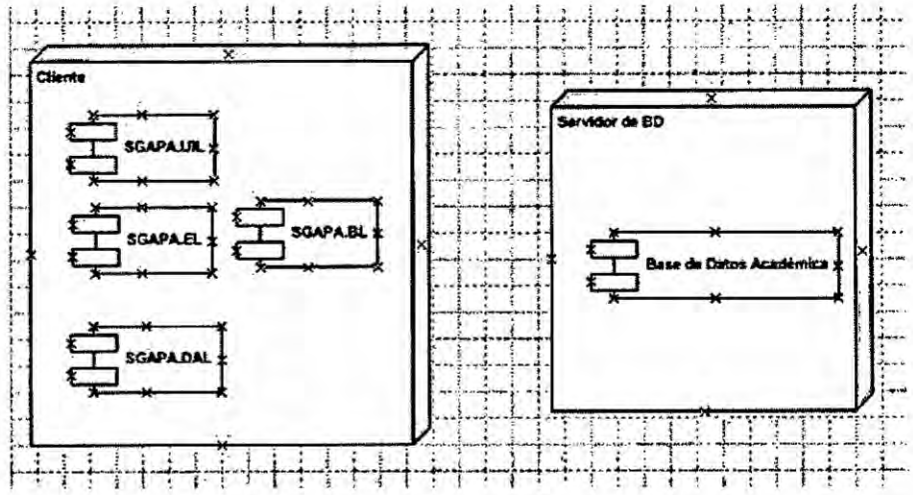

Figura 5.16 Diagrama de Despliegue de la solución

5.9. Prototipos del Sistema

A continuación presentamos cada uno de las pantallas prototipo del sistema. Cada pantalla corresponde a un caso de uso descrito en las secciones anteriores. De esta forma en la Figura 5.17 observamos el prototipo para el caso de uso Configurar Parámetros Genéticos, luego en la Figura 5.17 el prototipo para el caso de uso Asignar Aulas Automáticamente y en la Figura 5.18 el prototipo para el caso de uso Asignar Docentes Automáticamente. Finalmente en la Figura 5.20 encontramos el prototipo para el caso de uso Aprobar Programas Académicos.

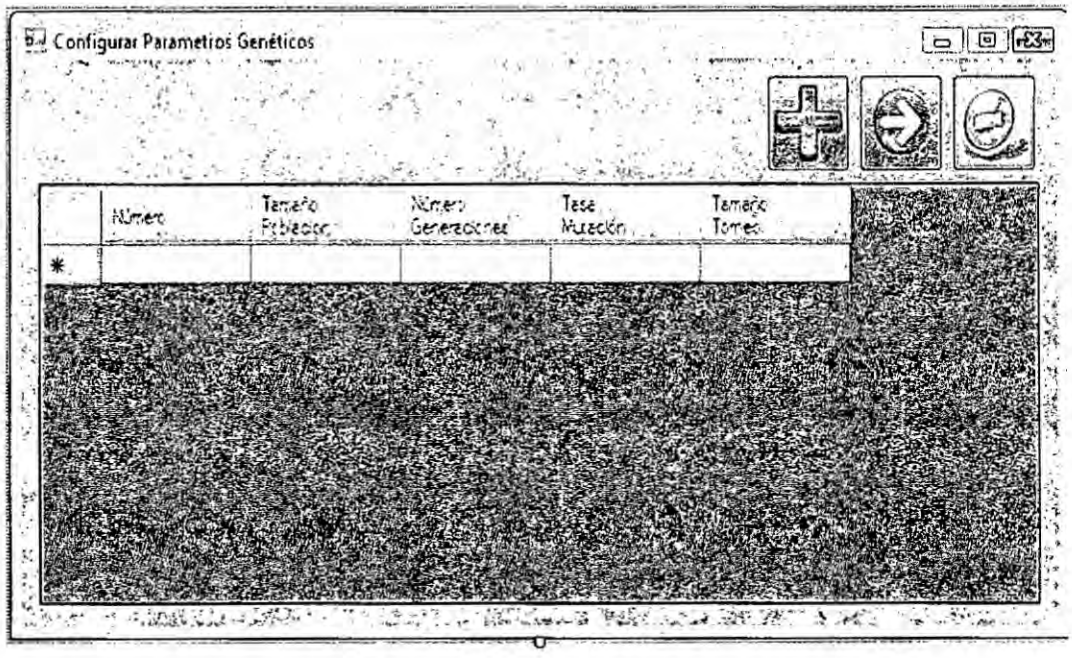

Figura 5.17 Prototipo del Caso de Uso Configurar Parámetros Genéticos.

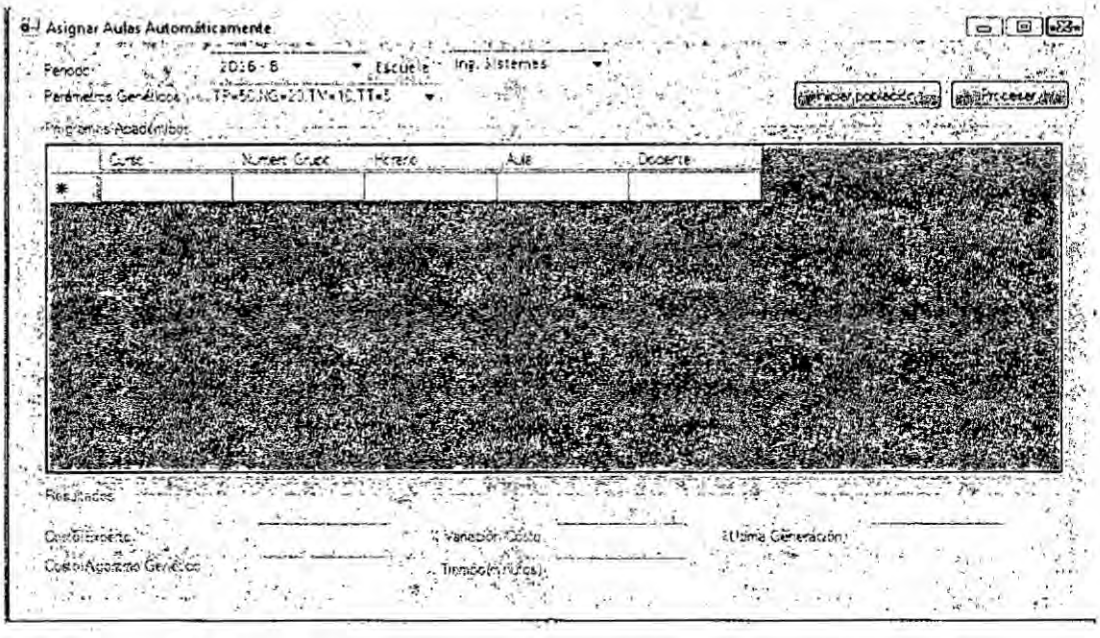

Figura 5.18 Prototipo del Caso de Uso Asignar Aulas Automáticamente.

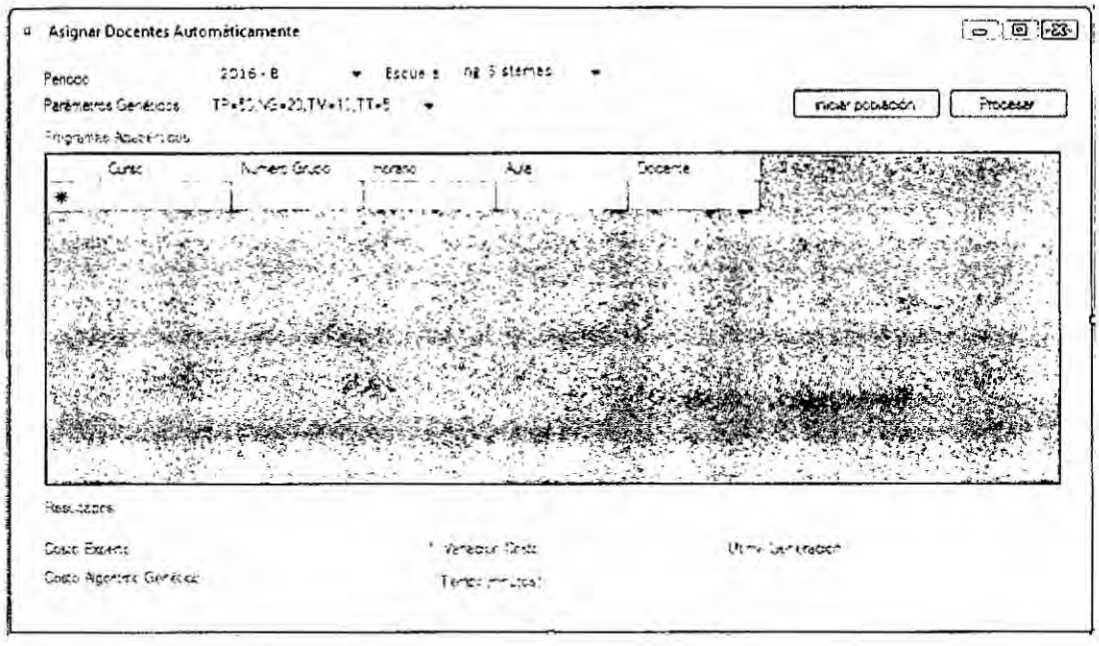

Figura 5.19 Prototipo del Caso de Uso Asignar Docentes Automáticamente.

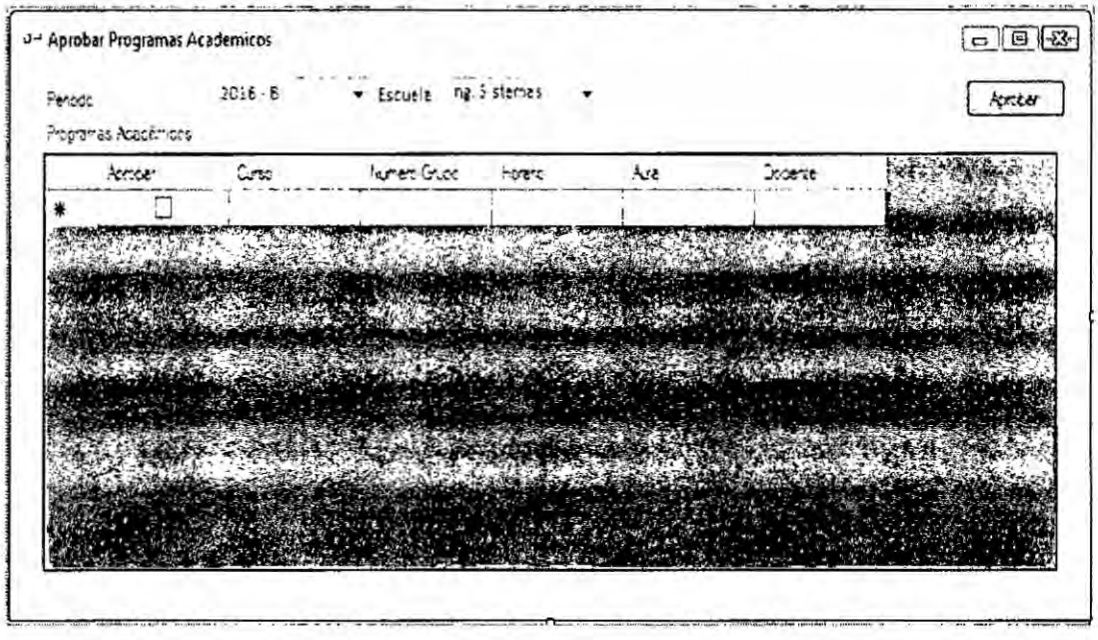

Figura 5.20 Prototipo del Caso de Uso Aprobar Programas Académicos.

5.10. Consideraciones del Entorno de Desarrollo

El sistema ha sido desarrollado en la plataforma .Net utilizando la herramienta Visual Studio 2013. El lenguaje utilizado para el desarrollo de la aplicación ha sido Visual Basic. Net, mientras que el servidor de Base de Datos está bajo Sql Server 2012. Los modelos del análisis y diseño han sido desarrollados utilizando la herramienta StarUML y SQL Designer.

Requerimientos de Software y Hardware

El Sistema será una aplicación Windows Cliente Servidor, por lo que tanto del lado del cliente como del lado del Servidor habrá algunos requerimientos de hardware y de software.

Requerimiento de Hardware y Software del Servidor

Los requerimientos de hardware y software para el servidor son básicamente para el manejo y administración de la Base de Datos. De esta forma en el Software se requiere:

- Una Base de Datos Microsoft Sql Server 2012.
- Un Sistema Operativo Windows 2012 o superior

Mientras que las características de hardware son las siguientes:

- Un Servidor IBM System x3400 con Procesadores Intel Xeon Dual-Core de 2.5 GHz de velocidad y 5 GB de memoria RAM.

Requerimientos de Hardware y Software del Cliente.

Los requerimientos del cliente son los requerimientos para el host donde finalmente va residir la aplicación Windows, de esta forma en cuanto software se requiere lo siguiente:

- Sistema Operativo Windows7 o superior
- NetFramework 4.0

Mientras que en el hardware las características son las siguientes.

- Un computador con procesador Core 3 o superior y con 2GB de memoria RAM.

5.11. Codificación del Algoritmo Genético

A continuación se describirá la codificación de cada parte del algoritmo genético utilizado para la asignación de aulas y docentes. Se empezará por explicar el código para el cálculo de la Función de Costo y Función de Fitness para cada uno de los problemas. Luego se continuará con la presentación del código del algoritmo utilizado para la formación de la población inicial, aquí es preciso señalar que este algoritmo de generación de la población inicial está separado de la rutina principal del algoritmo genético debido al tiempo que toma en generar las "n" soluciones iniciales. Posteriormente se expondrá el código del algoritmo de selección de padres para la siguiente generación. Luego el código del algoritmo de mutación y finalmente la rutina principal del algoritmo genético que llama a los demás métodos anteriores.

5.11.1. Función de Costo y de Fitness

Son dos las funciones de costo y de fitness utilizadas, una para la asignación de aulas y otra para la asignación de docentes, a continuación describiremos cada una de ellas. Las funciones de costo, dadas las restricciones al problema, calculan la cantidad de veces en que se violen o no se satisfagan estas restricciones en cada una de las soluciones (individuos o cromosomas) de la población y asimismo definen costos, penalidades o ponderaciones por el incumplimiento de cada una de estas restricciones. Estos costos están dados de acuerdo a la importancia que tiene el cumplimiento de cada restricción dentro de la solución. De esta forma los costos más altos serán para las restricciones que son más necesarias que se cumplan. Finalmente se suma el producto de los costos por el número de restricciones incumplidas y se obtiene un costo para cada solución de la población. La solución con mejor valor de fitness o que mejor se acomode a los que se necesita será la que tenga un valor mínimo de esta función de costo. Lo que busca el algoritmo genético es tratar de disminuir este costo lo más bajo posible, en el caso de este trabajo es al menos alcanzar el costo dado por el experto o encargado de realizar esta tarea.

Función de Costo

A continuación en la Figura 5.17 y 5.18 presentamos la codificación en lenguaje de programación Visual Basic.NET para el cálculo de costo en la asignación de docentes y aulas respectivamente.

```
Private Function Fx(ByVal Cromosoma()() As ProgramaAcademico) As Double
        'Debe cumplir con las horas obligatorias según dedicación
        Dim numErrores1 As Integer = 0
        'Debe cumplir con las horas obligatorias si son extraordinarios
        Dim numErrores2 As Integer = 0
        'Debe cumplir con solo horarios de práctica y laboratorio para los jefes de práctica
        Dim numErrores3 As Integer = 0
        'Debe cumplir con la disponibilidad
        Dim numErrores4 As Integer = 0
        'Debe cumplir con dictar cursos para los cuales estén aptos
        Dim numErrores5 As Integer = 0
        For j As Integer = 0 To Profes.Length - 1
            Dim numHoras As Integer = 0
            For i As Integer = 0 To Hs.Length - 1
                Dim Programa As ProgramaAcademico = Cromosoma(i)(j)
                filtroHorario = Hs(i).Codigo
                filtroProfesor = Profes(j).Codigo
                Dim d As Disponibilidad = CType(Array.Find(Profes(j).disponibilidad, AddressOf LeerDisponibilidad), Disponibilidad)
                Dim a As CursosAptos = CType(Array.Find(Profes(j).CursosAptos, AddressOf LeerCursosAptos), CursosAptos)
                If Not Programa Is Nothing Then
                    If d Is Nothing Then
                        numErrores4 = numErrores4 + 1
                    End If
                    If a Is Nothing Then
                        numErrores5 = numErrores5 + 1
                    End If
                    numHoras = numHoras + Programa.TotalHoras
                End If
            Next
            Dim c As Categoria = Profes(j).Categoria
            If c.Descripcion = "PRINCIPAL" Or c.Descripcion = "ASOCIADO" Or c.Descripcion = "AUXILIAR" Then
                If Not CumpleHorasporCategoria(c.Descripcion, numHoras) Then
                    numErrores1 = numErrores1 + 1
                End If
            End If
            If c.Descripcion = "EXTRAORDINARIO" Then
                If Not CumpleHorasporCategoria(c.Descripcion, numHoras) Then
                    numErrores2 = numErrores2 + 1
                End If
            End If
            If c.Descripcion = "JEFE PRACTICA" Then
                If Not CumpleHorasporCategoria(c.Descripcion, numHoras) Then
                    numErrores3 = numErrores3 + 1
                End If
            End If

        Next
        Return 0.1 * numErrores1 + 0.1 * numErrores2 + 0.1 * numErrores3 + 0.2 * numErrores4 + 0.5 * numErrores5
    End Function
```

Figura 5.21 Código de la Función de Costo en la Asignación de Docentes

```
Private Function Fx(ByVal Cromosoma()() As ProgramaAcademico) As Double
        'Debe priorizar las aulas mas grandes para los cursos del ciclo 1
        Dim numErrores1 As Integer = 0
        'Debe asignar aulas de acuerdo a la cantidad proyectada de alumnos para los cursos diferentes al ciclo1
        Dim numErrores2 As Integer = 0
        'Debe cumplir con que los cursos con horas de laboratorio sean asignados en aulas de laboratorio
        Dim numErrores3 As Integer = 0

        For j As Integer = 0 To Aulas.Length - 1
            Dim numHoras As Integer = 0
            For i As Integer = 0 To Hs.Length - 1
                Dim Programa As ProgramaAcademico = Cromosoma(i)(j)
                If Not Programa Is Nothing Then
                    Dim Ciclo As Integer = Programa.Curso.Ciclo
                    Dim Proyeccion As Integer = Programa.CantidadProyectada
                    If Ciclo = 1 Then
                        If Programa.Aula.capacidad <> CapacidadMaxima() Then
                            numErrores1 = numErrores1 + 1
                        End If
                    End If
                    If Ciclo <> 1 Then
                        If Proyeccion > Programa.Aula.capacidad Then
                            numErrores2 = numErrores2 + 1
                        End If
                    End If
                    If Programa.Curso.HorasLaboratorio <> 0 Then
                        If Programa.Aula.Tipo <> "LABORATORIO" Then
                            numErrores3 = numErrores3 + 1
                        End If
                    End If
                End If
            Next
        Next
        Return 0.5 * numErrores1 + 0.1 * numErrores2 + 0.4 * numErrores3
    End Function
```

Figura 5.22 Código de la Función de Costo en la Asignación de Aulas

Como se puede observar lo que hacen ambos códigos es contar el número de errores en el cromosoma o solución pasado como parámetro de tal forma que cada variable numErrores representa el número de errores por no cumplir con una restricción. Al final del algoritmo podemos ver que se hace ponderaciones a cada una de estas variables, en el caso de la asignación de docentes se le da más ponderación a cumplir con los cursos aptos, es decir los cursos en los cuáles puede dictar un profesor de acuerdo a su experiencia y conocimiento y en el caso de la asignación de aulas se le da más importancia a cumplir con las aulas asignadas a los cursos del primer ciclo. Esta importancia de los costos fue establecida por los expertos encargados de realizar la tarea de asignaciones y es por eso que se colocan como parámetros fijos, sin embargo se podría utilizarlos como variables de entrada para el algoritmo genético.

Función de Fitness

La función de fitness es calculada basándose en la función de costo, como ya se comentó anteriormente un individuo tendrá mejor fitness si la función de costo es menor. Por lo tanto la función de costo es lo contrario a la función de fitness. Debido a esto y a que los números arrojados por la función de costo son siempre mayores o iguales a 0, el cálculo de la función de fitness estará dado por el producto de la función de costo por el valor de menos -1 como se muestra en el código de la Figura 5.23

```
Private Function Fitness(ByVal Cromosoma()() As Programaacademico) As Double
        Return (-1) * Fx(Cromosoma)
End Function
```

Figura 5.3 Código para cálculo de la Función de Fitness

En ambos casos de la asignación de docentes y asignación de aulas se utiliza la misma función de fitness.

Algoritmo de Generación de la Población Inicial

La generación de la población inicial como se observa en los códigos de las Figura 5.20 y la Figura 5.21 se hace de forma aleatoria seleccionando uno por uno cado uno de los programas académicos proyectados y ubicándolo aleatoriamente en un aula (caso asignación de aulas) o asignándole aleatoriamente un docente (caso asignación de docente) que no se encuentren ocupados dentro de un mismo horario (fila de la matriz de programas académicos). Este proceso se realiza m veces donde m es el tamaño de la población.

```
Private Sub CrearPoblacionInicial_AsignarAulas(ByVal Programas() As ProgramaAcademico)
        Dim m = TAMAÑO_POBLACION
        Dim x As Integer = 0
        Do While x < m
            Dim Q(0 To Hs.Length - 1)() As ProgramaAcademico
            Dim i As Integer = 0
            For i = 0 To Hs.Length - 1
                Dim Ps(0 To Aus.Length) As ProgramaAcademico
                For j As Integer = 0 To Aus.Length - 1
                    Ps(j) = Nothing
                Next
                Q(i) = Ps
            Next
            i = 0
            For i = 0 To Hs.Length - 1
                For Each A As Aula In Aus
                    A.bitAsignada = False
                Next
                filtroHorario = Hs(i).Codigo
                Dim ProgramasxHorario As ProgramaAcademico() = CType(Array.FindAll(Programas, AddressOf LeerProgramasxHorario),
ProgramaAcademico())
                For Each Programa As ProgramaAcademico In ProgramasxHorario
                    Programa.Aula = Nothing
                    Dim r As Integer = 0
                    Do
                        Dim random As New Random
                        r = random.Next(Aus.Length)
                    Loop While Aus(r).bitAsignada
                    Q(i)(r) = Programa
                    Aus(r).bitAsignada = True
                Next
            Next
            Qs(x) = Q
            QPIs(x) = Q
            x = x + 1
            Console.WriteLine(x.ToString())
        Loop
    End Sub
```

Figura 5.24 Código para la Creación de la Población Inicial en la Asignación de Aulas

```
Private Sub CrearPoblacionInicial_AsignarProfes(ByVal Programas() As ProgramaAcademico)
        Dim m = TAMAÑO_POBLACION
        Dim x As Integer = 0
        Do While x < m
            Dim Q(0 To Hs.Length - 1)() As ProgramaAcademico
            Dim i As Integer = 0
            For i = 0 To Hs.Length - 1
                Dim Ps(0 To Profes.Length) As ProgramaAcademico
                For j As Integer = 0 To Profes.Length - 1
                    Ps(j) = Nothing
                Next
                Q(i) = Ps
            Next
            i = 0
            For i = 0 To Hs.Length - 1
                For Each P As Profesor In Profes
                    P.bitAsignado = False
                Next
                filtroHorario = Hs(i).Codigo
                Dim ProgramasHorario As ProgramaAcademico() = CType(Array.FindAll(Programas, AddressOf LeerProgramasHorario),
ProgramaAcademico())
                For Each Programa As ProgramaAcademico In ProgramasHorario
                    Programa.Profesor = Nothing
                    Dim r As Integer = 0
                    Do
                        Dim random As New Random
                        r = random.Next(Profes.Length)
                    Loop While Profes(r).bitAsignada
                    Q(i)(r) = Programa
                    Profes(r).bitAsignado = True
                Next
            Next
            Qs(x) = Q
            QPIs(x) = Q
            x = x + 1
            Console.WriteLine(x.ToString())
        Loop
    End Sub
```

Figura 5.25 Código para la Creación de la Población Inicial en la Asignación de Docentes

Algoritmo de Selección de Padres

El criterio de selección del padre para formar la siguiente generación se va realizar con una variante del algoritmo de selección por competencia propuesto en [Raghavjee+08], para lo cual se va seleccionar aleatoriamente t elementos (donde t es el tournsize o tamaño de la competencia o torneo), de estos t elementos inicialmente se selecciona un elemento aleatoriamente como el mejor, y posteriormente se realizan t-1 iteraciones en la cual se compara el mejor elemento actual con otro elemento B seleccionado aleatoriamente de la competencia, de esta comparación resulta otro

mejor elemento actual que o bien podría ser el mismo mejor actual o el elemento actual B.

Esta rutina será repetida n veces, donde n es el tamaño de la población. A continuación presentamos el código del algoritmo en la Figura 5.26. Tanto para la asignación de docentes como para la asignación de aulas se utilizará el mismo código.

```
Private Function SeleccionarPadre() As Programaacademico()()
        Dim m = TAMAÑO_POBLACION
        Dim random As New Random
        Dim A()() As Programaacademico
        A = Qs(random.Next(TAMAÑO_POBLACION))
        For i As Integer = 0 To TAMAÑO_TORNEO - 1
            Dim j As Integer = random.Next(TAMAÑO_POBLACION)
            Dim B()() As Programaacademico = Qs(j)
            If Fitness(B) > Fitness(A) Then
                A = B
            End If
        Next
        Return A
End Function
```

Figura 5.26 Código para la Selección del Padre

Algoritmo de Mutación

Una vez seleccionados los elementos padres de la siguiente generación se procederá a aplicar el operador de mutación sobre cada padre seleccionado para poder obtener los nuevos hijos de la siguiente generación. El operador de mutación se basa en el algoritmo propuesto en [Raghavjee+08] en el cual se selecciona aleatoriamente una posición de la matriz solución y se intercambia por otra posición de la matriz tratando de encontrar siempre una solución que cumpla con las restricciones del problema. Este proceso se produce s veces donde s es el tamaño de la mutación o número de

intercambios de mutación. A continuación en la Figura 5.27 se presenta el código del algoritmo de mutación para la asignación de aulas.

```
Private Function Mutacion() As Object()
    Dim m = TAMAÑO_POBLACION
    Dim s = TAMAÑO_MUTACION
    Dim x As Integer = 0
    Dim y As Integer = 0
    Dim random As New Random
    Dim ReglasIncumplidas_Actual, ReglasIncumplidas_Nueva As Integer
    Do While x < m
        Dim CromosomoPadre()() As ProgramaAcademico = SeleccionarPadre()
        Dim Q()() As ProgramaAcademico = CromosomoPadre
        y = 0
        Do While y < s
            Dim condicion1, condicion2, condicion3 As Boolean
            Dim i, j, k As Integer
            Dim numeroInternos As Integer = 50
            Dim contIntentos As Integer = 0
            Do
                i = random.Next(Ms.Length)
                j = random.Next(Aus.Length)
                condicion1 = Not CumpleRegla1(Q, i, j)'Debe priorizar las aulas más grandes para los cursos del ciclo i
                condicion2 = Not CumpleRegla2(Q, i, j)'Debe asignar aulas de acuerdo a la cantidad proyectada
                condicion3 = Not CumpleRegla3(Q, i, j)'Debe cumplir cursos con horas de laboratorio en aulas de laboratorio
                contIntentos = contIntentos + 1
            Loop While (condicion1 = False And condicion2 = False And condicion3 = False) And contIntentos <= 50
            contIntentos = 0
            ReglasIncumplidas_Actual = CType(condicion1, Integer) + CType(condicion2, Integer) + CType(condicion3, Integer)
            If condicion1 = True Or condicion2 = True Or condicion3 = True Then
                Do
                    k = random.Next(Aus.Length)
                    If k <> j Then
                        Dim Temp As ProgramaAcademico = Q(i)(j)
                        Q(i)(j) = Q(i)(k)
                        Q(i)(k) = Temp
                        If (condicion1 = True And CumpleRegla1(Q, i, j)) Then condicion1 = False End If
                        If (condicion2 = True And CumpleRegla2(Q, i, j)) Then condicion2 = False End If
                        If (condicion3 = True And CumpleRegla3(Q, i, j)) Then condicion3 = False End If
                        ReglasIncumplidas_Nueva = CType(condicion1, Integer) + CType(condicion2, Integer) + CType(condicior
                        If ReglasIncumplidas_Nueva > ReglasIncumplidas_Actual Then
                            Temp = Q(i)(j)
                            Q(i)(j) = Q(i)(k)
                            Q(i)(k) = Temp
                        End If
                    End If
                    contIntentos = contIntentos + 1
                Loop While (ReglasIncumplidas_Nueva > ReglasIncumplidas_Actual) And contIntentos <= 50
            End If
            y = y + 1
        Loop
        If Fitness(Q) > Fitness(CromosomoPadre) Then
            Ms(x) = Q
        Else
            Ms(x) = CromosomoPadre
        End If
        x = x + 1
    Loop
    Return Ms
End Function
```

Figura 5.27 Código del Algoritmo de Mutación para la Asignación de Aulas

En resumen el algoritmo de mutación para asignación de aulas sigue los siguientes pasos:

1°. Selecciona un cromosoma padre

2°. Busca aleatoriamente una posición en la matriz solución seleccionada (cromosoma padre) que no cumpla con las reglas de negocio requeridas para la asignación de aulas: cumplimiento de la capacidad de aulas (para el caso especial de los cursos del primer ciclo o para los demás cursos), cumplimiento de la cantidad proyectada con la capacidad del aula (para los cursos que no son del primer ciclo), cumplimiento de los cursos con laboratorio en las aulas que cuenta con laboratorio.

3°. Una vez encontrada esta posición se selecciona aleatoriamente otra posición (diferente a la primera) en la misma fila (mismo horario) y se intercambian los programas académicos existentes en estas posiciones (se intercambias las aulas asignadas a cada programa académico seleccionado). Este proceso se realiza hasta que producto del intercambio realizado la cantidad de reglas de negocio incumplidas de la nueva matriz de programas académicos generada producto del intercambio sea menor o igual a la cantidad de reglas de negocio incumplidas de la anterior matriz de programas académicos, antes de hacer el intercambio.

4°. Los pasos 2 y 3 se realizan de manera iterativa S veces donde S es el tamaño de la mutación.

5°. Se compara el fitness de la solución generada por el proceso de mutación con la solución del cromosoma padre. Si el Fitness de la nueva solución es mejor que la del padre entonces esta es guardada en un arreglo de cromosomas que representa la nueva

generación, de lo contrario se conserva el cromosoma de la solución padre para la siguiente generación.

6° Desde el paso 1 al paso 5 se realiza iterativamente hasta completar todos los individuos de la población.

Por otro lado, el algoritmo de mutación de la asignación de docentes es similar al algoritmo de mutación de asignación de aulas, la diferencia radica en las reglas que se deben cumplir para que las generaciones cambien, a continuación en la Figura 5.28 se presente el algoritmo.

```
Private Function Mutacion() As Object()
        Dim m = TAMAÑO_POBLACION
        Dim s = TAMAÑO_MUTACION
        Dim x As Integer = 0
        Dim y As Integer = 0
        Dim random As New Random
        Dim ReglasIncumplidas_Actual As Integer
        Dim ReglasIncumplidas_Nueva As Integer
        Do While x < m
            Dim CromosomaPadre() As ProgramaAcademico = SeleccionarPadre()
            Dim Q() As ProgramaAcademico = CromosomaPadre
            y = 0
            Do While y < s
                Dim condicion1, condicion2, condicion3, condicion4, condicion5 As Boolean
                Dim i, j, k As Integer
                Dim numeroIntentos As Integer = 50
                Dim contIntentos As Integer = 0
                Do
                    i = random.Next(Ms.Length)
                    j = random.Next(Profes.Length)
                    condicion1 = Not CumpleRegla1(Q, i, j) 'Debe cumplir con las horas obligatorias según dedicación
                    condicion2 = Not CumpleRegla2(Q, i, j) 'Debe cumplir con las horas obligatorias si son extraordinarias
                    condicion3 = Not CumpleRegla3(Q, i, j) 'Debe cumplir con horarios de práctica y lab. para los jefes de práctica
                    condicion4 = Not CumpleRegla4(Q, i, j) 'Debe cumplir con la disponibilidad
                    condicion5 = Not CumpleRegla5(Q, i, j) 'Debe cumplir con dictar cursos para los cuales estén aptos
                    contIntentos = contIntentos + 1
                Loop While (condicion1 = False And condicion2 = False And condicion3 = False And condicion4 = False And condicion5 = False) And contIntentos <= 50
                contIntentos = 0
        ReglasIncumplidas_Actual=CType(condicion1,Integer)+CType(condicion2,Integer)+CType(condicion3,Integer)+CType(condicion4,Integer)+CType(condicion5,Integer)
                If condicion1 = True Or condicion2 = True Or condicion3 = True Or condicion4 = True Or condicion5 = True Then
                    Do
                        k = random.Next(Profes.Length)
                        If k <> j Then
                            Dim Temp As ProgramaAcademico = Q(i)(j)
                            Q(i)(j) = Q(i)(k)
                            Q(i)(k) = Temp
                            If (condicion1 = True And CumpleRegla1(Q, i, j)) Then condicion1 = False End If
                            If (condicion2 = True And CumpleRegla2(Q, i, j)) Then condicion2 = False End If
                            If (condicion3 = True And CumpleRegla3(Q, i, j)) Then condicion3 = False End If
                            If (condicion4 = True And CumpleRegla4(Q, i, j)) Then condicion4 = False End If
                            If (condicion5 = True And CumpleRegla5(Q, i, j)) Then condicion5 = False End If
ReglasIncumplidas_Nueva=CType(condicion1,Integer)+CType(condicion2,Integer)+CType(condicion3,Integer)+CType(condicion4,Integer)+CType(condicion5,Integer)
                            If ReglasIncumplidas_Nueva > ReglasIncumplidas_Actual Then
                                Temp = Q(i)(j)
                                Q(i)(j) = Q(i)(k)
                                Q(i)(k) = Temp
                            End If
                        End If
                        contIntentos = contIntentos + 1
                    Loop While (ReglasIncumplidas_Nueva > ReglasIncumplidas_Actual) And contIntentos <= 50
                End If
                y = y + 1
            Loop
            If Fitness(Q) > Fitness(CromosomaPadre) Then
                Ms(x) = Q
            Else
                Ms(x) = CromosomaPadre
            End If
            y = y + 1
        Loop
        If Fitness(Q) > Fitness(CromosomaPadre) Then
            Ms(x) = Q
        Else
            Ms(x) = CromosomaPadre
        End If
        x = x + 1
    Loop
    Return Ms
End Function
```

Figura 5.28 Código del Algoritmo de Mutación para la Asignación de Docentes

A continuación se resumen paso a paso el funcionamiento del algoritmo:

1°. Selecciona un cromosoma padre

2°. Busca aleatoriamente una posición en la matriz solución seleccionada (cromosoma padre) que no cumpla con las reglas de negocio requeridas para la asignación de docentes: cumplimiento de horas obligatorias de profesores según dedicación, cumplimiento de horas obligatorias de profesores extraordinarios, cumplimiento de horas asignadas de cursos de laboratorio o práctica para jefes de práctica, cumplimiento de disponibilidad horario del profesor y finalmente cumplimiento de aptitud del profesor para el dictado del curso .

3°. Una vez encontrada esta posición se selecciona aleatoriamente otra posición (diferente a la primera) en la misma fila (mismo horario) y se intercambian los programas académicos existentes en estas posiciones (se intercambias los profesores asignadas a cada programa académico seleccionado). Este proceso se realiza hasta que producto del intercambio realizado la cantidad de reglas de negocio incumplidas de la nueva matriz de programas académicos generada producto del intercambio sea menor o igual a la cantidad de reglas de negocio incumplidas de la anterior matriz de programas académicos, antes de hacer el intercambio.

4°. Los pasos 2 y 3 se realizan de manera iterativa S veces donde S es el tamaño de la mutación.

5°. Se compara el fitness de la solución generada por el proceso de mutación con la solución del cromosoma padre. Si el Fitness de la nueva solución es mejor que la del padre entonces esta es guardada en un arreglo de cromosomas que representa la nueva

generación, de lo contrario se conserva el cromosoma de la solución padre para la siguiente generación.

6° Desde el paso 1 al paso 5 se realiza iterativamente hasta completar todos los individuos de la población.

Rutina Principal del Algoritmo Genético

Finalmente presentamos la rutina principal que llama a las demás rutinas presentadas anteriormente, como podemos ver la rutina itera una cantidad de veces igual al número de generaciones y se detiene cuando encuentra una solución que tiene como costo el valor de cero (la mejor solución del problema ha sido encontrada) o cuando se ha cumplido el número de generaciones (es decir cuando el algoritmo ha iterado la totalidad de generaciones establecida como parámetro genético en búsqueda de la mejor solución). A continuación en la Figura 5.29 podemos ver el código de la rutina principal del algoritmo genético.

```
Private Sub AlgoritmoGenetico_Aulas()
        Dim inicio As New DateTime
        inicio = DateTime.Now
        Dim x As Integer = 0
        While x < TAMAÑO_ITERACIONES
            Qs = Mutacion()
            x = x + 1
            If MejorCosto(Qs) = 0 Then
                Exit While
            End If
        End While
        ULTIMA_GENERACION = x
        Dim tiempo As Double =
((DateTime.Now.Subtract(inicio)).TotalSeconds) / 60
        Dim CE As Double = Double.Parse(txtCostoExperto.Text)
        Dim CAG As Double = MejorCosto(Qs)
        Dim VC As Double = ((CE - CAG) / CE) * 100
End Sub
```

Figura 5.29 Código de la Rutina Principal del Algoritmo Genético para la Asignación de Aulas.

Como hemos podido ver en este apartado hemos presentado cada uno de los componentes del algoritmo genético desarrollado tanto para la asignación de docentes como para la asignación de aulas. La mejor solución obtenida por el algoritmo genético nos servirá para luego guardar todo este resultado en la base de datos recorriendo toda la matriz de programas académicos con las aulas y docentes asignados.

CAPITULO VI. DISCUSIÓN DE RESULTADOS

En esta parte se discuten y analizan algunos experimentos, los mismos que se comparan con los obtenidos por el usuario encargado de realizar la programación académica. Los resultados son evaluados en función al costo que el algoritmo genético trata de minimizar y al tiempo en que se obtuvieron los resultados. Al final se compara el costo del algoritmo genético diseñado para la asignación de aulas y docentes con el costo anterior de alguna asignación generada de forma manual por los encargados.

6.1. Contrastación de hipótesis con resultados

Proceso de experimentos en la asignación de docentes

Los experimentos en la asignación de docentes se realizaron de dos formas, en la primera se creó una población inicial y a partir de ella se ejecutó el algoritmo genético una sola vez manteniendo todos los parámetros genéticos fijos. En los segundos experimentos esperando conseguir mejores resultados se generó una población inicial y a partir de ella se ejecutó varias veces el algoritmo genético variando los parámetros genéticos en cada ejecución. A continuación presentamos cada uno de los experimentos realizados.

Proceso de experimentos con parámetros genéticos fijos.

Se realizaron 6 experimentos diferentes de manera independiente, cada uno bajo su propia población inicial diferente y se obtuvieron los resultados presentados en la

Tabla 6.1. En esta tabla se tiene el Número del Experimento (N°), el tamaño de la población (TP), el tiempo en que se ejecutó la población inicial y el mejor costo obtenido de las primeras soluciones generadas aleatoriamente (tPI y CPI respectivamente), el tamaño del torneo (TT), el tamaño de mutación o tasa de mutación (TM), el número de generaciones (NG), el costo del algoritmo genético obtenido (CAG), el tiempo en que se ejecutó el algoritmo genético (tAG), el costo del experto (el cual ha sido obtenido al evaluar la función de costo definida y utilizada por el algoritmo genético en un conjunto de programas académicos en los cuáles un experto realizó la asignación de docentes en el mismo semestre y partiendo de lo mismos cursos proyectados). Dado el costo del experto se calcula el porcentaje de variación del costo del algoritmo genético (CAG) sobre el costo del experto, este cálculo está dado por la siguiente fórmula: (((CE – CAG)/CE)*100)

N°	TP	tPI	CPI	TT	TM	NG	CAG	tAG	CE	%Var,
1,00	25,00	3,01	97,33	50,00	10,00	30,00	45,65	5,32	60,00	23,92
2,00	50,00	3,50	100.45	50,00	20,00	30,00	40,89	6,13	60,00	31,85
3,00	75,00	5,37	118,00	50,00	30,00	30,00	38,45	7,18	60,00	35,92
4,00	100,00	6,55	126,50	50,00	10,00	30,00	35,81	8,11	60,00	40,32
5,00	125,00	7,56	140,12	50,00	20,00	30,00	30,12	9,21	60,00	49,80
6,00	150,00	9,15	148,14	50,00	30,00	30,00	28,42	10,88	60,00	52,63

Tabla 6.1 Tabla de Resultados Para la Asignación de Docentes con parámetros Genéticos Fijos en cada experimento.

Como podemos observar de la Tabla 6.1, en todos los casos se obtiene un costo mejor que la del experto en realizar la tarea de asignación, llegándose a alcanzar una mejora máxima del 52.63% con respecto al costo del experto en casi 19 minutos (suma de tPI y tAG).

Proceso de experimentos con parámetros genéticos variables

Al tratar de obtener mejores resultados de los que ya se habían obtenido se optó por generar sólo una población inicial y sobre la misma hacer cambios en los parámetros genéticos en cada experimento, por lo que se empezó a hacer cambios en el tamaño de mutación o tasa de mutación, de tal forma que la tasa de mutación se mantuvo alta en las primeras generaciones y se fue reduciendo conforme se volvía a correr el algoritmo genético sobre las mismas soluciones obtenidas en la ejecución anterior. Todo esto se hizo debido a que se observó que en las primeras generaciones se producían muchos intercambios entre las posiciones de los elementos de la matriz solución consiguiendo una diversidad de soluciones que lograban mejorar los resultados de las generaciones padres. Pero posteriormente cuando el algoritmo demoraba en encontrar soluciones mejores era necesario reducir la tasa de mutación de tal forma que se hagan pequeños cambios sobre las soluciones y se obtengan mejores resultados.

Dado esto se hizo varias corridas del algoritmo genético de asignación de docentes bajo dos experimentos con poblaciones iniciales generadas aleatoriamente y sobre

otro conjunto de programas académicos proyectados que tenían como costo del experto el valor de 60. A continuación vamos a mostrar los resultados obtenidos en cada una de las corridas del algoritmo bajo cada una de las poblaciones iniciales.

La primera población inicial generada tuvo un tamaño de 75 individuos o soluciones y tomó un tiempo de 3.4 minutos en generarse. Luego se realizaron 20 corridas del algoritmo genético de asignación de docentes. Para esto se mantuvo constante el valor del Tamaño del Torneo en 50, mientras que los valores del número de intercambios de mutación o tasa de mutación (TM) y el número de generaciones (NG) se cambiaron con cada corrida esperando conseguir mejores resultados. Los valores usados para ambos parámetros son mostrados en la Tabla 6.2. Asimismo en esta tabla se muestra el valor del tiempo en minutos (t) que demoró cada corrida y los valores de los costos obtenidos tanto por el algoritmo genético (CAG) como por el experto (CE). A continuación se muestran los resultados obtenidos en la Tabla 6.2.

N°	TM	NG	CAG	t	CE	%Var,
1	30	5	37,82	0,60	60	36,97
2	20	10	36,56	1,20	60	39,07
3	20	10	35,88	1,32	60	40,2
4	20	10	34,22	1,28	60	42,97
5	20	10	34,11	1,25	60	43,15
6	20	10	33,13	1,31	60	44,78
7	10	10	29,2	1,29	60	51,33
8	10	20	28,7	1,37	60	52,17
9	10	20	26,4	1,35	60	56
10	10	20	23,56	1,39	60	60,73
11	10	20	22,44	1,36	60	62,6
12	10	20	20,19	1,39	60	66,35
13	5	20	18,8	1,40	60	68,67
14	5	20	18,2	1,41	60	69,67
15	5	20	17,98	1,39	60	70,03
16	2	20	17,46	1,38	60	70,9
17	2	20	17,1	1,36	60	71,5
18	2	20	16,87	1,37	60	71,88
19	1	20	16,29	1,39	60	72,85
20	1	20	15,78	1,37	60	73,7

Tabla 6.2 Resultados obtenidos por el Algoritmo Genético de Asignación de Docentes sobre una Primera Población Inicial de 20 Soluciones.

Como podemos ver de la Tabla 6.2 se consigue alcanzar y mejorar el costo del experto después de 5 generaciones (en la primera corrida) con la misma tasa de mutación (30 intercambios) en un tiempo de 0.60 minutos. Asimismo se puede ver de que hasta la sexta corrida (luego de 55 generaciones) y con la misma tasa de mutación (20) se alcanzó a obtener un costo de 33.13 en 6.96 minutos. En las posteriores corridas se bajó la tasa de mutación en cada una de ellas hasta que en la corrida

número 20 se obtuvo la mejor solución con un costo de 15.78 luego de 325 generaciones adicionales en 26.18 minutos adicionales.

En el segundo experimento realizado se volvió a correr otra población inicial de 150 individuos en un tiempo de 8.98 minutos, luego con un tamaño de torneo de 50 se realizaron 10 corridas del algoritmo genético con diferentes parámetros genéticos en cada corrida, obteniéndose los resultados mostrados en la Tabla 6.3.

N°	TM	NG	CAG	t	CE	%Var,
1	30	5	27,34	2,11	60	54,43
2	20	10	26,89	2,43	60	55,18
3	20	10	25,32	2,62	60	57,8
4	20	10	24,78	2,65	60	58,7
5	20	10	23,52	2,71	60	60,8
6	10	20	21,45	2,87	60	64,25
7	10	20	19,32	2,85	60	67,8
8	10	20	17,49	2,82	60	70,85
9	10	20	17,02	2,79	60	71,63
10	10	20	16,89	2,76	60	71,85

Tabla 6.3 Resultados obtenidos por el Algoritmo Genético de Asignación de Docentes sobre una Segunda Población Inicial de 10 Soluciones.

De los resultados obtenidos en este segundo experimento con el algoritmo genético de asignación de docentes podemos notar que los resultados son similares que en el primer experimento ya que se logró alcanzar un buen costo del algoritmo genético en las 5 primeras generaciones (en la primera corrida) con un costo de 27.34 en 2.11

minutos con una tasa de mutación de 30. La tasa de mutación influyo en los resultados obtenidos ya que se hicieron menos intercambios de mutación que en el experimento anterior. Finalmente como observamos en la Tabla 6.3 se logró alcanzar un costo de 16.89 después de 10 corridas, 145 generaciones y 26.61 minutos, este costo y tiempo fue similar al del primer experimento.

En general se pudo notar que al hacer cambios en los valores de la tasa de mutación entre cada corrida del algoritmo genético los resultados mejoraban de manera más rápida, esto lo podemos observar en la Figura 6.1 y Figura 6.3 donde se ha graficado el comportamiento del costo según la tasa de mutación para cada experimento.

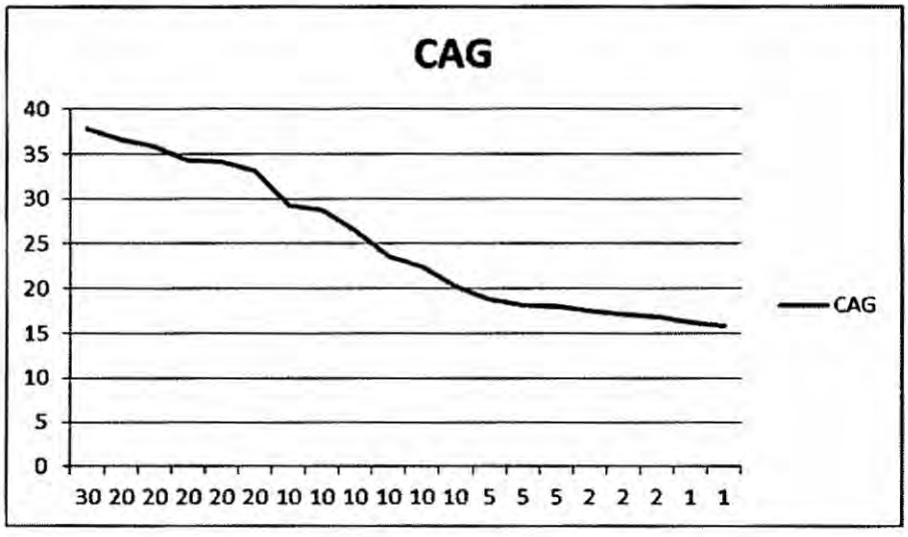

Figura 6.1 Comportamiento del Costo con Respecto al Tamaño de mutación en el primer experimento

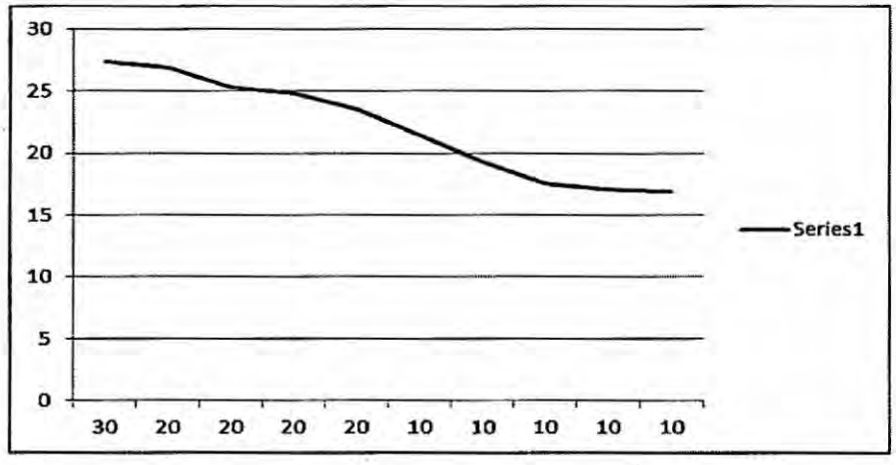

Figura 6.2 Comportamiento del Costo con Respecto al Tamaño de mutación en el segundo experimento

De los resultados obtenidos en ambos experimentos se observa que siempre se mejora el costo y el tiempo con respecto al costo y tiempo del experto (Costo de 60 y el tiempo es de alrededor de 5 o 6 días) mejorándolo en el primer experimento en un 73.7% luego de 325 generaciones y en un tiempo de 26.18 + 3.4 = 29.58 minutos. Mientras que en el segundo experimento se mejoró el costo del experto en un 71.85% luego de 145 generaciones en 26.61 + 8.98 = 35.59 minutos.

Proceso de experimentos en la asignación de aulas

Para la asignación de aulas se generaron 6 diferentes poblaciones iniciales con diferentes valores del tamaño de la población (TP), tasa de mutación (TM), obteniéndose diferentes tiempos (t) y costos del algoritmo genético (CAG) como se

muestra a continuación en la Tabla 6.4. El número de generaciones y el tamaño del torneo en todos los experimentos fueron de 20 y 5 respectivamente.

N°	TP	TM	NG	CAG	T	CE	%Var.
1	10	5	20	5,78	0.0078	35,4	83,67
2	20	5	20	4,43	0.02	35,4	87,49
3	20	10	20	4,11	0.04	35,4	88,39
4	30	20	20	3,86	0.09	35,4	89,1
5	30	30	20	3,21	0.09	35,4	90,93
6	30	40	20	2,95	0.09	35,4	91,67

Tabla 6.4 Resultados obtenidos por el Algoritmo Genético de Asignación de Aulas.

Dados estos resultados podemos ver que en todos los casos se alcanzan a mejorar los costos del experto. El mejor costo que se obtuvo fue de 2.95 en el experimento 6. Asimismo se puede apreciar que el costo alcanzado por el algoritmo genético logró mejorar el costo del experto en un 91.67%.

En estos experimentos podemos apreciar que la tasa de mutación sigue influyendo en los resultados obtenidos, es así que cuanta más alta es la tasa de mutación se obtuvo los mejores resultados.

No se realizaron experimentos con tasas variables debido a que el costo y el tiempo fue muy favorable en todos los experimentos realizados en la asignación de aulas.

CAPITULO VII. CONCLUSIONES

7.1. Conclusiones

El principal objetivo de este trabajo de investigación se logró a través de la construcción de una solución que ayuda a reducir el tiempo destinado a la creación de horarios académicos para una escuela profesional universitaria de una facultad.

El algoritmo genético de asignación de aulas y de docentes obtiene mejores resultados que los costos alcanzados por las personas encargadas de realizar la misma tarea, llegándose a alcanzar una mejora porcentual con respecto al experto de 73.7% y 91.67% en la asignación de docentes y aulas respectivamente.

Al automatizar la asignación de aulas y docentes se reduce notablemente el tiempo, la misma tarea antes era realizada en un promedio 5 a 6 días, mientras que con el algoritmo genético ya se están obteniendo resultados comparados con los del experto antes de los 5 primeros minutos en promedio.

Los parámetros genéticos tienen significativa importancia sobre los resultados obtenidos, en especial el del tamaño de mutación o número de intercambios de mutación.

De los experimentos realizados y de las experiencias obtenidas a lo largo del desarrollo del algoritmo genético se concluye que los algoritmos genéticos son más efectivos cuando se ajustan mejor a las realidades de estudio donde se aplican y que no se puede hacer un algoritmo general para cualquier solución, siempre hay un componente propio del dominio del problema que se intenta resolver.

Se efectuó un análisis en profundidad del problema de generar programas académicos para una escuela profesional, contando para ello con la participación de los encargados de realizar esta tarea y utilizando data real.

Se pudo construir el algoritmo genético ad hoc para la realidad de la organización de estudio, obteniéndose resultados que sobrepasaron el 70% de mejora sobre los costos obtenidos por los expertos.

CAPITULO VIII. RECOMENDACIONES

8.1. Recomendaciones

Es posible aplicar algoritmos genéticos para la programación académica de cualquier institución educativa, por ser una técnica estándar que se puede aplicar rápidamente, aunque su modelamiento y los ajustes son fundamentales para alcanzar una solución al problema a resolver.

Es posible juntar el algoritmo genético de asignación de aulas y el de asignación de docentes en un solo y evaluar los resultados obtenidos, esperando siempre conseguir mejores resultados que haciéndolos de manera separada tal como se desarrolló en este trabajo.

Es posible utilizar el diseño de agentes propuesto por Yang, en el cual cada restricción sería un agente inteligente, el cual sería muy fácil de actualizar sin alternar mucho el diseño de la solución ello permitiría maneja de mejor forma la naturaleza variante de las restricciones del problema.

El know how generado por esta investigación pueda ser utilizado y mejorado para otras aplicaciones prácticas.

REFERENCIAS BIBLIOGRÁFICAS

1. **POHLHIM, Harmut** "Evolutionary Algotithms". 2006
2. **SUEYCHYUN, Fang.** "University Course Scheduling System (UCSS): A UML application with database and visual programming, ACM Digital Library, Journal of Computing Sciences in Colleges", v. 20, 2005, pp 160-169.
3. **TIM, M.** "Artificial Intelligence". Editorial Infinity Science Press. 2008.
4. **VILLAVICENCIO, Jhony.** "Sistema Informático Para el Planteamiento de un Adecuado Sistema de Medición en una Red Eléctrica usando Algoritmos Genéticos" Tesis Universidad Ricardo Palma 2007.
5. **SANTOS Harold, OCHI Luiz, SOUZA Marcote**. "A Tabu Search heuristic with efficient diversification strategies for the class/teacher timetabling problem, ACM Digital Library, Journal of Experimental Algorithmic(JEA)" , v. 10, n. 2.9, 2005.
6. **KALYNMOY Deb.** "Multi-Objective Optimization Using Evolutionary Algorithms". England Editorial John Wiley & Sons Ltda 2001.
7. **RODRÍGUEZ ULLOA, Ricardo.** "La sistémica, los sistemas blandos y los sistemas de información". Lima Universidad del Pacífico. 1994.

8. **YANG Yan, PARANJAPE Raman**, BENEDICENTI Luigi. " An agent base general solution model for the course timetabling problem, ACM Digital Library, International Conference on Autonomous Agents". 2006, pp 1430-1432.

9. **BEASLEY D, MARTIN R.R, & BULL D.R**. "An overview of genetic algorithms: Part 1. Fundamental. University computing. 1993.

ANEXOS

MODELO BASE DATOS