

UNIVERSIDAD MAYOR DE SAN ANDRÉS
FACULTAD DE CIENCIAS PURAS Y NATURALES
CARRERA DE INFORMÁTICA



TESIS DE GRADO

**“DESIGNACIÓN DINÁMICA DE INFRAESTRUCTURA
ACADÉMICA APLICANDO ALGORITMOS GENÉTICOS”**

**PARA OPTAR AL TÍTULO DE LICENCIATURA EN INFORMÁTICA
MENCIÓN: INGENIERÍA DE SISTEMAS INFORMÁTICOS**

POSTULANTE : Luís Pacajes Quispe
TUTOR : Lic. Freddy Miguel Toledo Paz
REVISOR : Lic. Miguel Cotaña Mier MSc.

LA PAZ – BOLIVIA

2009

Dedico esta Tesis a:

A Dios por ser quien ha estado a mi lado en todo momento dándome las fuerzas necesarias para continuar luchando día tras día y seguir adelante rompiendo todas las barreras que se me presenten. Y sobre todo a mi querida madre, ya que gracias a ella soy quien soy hoy en día, fue la que me dio ese cariño y calor humano necesario, la que ha velado por mi salud, mis estudios, mi educación alimentación entre otros, es a ella a quien le debo todo, horas de consejos , de regaños, de reprimendas de tristezas y de alegrías de las cuales estoy muy seguro que lo ha hecho con todo el amor del mundo para formarme como un ser integral y de las cuales me siento extremadamente orgulloso.

A mis hermanos por su apoyo, confianza y amor.

A mis queridos Sobrinos por robarme una sonrisa en los momentos más difíciles.

“Es la hora de partir, la dura y fría hora que la noche sujeta a todo horario.”

(Pablo Neruda)

AGRADECIMIENTOS

Deseo expresar mi agradecimiento a mi segundo hogar, la Universidad Mayor de San Andrés por acogerme todos estos años.

A cada un de los docentes de la Facultad de Ciencias Puras y Naturales por su enseñanza y formación a lo largo de toda mi vida universitaria.

Un cordial agradecimiento al Lic. Freddy Miguel Toledo Paz por sus consejos y recomendaciones que me ayudaron mucho en la realización de la presente tesis.

Un agradecimiento muy especial al Mg. Sc. Miguel Cotaña Mier, ya que este trabajo no hubiera podido realizarse sin el apoyo y los consejos tan acertados que en su momento me brindo. Gracias por su colaboración incondicional.

Por ultimo agradezco a todos mis amigos quienes me acompañaron en el transcurso de estos años, en especial a Willy, Nelzon, Alejandra y Julio por ser amigos de verdad, por todo el apoyo y comprensión que incondicionalmente me brindan.

Y no me puedo ir sin antes decirles, que sin ustedes a mi lado no lo hubiera logrado, tantas desveladas sirvieron de algo y aquí esta el fruto. Les agradezco a ustedes con toda mi alma el haber llegado a mi vida y el compartir momentos agradables y momentos tristes, pero esos momentos son los que nos hacen crecer y valorar a las personas que nos rodean. Los quiero mucho y nunca los olvidaré.

RESUMEN

La mayoría de las instituciones académicas enfrentan uno de los problemas más típicos en la designación de infraestructura, como ser mala distribución de aulas y cruce de horarios, ocasionando grandes perjuicios a la institución y estudiantes.

Muchas de las instituciones emplean métodos tradicionales (matrices prediseñadas, búsqueda secuencial y otras) para la designación de infraestructura resultando en algunos casos satisfactorios y en otros originando resultados no factibles. Razón por la que se decide realizar un estudio y análisis del problema identificado, para lo cual se propone un modelo donde se aplica una metodología de búsqueda de soluciones como son los Algoritmos Genéticos.

Esta metodología se desempeña bajo ciertos componentes y operadores definidos, las mismas son adaptadas a las características y restricciones del problema para obtener un resultado viable.

A través de un prototipo se observa el comportamiento de los Algoritmos Genéticos y se demostrará la obtención de resultados factibles. Llegando a la conclusión de que los Algoritmos Genéticos son considerados como un método mas para la solución a este problema.

En el primer capítulo de esta tesis se plantea los objetivos y la hipótesis, la cual es la base de esta investigación, el segundo capítulo contiene conceptos y definiciones necesarios para poder llegar al objetivo principal, el tercer capítulo presenta la construcción del modelo para dar solución al problema, el cuarto capítulo expone los resultados obtenidos mediante un prototipo y finalmente el quinto capítulo contiene las conclusiones y recomendaciones de acuerdo al desarrollo y objetivos del presente trabajo.

ÍNDICE

CAPÍTULO 1: INTRODUCCIÓN

1.1.	ANTECEDENTES.....	2
1.2.	DESCRIPCIÓN DEL PROBLEMA.....	3
1.3.	PLANTEAMIENTO DEL PROBLEMA.....	4
1.3.1.	Enfoque del problema.....	4
1.4.	OBJETIVOS.....	4
1.4.1	Objetivo General.....	4
1.4.2	Objetivos Específicos.....	5
1.5.	HIPOTESIS.....	5
1.5.1.	Identificación de variables.....	5
1.5.2.	Definición de variables.....	5
1.6.	JUSTIFICACION.....	7
1.6.1.	Justificación Tecnológica.....	7
1.6.2.	Justificación Económica.....	7
1.6.3.	Justificación Social.....	7
1.7.	ALCANCES Y LIMITES.....	8
1.7.1	Alcances.....	8
1.7.2	Limites.....	8
1.8.	METODOLOGÍA.....	9
1.8.1.	MÉTODOS Y MEDIOS DE INVESTIGACIÓN CIENTÍFICA.....	9

CAPÍTULO 2: MARCO TEÓRICO

2.1.	EVOLUCION DE LOS ALGORITMOS EVOLUTIVOS.....	10
2.2.	ALGORITMOS GENÉTICOS.....	11
2.2.1.	Introducción.....	11
2.2.2.	Orígenes.....	12
2.2.3.	Terminología.....	13
2.2.4.	Función de los algoritmos genéticos.....	14
2.2.5.	Conceptos básicos.....	15
2.2.5.1.	Codificación.....	15
2.2.5.2.	Población.....	16

2.2.5.3. Función objetivo	16
2.2.5.4. Selección	17
2.2.5.5. Elitismo	18
2.2.6. Reproducción	19
2.2.6.1. Cruzamiento	19
2.2.6.2. Mutación	21
2.2.7. Reemplazo	22
2.2.8. Aplicaciones de los algoritmos genéticos	24
2.3. LA COMPUTACIÓN EVOLUTIVA	25
2.4. MÉTRICAS PARA PRUEBAS ORIENTADAS A OBJETOS	28
2.4.1. Métricas orientadas a la Función	29
2.4.2. Métricas de Halstead	32
2.5. PRUEBA DE HIPÓTESIS	33
2.5.1. Prueba de t student	35
2.6. LENGUAJES DE PROGRAMACIÓN	36
2.6.1. Java	36
2.6.2. C++	39
2.6.3. Visual Basic .NET	41

CAPÍTULO 3: CONSTRUCCION DEL MODELO

3.1 INTRODUCCIÓN	43
3.2 MÉTODO CIENTÍFICO	45
3.3 MÉTODO INFORMÁTICO	46
3.4 IMPLEMENTACIÓN DEL PROTOTIPO	48
3.4.1. Estructura de un cromosoma	50
3.4.2. Población	54
3.4.3. Evaluación	55
3.4.4. Selección	57
3.4.5. Cruzamiento	57
3.4.6. Mutación	60
3.5 TECNOLOGÍA EMPLEADA	60
3.6 INTERFAZ DEL PROTOTIPO	61

CAPITULO 4: PRUEBAS Y RESULTADOS

4.1.	ANÁLISIS DE RESULTADOS.....	62
4.1.1.	Métricas orientadas a la función.....	63
4.2.	PRUEBAS DEL PROTOTIPO.....	66
4.2.1.	Métricas de Halstead.....	68
4.3.	VALIDACIÓN DEL PROTOTIPO.....	70

CAPITULO 5: CONCLUSIONES Y RECOMENDACIONES

5.1.	CONCLUSIONES.....	73
5.2.	RECOMENDACIONES.....	74

BIBLIOGRAFÍA

ANEXOS



INDICE DE FIGURAS

Figura 1.	Estructuras de los cromosomas.....	13
Figura 2.	Cromosoma de 8 genes con valores binarios.....	15
Figura 3.	Cruza en un punto.....	20
Figura 4.	Cruza en dos puntos.....	20
Figura 5.	Cruza uniforme.....	21
Figura 6.	Operador de mutación.....	22
Figura 7.	Métrica de punto Función.....	30
Figura 8.	Etapas de la metodología del AG.....	47
Figura 9.	Representación del gen.....	49
Figura 10.	Representación del cromosoma.....	51
Figura 11.	Diagrama de procesos del AG.....	53
Figura 12.	Interfaz del prototipo.....	61
Figura 13.	Introducción de parámetros.....	66
Figura 14.	Generación de la población inicial.....	67
Figura 15.	Formato de parámetros.....	67
Figura 16.	Horario optimo.....	68
Figura 17.	Prueba del estadístico t student.....	72

INDICE DE TABLAS

Tabla 1.	Matriz Causa – Efecto.....	3
Tabla 2.	Algoritmos genéticos vs. Métodos tradicionales.....	23
Tabla 3.	Ajuste de complejidad.....	31
Tabla 4.	Tabla de sugerencias.....	52
Tabla 5.	Resultados del caso 1.....	63
Tabla 6.	Resultados del caso 2.....	63
Tabla 7.	Calculo de PF sin ajuste.....	65
Tabla 8.	Resultados del prototipo.....	71





1

INTRODUCCIÓN

1. INTRODUCCIÓN

La ciencia y la tecnología han evolucionado basadas en el estudio del comportamiento de la naturaleza, formulando nuevos mecanismos, métodos, técnicas y teorías para solucionar, interpretar y mejorar nuestra relación con el mundo.

La observación de los fenómenos naturales ha permitido generar nuevas alternativas de solución a problemas relacionados con la computación y la matemática. En nuestra investigación utilizaremos la teoría de la evolución de Darwin y la Genética Moderna (se fundamenta en los principios de Darwin) para construir una heurística en la solución de un problema de optimización. *La computación evolutiva* es parte del tipo de investigaciones mencionadas, surge como una rama de la Inteligencia Artificial y con ella, la naturaleza junto con la estructura genética de sus pobladores encuentra soluciones a problemas comunes.

De acuerdo a las teorías de Darwin, con el paso de las generaciones las poblaciones evolucionan según criterios de selección natural y la supervivencia de un individuo esta dada por el grado de adaptación con su entorno.

Los Algoritmos Genéticos son técnicas de búsqueda guiadas hacia un conjunto de soluciones usando diferentes medios tales como la selección natural y operadores genéticos, los cuales a diferencia de las técnicas basadas en cálculos como el de Newton, funcionan bajo características dadas en la teoría de la evolución donde solo las mejores soluciones sobreviven y mejoran las cualidades de la población. Como se plantea esta investigación, los *Algoritmos Genéticos* permiten construir una solución al problema de Designación de Infraestructura Académica basado en operadores genéticos y en el mejoramiento de los resultados obtenidos en su aplicación.

El interés inicial de esta investigación se centra en el estudio, la comprensión y correcta utilización de un método heurística denominado Algoritmos Genéticos, el cual con bases evolutivas encierra muchos aspectos de investigación al ser un mecanismo que contiene diversas ventajas y propiedades en la solución de problemas de búsqueda y

optimización, involucrando una serie de parámetros que se adaptan muy propiamente a cada espacio con el fin de mejorar el proceso en forma evolutiva.

1.1. ANTECEDENTES

Hace tiempo, las universidades e instituciones solo contaban con carreras muy específicas y estas a su vez con pocas materias y pocos alumnos, por lo que no se requería un estudio de búsqueda mas optima de soluciones con referencia a la designación de la infraestructura. Con el paso del tiempo y el desarrollo gigantesco de la tecnología como tanto del conocimiento, las instituciones van en aumento en las disciplinas educativas, por lo que están en la obligación de ampliar su infraestructura y la malla curricular. En ese entonces, el estudio de designación o asignación era realizado sobre estándares ya definidos, el cual era llenada manualmente de acuerdo a las normas y reglas ya reglamentadas dentro de la institución.

Actualmente, muchas instituciones cuentan con pequeños procedimientos o sistemas de designación, que ayuda realizar o sacar informes de estas designaciones como ser hojas de cálculo (Excel) o programas que ayudan a resolver de alguna manera estos problemas.

Hay pocas instituciones que se preocupan para dar solución a este problema de designación de infraestructura y en el caso, de que lo hacen, utilizan métodos tradicionales de designación y búsqueda, es decir, no buscan un estudio mas optimo de solución que ayude a mejorar el problema.

En nuestro medio, en la carrera de informática se desarrollo un proyecto de grado que trata del mismo tema, pero con un enfoque relacionado con base de datos y utilizando un método de búsqueda tradicional, denominado: "Sistema automatizado de asignación de aulas para la carrera de Informática en base a un modelo de proporción lineal" de David Calle Acarapi.

El tema de tesis se centra en el estudio, comprensión y correcta utilización del método heurística denominado Algoritmos Genéticos enfocando una metodología de búsqueda más eficiente y óptimo que se necesita para designar aulas.

1.2. DESCRIPCION DEL PROBLEMA

En la siguiente tabla se detalla las causas y efectos de cada problema identificado:

MATRIZ CAUSA – EFECTO

	PROBLEMA	CAUSA	EFECTO	SOLUCION
1	Frecuentes choques de horarios.	Mala manipulación de los recursos.	Perjuicio de los Estudiantes.	Obtener todas las posibles formas de designar un horario apto para los estudiantes.
2	La desmotivación de los estudiantes con respecto a la carrera.	Mala distribución de ambientes y/o incumplimiento de parte de los Docentes.	Estudiantes con malos conocimientos y mala opinión de la Universidad y la carrera.	Un control más efectivo del plantel docente. Una adecuación de los ambientes.
3	La Mala distribución de aulas.	El plantel administrativo no manipula los ambientes en forma efectiva.	Suspensión de clases y estudiantes perjudicados.	Crear un software para una designación optima de aulas.
4	Ambientes cerrados y pequeños.	Falta de capacidad y/o información sobre la cantidad de alumnado.	Incomodidad en los alumnos y docentes. Mala captación de lo enseñado.	Con el software se podrá evitar esos contratiempos, logrando la satisfacción del estudiante como del docente.
5	Falta de coordinación y/o planificación de parte de los Docentes y Dirección de carrera.	Docentes y jefes de carrera no definen un plan de gestión adecuada.	El constante desacuerdo entre docentes, plantel Administrativo y dirección de carrera.	El software que se diseñará podrá ser capaz de establecer formas de planificación de gestión académica.
6	El jefe de carrera desconoce los distintos problemas que afrontan los estudiantes.	Mala administración de parte del jefe de carrera.	El jefe de carrera puede tomar las decisiones con respecto a los ambientes que van a ser designados.	Con el software se podrá especificar que cada ambiente sea la más adecuada para el estudiante sin ninguna preocupación.

Tabla 1. Matriz Causa- Efecto

1.3. PLANTEAMIENTO DEL PROBLEMA

1.3.1. Enfoque del problema

Existen diversos escenarios posibles, de los cuales se puede plantear el estudio o análisis de cómo designar aulas a todas las materias que engloba a la carrera o institución, sin ocasionar perjuicios entre ellos. Analizar el aula en sus diferentes estados como ser; en un día y sus horas de uso, este análisis es muy necesario y determinante ya que una mala designación genera conflictos, insatisfacciones y pérdida de tiempo tanto en docentes como en los alumnos.

Por lo que, el principal problema que se observa es la mala distribución de aulas, choque de horarios y otros, lo cual puede ser mejorado utilizando un planteamiento científico o simplemente herramientas que ayuden a la solución de una búsqueda dinámica para la designación de aulas. Ya que, en algunas instituciones la designación de aulas es realizada manualmente, haciendo uso de métodos de búsquedas tradicionales (búsqueda secuencial a simple vista).

Para una solución óptima a este problema de designación de aulas es necesario representar o dar un enfoque de un modelo matemático, en cual se identificarán aulas, días y horas a utilizarse en un determinado espacio áulico o infraestructura, es decir que se tomará en cuenta un aula a la cual se le aplicarán los distintos tipos de operadores y procesos, tratados en la investigación. Todo esto con el fin de ubicar a una materia en un aula con su respectivo día y hora a utilizarse.

1.4. OBJETIVOS

1.4.1. Objetivo General.

Desarrollar un modelo óptimo y eficiente para una mejor designación de infraestructura académica, utilizando Algoritmos Genéticos.

1.4.2. Objetivos Específicos.

- Mostrar la aplicabilidad de los Algoritmos Genéticos en el proceso de distribución de ambientes;
- Proporcionar una base teórica fundamental en el proceso de designación;
- Enfocar un modelo de búsqueda de soluciones distinto a las tradicionales;
- Realizar un estudio claro sobre Algoritmos Genéticos;
- Desarrollar un prototipo en el que se enfoque el comportamiento de los Algoritmos Genéticos y los resultados;
- Evaluar el método propuesto mediante la construcción de un prototipo de prueba que permita una correcta localización de espacios áulicos.

1.5. HIPOTESIS

Hi: La designación de infraestructura académica es mejorada a través de un modelo con algoritmos genéticos, permitiendo una correcta localización y designación de espacios áulicos.

1.5.1. Identificación de variables

- **Las unidades de observación:** La infraestructura académica;
- **Las variables:**

Variable Independiente: Modelo de designación con algoritmos genéticos;

Variable Dependiente: Espacios áulicos;

- **Término de relación:** Es mejorada, correcta localización y designación.

1.5.2. Definición de variables

La Variable Independiente esta compuesta por *Desig*, en esta variable es la cual se basa el modelo de designación con algoritmos genéticos, en la cual se tomará

en cuenta las aulas, días y horas, estos junto a los operadores genéticos determinarán un espacio áulico en la cual será designada una determinada materia.

$$Desig = \{(a_i, d_j, h_k) | a_i \in Au, d_j \in Ds, h_k \in Hrs\}$$

La *Variable Dependiente* esta compuesta por: las aulas, días, horas y los operadores de los algoritmos genéticos, las cuales interactuaran entre sí, permitiendo una correcta localización o designación de un determinado espacio áulico.

Donde:

Au, Ds y Hrs son tres espacios vectoriales en R^1 .

$$\text{Aula } Au = (a_1, a_2, \dots, a_i, \dots, a_N)$$

$$\text{Días } Ds = (d_1, d_2, \dots, d_j, \dots, d_D), D \leq 6$$

$$\text{Horas } Hrs = (h_1, h_2, \dots, h_k, \dots, h_M)$$

Aplicando funciones y operadores genéticos:

Evaluación P ($Desig_p$), Selección P ($Desig_p$),

Cruzamiento P ($Desig_p, Desig_q$); $p \neq q$ y Mutación P ($Desig_p$)

De esta manera el AG² encontrará un individuo del grupo de individuos analizados, esto se realizará mediante una función de aptitud, el cual estará en función a la cantidad de generaciones, de esta se seleccionará el mejor de ellos el cual será un candidato para la solución óptima al problema.

¹ Números Reales con los tres ejes de coordenadas X, Y, Z.

² AG, Algoritmo Genético, y la función de aptitud se basan en el número de generaciones

1.6. JUSTIFICACIÓN

1.6.1. Justificación Tecnológica

El problema de la designación es de gran interés, ya que la tecnología y la infraestructura en diferentes institutos van aumentando como también el alumnado, es por eso que se utilizará un modelo científico y eficaz en la búsqueda de una distribución de aulas.

La comunidad tecnológica y científica necesita novedosos campos de aplicación a tendencias que de alguna manera están de moda en el mundo de la Informática.

Un sistema de este tipo desde el punto de vista aplicativo es muy valioso puesto que en su desarrollo se hace uso de conocimientos en las áreas de la programación orientada a objetos, la matemática, manejo de estructura de datos y de almacenamiento, su estudio y desarrollo, amplían y fortalecen los conocimientos sobre estas y otras áreas relacionadas.

1.6.2. Justificación Económica

Este trabajo de investigación enfocara una metodología en cual se podrá contar en los distintos ambientes educativos, con cual se optimizará el tiempo como el costo de materiales para una distribución de infraestructura académica. No solamente será capaz de designar aulas sino definirá el horario.

1.6.3. Justificación Social

En la actualidad muchas son las operaciones realizadas en cuanto a la designación de aulas en las distintas instituciones, como la tecnología y la ciencia han aumentado considerablemente, es muy necesario en la sociedad el uso de una herramienta capaz de optimizar y agilizar la distribución de espacios áulicos.

El presente trabajo puede ayudar a todas las instituciones educativas que deseen agilizar y optimizar la distribución de los ambientes, sin el temor de ocasionar choques de horarios, y perjuicios a los alumnos y docentes.

1.7. ALCANCES Y LIMITES

1.7.1. Alcances

- El interés inicial de esta investigación se centra en el estudio, la comprensión y correcta utilización de los *Algoritmos Genéticos* como una alternativa de solución al problema de designación;
- En el desarrollo de la investigación se espera analizar el comportamiento de la distribución de aulas, así como las diferentes modificaciones o cambios que el algoritmo genético pueda tener en el proceso de búsqueda de una solución viable, efectiva y óptima, tanto en los datos como en los procesos y resultados.

1.7.2. Limites

- El software con Algoritmos Genéticos solo abarcará todo lo que es la infraestructura de una determinada institución educativa (aulas y laboratorios), para lo cual se hará un estudio minucioso de un determinado ambiente;
- El estudio de los Algoritmos Genéticos en el presente trabajo se basa solamente a designar o asignar un determinado ambiente (aula), con su respectivo día, hora y materia;
- La materia que tenga paralelos se tomará como una materia única e independiente, es decir tendrá su propio código de entrada. Además para este problema, no se tomará en cuenta la capacidad del aula, ni la cantidad de alumnos y tampoco a los docentes designados a cada materia.

1.8. METODOLOGÍA

El desarrollo de la presente tesis se apoya en el Método Científico, en la metodología Explorativa, descriptiva y la metodología de los Algoritmos Genéticos que sirve de guía en la organización de todo el proceso de investigación, el mismo que llegará a cubrir los requerimientos necesarios para que los objetivos planteados se lleguen a cumplir.

1.8.1. METODOS Y MEDIOS DE INVESTIGACION CIENTIFICA

El método científico es el camino planeado o la estrategia que se sigue para descubrir las propiedades del objeto de estudio.

El método científico es un proceso de razonamiento que intenta no solamente describir los hechos sino también explicarlos, es por esta razón, que se sugiere lo siguiente para la elaboración del presente trabajo:

- **Análisis y Síntesis:** El análisis en términos generales se refiere a la descomposición de un todo en sus distintos elementos, con el fin de estudiarlos de manera separada, luego en un proceso de síntesis se debe integrar todos estos elementos. En el presente trabajo se realiza el análisis de los diferentes criterios que intervienen en la designación y tratamiento de los espacios áulicos. La síntesis se la realiza cuando se realice la construcción de la función objetivo, los operadores genéticos y en la infraestructura académica de una determinada institución;
- **Inducción y Deducción:** La inducción se refiere a la generalización de una observación, razonamiento o conocimiento establecido a partir de casos particulares. En este trabajo se induce que si el Algoritmo Genético realiza el análisis de los diferentes ambientes, entonces también podrá realizar lo mismo en casos generales. La deducción es la aplicación de teorías genéricas a situaciones. La deducción se aplica cuando el Algoritmo Genético deduzca (empleando los operadores genéticos) el espacio áulico mas optimo;
- **Observación:** Se recopila la teoría de la designación de un determinado ambiente dentro de la infraestructura académica, así como el conocimiento de los diferentes ambientes.



2

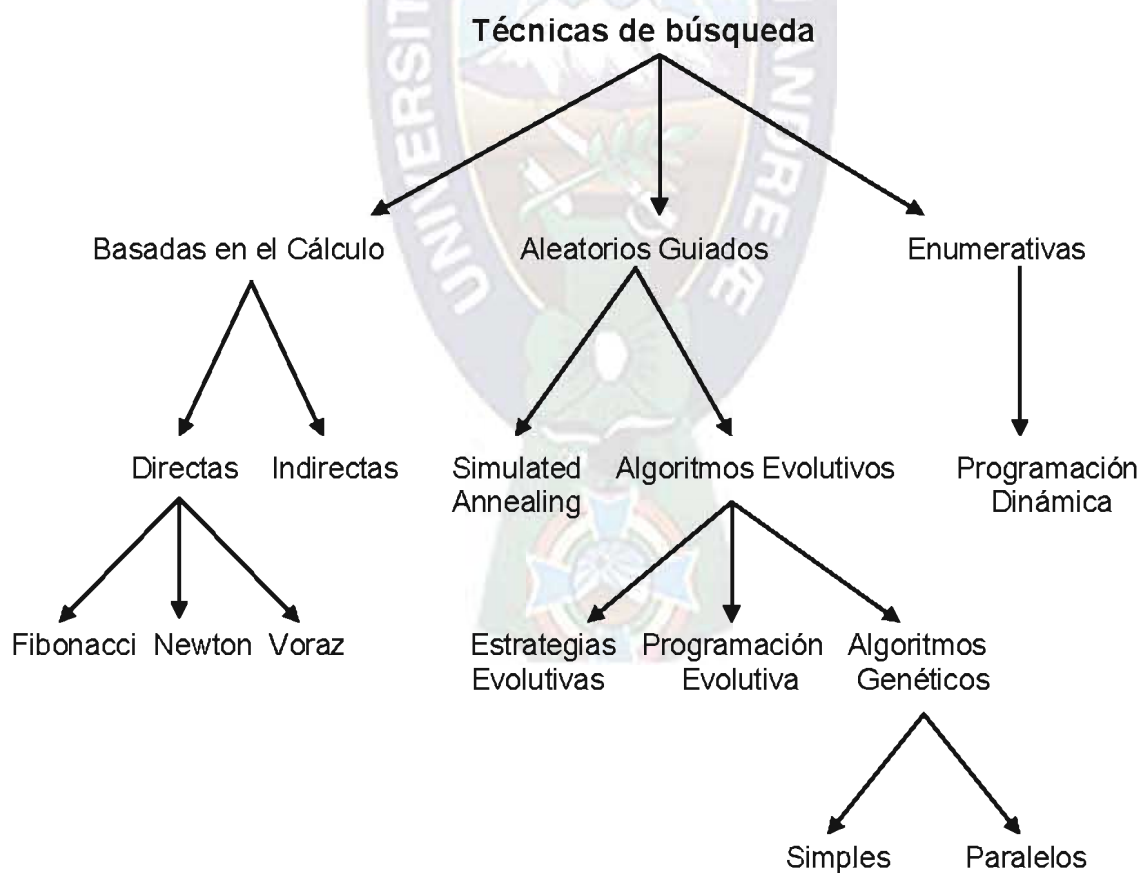
MARCO TEÓRICO

2. MARCO TEÓRICO

2.1. EVOLUCIÓN DE LOS ALGORITMOS GENÉTICOS.

Con la evolución de la tecnología, la programación también ha evolucionado, aplicando en su campo las diferentes áreas que existen de las cuales se abstraen objetos o entidades, las mismas que son programadas en un determinado lenguaje de programación. Por lo que las ciencias se han mezclado con los seres vivos, dando a lugar a la programación evolutiva, a la inteligencia artificial y a la bioprogramación.

Con esta evolución se trata de simular el comportamiento humano en todos sus aspectos, buscando técnicas en las distintas áreas, como se puede observar en el siguiente esquema.



Tipos de Algoritmos Evolutivos [González, 2003]

En este tema solo se abordará a los algoritmos genéticos simples ya que son el resultado de varias evoluciones o transformaciones, en otras palabras resulta la mejor técnica de búsqueda como se observa en el esquema.

2.2. ALGORITMOS GENÉTICOS

2.2.1. Introducción

Los Algoritmos Genéticos son métodos adaptativos que pueden usarse para resolver problemas de búsqueda y optimización. Están basados en el proceso genético de los organismos vivos. A lo largo de las generaciones, las poblaciones evolucionan en la naturaleza acorde con los principios de la selección natural y la supervivencia de los más fuertes, postulados por Darwin.

Por imitación de este proceso, los Algoritmos Genéticos son capaces de ir creando soluciones para problemas del mundo real. La evolución de dichas soluciones hacia valores óptimos del problema depende en buena medida de una adecuada codificación de las mismas. [John Holland, 1999]

Un **algoritmo genético** consiste en una función matemática o una rutina de software que toma como entradas a los ejemplares y retorna como salidas cuales de ellos deben generar descendencia para la nueva generación.

Versiónes más complejas de algoritmos genéticos generan un ciclo iterativo que directamente toma a la especie (el total de los ejemplares) y crea una nueva generación que reemplaza a la antigua una cantidad de veces determinada por su propio diseño. Una de sus características principales es la de ir perfeccionando su propia heurística en el proceso de ejecución, por lo que no requiere largos períodos de entrenamiento especializado por parte del ser humano, el principal defecto de otros métodos para solucionar problemas es la funcionalidad, como los Sistemas Expertos.

Según la ACM los AG son una herramienta muy poderosa de optimización que puede ser usada para resolver un gran número de problemas difíciles con gran eficiencia y exactitud, basándose en la genética natural y la teoría de la evolución de Darwin.

2.2.2. Orígenes

El algoritmo genético es una técnica de búsqueda basada en la teoría de la evolución de Darwin, que ha cobrado tremenda popularidad en todo el mundo durante los últimos años. Se presentarán aquí los conceptos básicos que se requieren para abordarla, así como unos sencillos ejemplos que permitan a los lectores comprender cómo aplicarla al problema de su elección.

En los últimos años, la comunidad científica internacional ha mostrado un creciente interés en una nueva técnica de búsqueda basada en la teoría de la evolución y que se conoce como el **algoritmo genético**. Esta técnica se basa en los mecanismos de selección que utiliza la naturaleza, de acuerdo a los cuales los individuos más aptos de una población son los que sobreviven, al adaptarse más fácilmente a los cambios que se producen en su entorno. Hoy en día se sabe que estos cambios se efectúan en los genes de un individuo (unidad básica de codificación de cada uno de los atributos de un ser vivo), y que sus atributos más deseables (i.e., los que le permiten adaptarse mejor a su entorno) se transmiten a sus descendientes cuando éste se reproduce sexualmente.

Un investigador de la Universidad de Michigan llamado John Holland era consciente de la importancia de la selección natural, y a fines de los 60s desarrolló una técnica que permitió incorporarla a un programa. Su objetivo era lograr que las computadoras aprendieran por sí mismas. A la técnica que inventó Holland se le llamó originalmente “planes reproductivos”, pero se hizo popular bajo el nombre “algoritmo genético” tras la publicación de su libro *Adaptation in Natural and Artificial System* en 1975.

Una definición bastante completa de un algoritmo genético es la propuesta por John Koza:

“Es un algoritmo matemático altamente paralelo que transforma un conjunto de objetos matemáticos individuales con respecto al tiempo usando operaciones modeladas de acuerdo al principio Darwiniano de reproducción y supervivencia del más apto, y tras

haberse presentado de forma natural una serie de operaciones genéticas de entre las que destaca la recombinación sexual. Cada uno de estos objetos matemáticos suele ser una cadena de caracteres (letras o números) de longitud fija que se ajusta al modelo de las cadenas de cromosomas, y se les asocia con una cierta función matemática que refleja su aptitud.”

2.2.3. Terminología

El concepto básico que se usa en los AG es que se basa en la búsqueda de mejores soluciones de problemas, de la misma forma en que las especies evolucionan a fin de adaptarse mejor a su habitat. Al igual que la biogenética, este proceso ocurre de forma iterativa y evolucionan a lo largo del tiempo. Por lo tanto, la jerga de los AG a menudo habla de la analogía de la supervivencia del más apto. [Jesús Alfonso López, 2000].

Con respecto a La Programación los Algoritmos Genéticos son aplicados a programas. La Programación Genética es más expresiva que las cadenas de bits de longitud fija de los AGs³, aunque los AGs pueden ser más eficientes para algunas clases de problemas.

Gen (Gen)- Partícula de los cromosomas que producen la aparición de caracteres hereditarios.

Cromosoma- Más o menos por la misma época, el biólogo alemán Walther Flemming describió los cromosomas, como ciertos filamentos en los que se agregaba la cromatina del núcleo celular durante la división; poco más adelante se descubrió que las células de cada especie viviente tenía un número fijo y característico de cromosomas, véase (*Figura 1*).



Figura 1. Estructuras de los cromosomas
[Guervós, 2003]

Evolución (Evolution)- Serie de transformaciones sucesivas que han sufrido los seres vivos desde los tiempos geológicos.

³ Algoritmos Genéticos, referente a la programación genética

2.2.4. Función de los algoritmos genéticos

Los algoritmos genéticos según Martín Pelikan et al, pasan por cuatro etapas primarias en su ciclo la solución del problema. Las mismas que se pueden comparar con los procesos evolutivos que ocurren con las especies del mundo real. Estas son:

- Creación de una población de soluciones candidata;
- Evaluación de cada solución;
- Selección de las soluciones sobre la base de su aptitud;
- Reproducción de soluciones utilizando operadores genéticos.

Según P. Larrañaga, las funciones y operaciones de un AG simple son las que se ilustran en el siguiente pseudo-código:

```

BEGIN /* Algoritmo Genético Simple */
    Generar una población inicial.
    Computar la función de evaluación de cada individuo.
    WHILE NOT Terminado DO
        BEGIN /* Producir nueva generación */
            FOR Tamaño población/2 DO
                BEGIN /*Ciclo Reproductivo */
                    Seleccionar dos individuos de la anterior generación,
                    para el cruce (probabilidad de selección proporcional
                    a la función de evaluación del individuo).
                    Cruzar con cierta probabilidad los dos
                    individuos obteniendo dos descendientes.
                    Mutar los dos descendientes con cierta probabilidad.
                    Computar la función de evaluación de los dos
                    descendientes mutados.
                    Insertar los dos descendientes mutados en la nueva generación.
                END
            END
            IF la población ha convergido THEN
                Terminado:= TRUE
            END
        END
    END
END

```

Pseudo-código del Algoritmo Genético Simple [Larrañaga, 2000]

Como se observa en el pseudo-código, se necesita una codificación o representación del problema, que resulte adecuada al mismo. Además se requiere una función de ajuste o adaptación al problema, la cual designa un número real a cada posible solución codificada. Durante la ejecución del algoritmo, los padres deben ser seleccionados para la reproducción, a continuación dichos padres seleccionados se cruzarán generando dos hijos, sobre cada uno de los cuales actuara un operador de mutación. El resultado de la combinación de las anteriores funciones será un conjunto de individuos (posibles soluciones al problema), los cuales en la evolución del Algoritmo Genético formaran parte de la siguiente población.

2.2.5. Conceptos básicos

Antes de continuar con la explicación de los AGs, definiremos algunas **terminologías claves** utilizados por los investigadores y los desarrolladores de aplicaciones con AGs.

2.2.5.1. Codificación

La representación de las soluciones candidatas en los AG es mediante una cadena de valores paramétricos (binario, alfabéticos). Si bien el alfabeto utilizado para representar los individuos no necesariamente tiene que estar constituido por (0,1). En términos biológicos, el conjunto de parámetros representando un cromosoma particular se denomina fenotipo. Un gen se identifica por la posición que ocupa en el cromosoma la cual representa un atributo del individuo, el cual puede tomar diferentes valores. Como se observa en la **Figura2**.

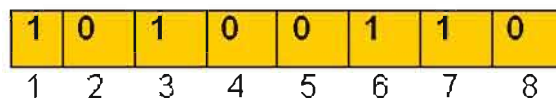


Figura 2. Cromosoma de 8 genes con valores binarios [Larrañaga, 2000]

Los cromosomas se conocen como individuos específicos de una población dada. Un cromosoma es un arreglo de genes, cuya cantidad es fija e igual para todos los individuos de la población.

Una de las ideas más importantes es definir estructuras *admisibles* en sentido que estén bien definidas y puedan ser evaluadas.

2.2.5.2. Población

En cuestión del tema computacional, un AG mapea un determinado problema en un conjunto de cadenas (cromosomas), hay que tomar muy en cuenta el tamaño de la población dado que esto influye de gran manera en la representación de la solución potencial o candidata.

Según la teoría investigada por Goldberg, que las poblaciones de menor tamaño son mas factibles y eficientes soluciones, pero carecen de cubrir el total del espacio de búsqueda, en otras palabras reducen su espacio.

Mientras que el trabajar con poblaciones de mayor tamaño pueden acarear problemas con el costo excesivo computacional. En conclusión el tamaño optimo de la población crece exponencialmente según la longitud de cromosomas. [Gastón Crevillén y David Díaz, 2001]

La población siempre se escoge generando poblaciones al azar, o se crea una población inicial de soluciones potenciales para un problema en particular.

2.2.5.3. Función Objetivo

En el comportamiento de los AG son dos aspectos que resultan cruciales, una determinación de una adecuada función de adaptación o función objetivo y la codificación utilizada.

La función objetivo es la base que determina la probabilidad de la sobre vivencia. Esta función debe ser diseñada de acuerdo a cada problema de manera específica.

La regla, general para construir una buena función objetivo es que ésta debe reflejar el valor del individuo de una manera "real", pero en muchos problemas de optimización

combinatoria, donde existe gran cantidad de restricciones, buena parte de los puntos del espacio de búsqueda representan individuos no válidos. Se tiene que tener un balance entre una función que haga diferencias muy grandes y diferencias pequeñas.

2.2.5.4. Selección

Una parte fundamental del funcionamiento de un AG es, sin lugar a dudas, el proceso de selección de candidatos a reproducirse. En el algoritmo genético este proceso de selección suele realizarse de forma probabilística (es decir, aun los individuos menos aptos tienen la posibilidad de sobrevivir), a diferencia de las estrategias evolutivas, en las que la selección es **extintiva** (los menos aptos tienen cero probabilidades de sobrevivir).

Las técnicas de selección usadas en algoritmos genéticos pueden clasificarse en tres grandes grupos:

- Selección proporcional;
- Selección mediante torneo;
- Selección de estado uniforme.

Selección proporcional: Este nombre a un grupo de esquemas de selección originalmente propuestas por Holland en las que se eligen a los individuos de acuerdo a su contribución de aptitud con respecto al total de población.

Suelen considerarse cuatro grandes grupos dentro de la selección proporcional:

- **La Ruleta:** Consiste en crear una ruleta en la que cada cromosoma tiene asignada una fracción proporcional a su aptitud, luego se genera un número aleatorio simple r con el que se asigna un área mayor de la ruleta. De esta manera se determina que individuos son seleccionados;
- **Sobrante Estocástico:** Propuestas por Booker y Brindle como una alternativa para aproximarse a los valores esperados ($Valesp$) de los individuos;

$$Valesp_i = f_i / f$$

La idea principal es asignar determinísticamente las partes enteras de los valores esperados para cada individuo, luego usar otro esquema (proporcional) para la parte fraccionaria.

El sobrante estocástico reduce los problemas de la ruleta, pero puede causar convergencia prematura al introducir una mayor presión de selección.

- **Universal Estocástica:** Propuesta por Baker con el objetivo de minimizar la mala distribución de los individuos en la población en función de los valores esperados;
- **Muestreo Determinístico:** Es una variante de la selección proporcional con la que experimento DeJong. Es similar al sobrante estocástico, pero requiere un algoritmo de ordenación.

Selección Mediante Torneo: En esta selección se escogen grupos de individuos y de cada grupo se selecciona como padre al mejor, es decir el que gana ese torneo. El tamaño de los grupos entre los que es muy pequeño comparado con el tamaño de la población.

Selección de Estado Uniforme: En esta selección los padres se eligen aleatoriamente entre la generación actual. Todos los individuos de la población, independiente de su aptitud, tienen equiprobabilidad de ser elegido. Esta técnica no garantiza que la población tienda a mejores soluciones puesto que no discrimina los individuos con los genes malos.

2.2.5.5 Elitismo

En el modelo de selección elitista se fuerza a que el mejor individuo de la población en un determinado tiempo, sea seleccionado como padre. Esto significa que no todos los individuos se cruzan o mutan, si no que habrán algunas que pasaran intactas a la

siguiente generación o a las distintas generaciones hasta que surja otro individuo mejor que él, que lo reemplace. Muchos investigadores han encontrado en el elitismo significativamente un progreso en el desarrollo de los AGs. Este procedimiento es opcional.

2.2.6. Reproducción

En esta fase de reproducción, se seleccionan una cantidad de individuos de la población para ser recombinados, los que forman descendientes que luego continuarán la siguiente generación. Los padres serán seleccionados al azar, usando un método que favorece a los individuos mejor adaptados. Después de ser escogidos los padres, sus cromosomas se mezclan y combinan usando operadores genéticos: Cruzamiento y Mutación. [Jesús Alfonso López, 2000].

2.2.6.1 Cruzamiento

El operador de cruzamiento se basa en generar dos nuevos hijos a partir de dos cadenas de cromosomas padre. Este cruzamiento de dos cromosomas se asemeja a la reproducción sexual de las especies, ya que se realiza el intercambio de información digital contenida en los genes [Gastón Crevillén y David Díaz, 2001], el cual permite que las próximas generaciones hereden sus características. La probabilidad de cruzamiento aplicada debe ser mayor que la probabilidad de mutación, en caso contrario el cruzamiento será erróneo.

Comenzaremos las tres técnicas básicas de cruce:

Cruzamiento en un punto: Este tipo de cruce se basa en un punto para lo cual se seleccionan dos padres y se corta en la cadena de cromosomas en una posición o punto escogida al azar, para producir dos subcadenas iniciales y dos subcadenas finales. Después se intercambian las subcadenas finales, produciéndose dos nuevos cromosomas completos. Este operador se conoce como operador de cruce basado en un punto. Habitualmente el operador de cruce no se aplica a todos los pares de individuos que han sido seleccionados para emparejarse, sino que se aplica de manera aleatoria, normalmente con una

probabilidad comprendida entre 0.5 y 1.0. En el caso en que el operador de cruce no se aplique, la descendencia se obtiene simplemente duplicando los padres. Véase (*Figura 3*)

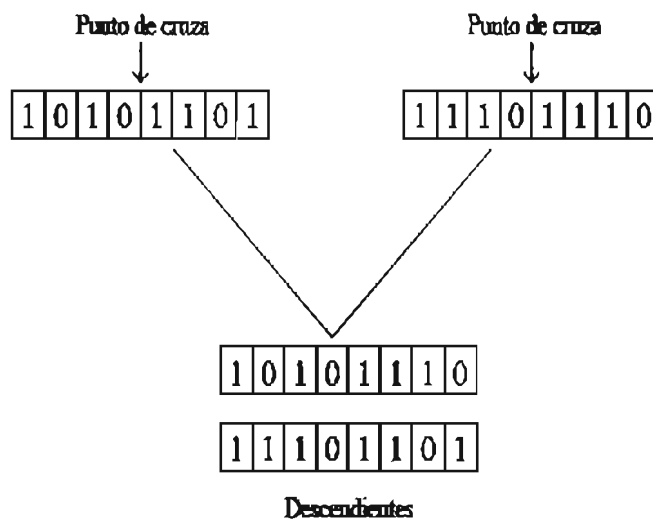


Figura 3. Cruzamiento en un Punto [Coello, 2002]

Cruzamiento en dos Puntos: DeJong fue el primero en implementar una cruce de n puntos, como generalización de la cruce de un punto.

Básicamente es lo mismo que la cruce de un solo punto solo que en esta se corta en dos puntos, como se observa en la *Figura 4*, las cuales se intercambian, formando así a los descendientes.

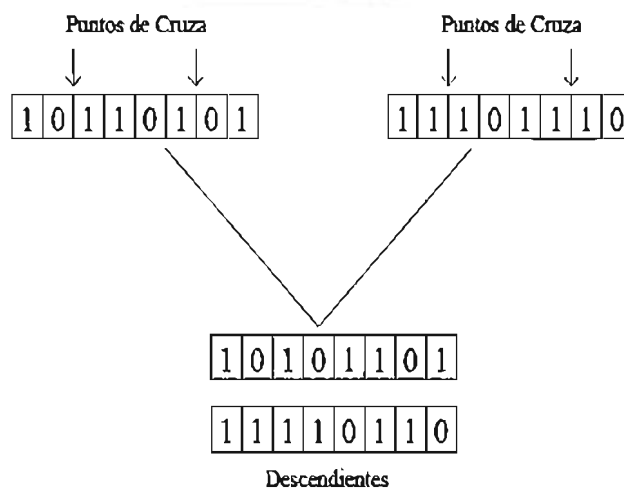


Figura 4. Cruzamiento en dos Puntos [DeJong, 2007]

Cruzamiento Uniforme: Esta técnica fue propuesta básicamente por Achley⁴, aunque suele atribuirse a Syswerda.

En este caso, se trata de una cruce de n puntos, pero en la cual el número de puntos de cruce no se fija previamente.

La cruce uniforme tiene un mayor efecto disruptivo que cualquiera de las dos anteriores. Cuando existe un 1 en la máscara de cruce el gen es copiado del primer padre, mientras que exista un 0 el gen es copiado del segundo padre, como se muestra en la **Figura 5**.

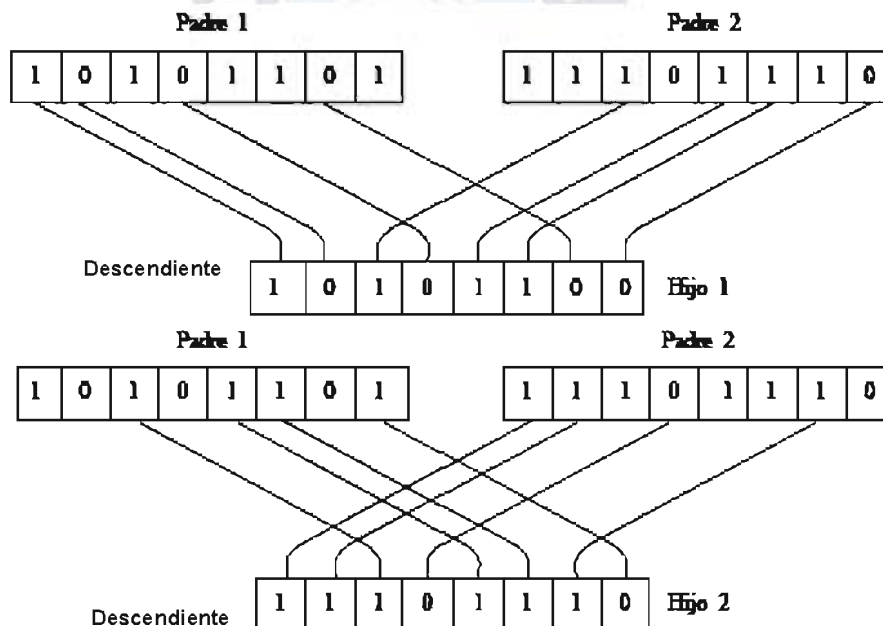


Figura 5. Cruzamiento Uniforme [Mitchell, 1997]

2.2.6.2. Mutación

El operador de mutación se considera como un operador secundario en los AG canónicos. Es decir, que su uso es menos frecuente que el de la cruce.

⁴ Achley M. y Syswerda L. "Científicos de los laboratorios de informática avanzada sobre programación en Algoritmos Genéticos"

En la práctica, suelen recomendar porcentajes de mutación de entre 0.001 y 0.01 para la representación binaria.

Algunos investigadores, sin embargo, han sugerido usar porcentajes altos de mutación al inicio de la búsqueda, y luego decrementarlos exponencialmente, esto favorece al desarrollo del AG.

Otros autores que $p_m = 1/L$ (donde L es la longitud de cadena cromosómica), es un límite inferior para el porcentaje de mutación.

La mutación es básicamente el intercambio de genes (bits). Como se puede observar en la **Figura 6**.

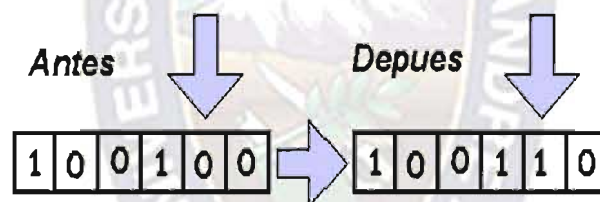


Figura 6. Operador de Mutación [Coello, 2002]

2.2.7. Reemplazo

Una vez obtenidos los i individuos usaremos el mecanismo de reemplazo de individuos para pasar a la próxima generación, como ser:

Reemplazo inmediato: Los descendientes sustituyen a sus propios progenitores.

Reemplazo con factor de llenado: Los i descendientes sustituyen a aquellos miembros de la población que más se aparezcan.

Reemplazo por inserción: Según el tamaño relativo de la descendencia respecto a la población.

$i \leq n$: Se muestra para ser eliminado i miembros de la población progenitora.

Esos miembros serán sustituidos por los descendientes.

$i > n$: Se muestra n miembros de la población de descendientes y se constituye con ellos la nueva población.

Reemplazo por inclusión: Se juntan los i descendientes con los n progenitores en una sola Población, y en ella se muestrean n miembros (los mejores).

La siguiente tabla muestra una breve comparación a modo de resumen las características que poseen los Algoritmos Genéticos, con respecto a otros métodos de optimización. [Goldberg, 1989].

Algoritmos Genéticos	Métodos de Optimización Tradicionales
<ul style="list-style-type: none"> ● Trabajan con parámetros codificados, es decir que deben codificarse como cadenas de longitud finita sobre algún alfabeto finito; ● Utilizan poblaciones de puntos, es decir que usa un conjunto de datos de puntos simultáneamente, de tal forma que la probabilidad de quedar atrapados en óptimos locales se reduce; ● No necesitan conocimientos auxiliares sobre el problema ya que usan información de la función evaluación con respecto a los cromosomas; ● Utilizan reglas de transición 	<ul style="list-style-type: none"> ● Trabajan con los parámetros mismos; ● Operan sobre puntos individuales, ya Sus movimientos en el espacio de búsqueda se hacen de un punto a otro usando reglas de transición determinística. Esto puede ocasionar que se encuentren óptimos locales en lugar de óptimos globales; ● Requieren de mucha información auxiliar para trabajar adecuadamente;

probabilística.	<ul style="list-style-type: none"> • Usan reglas determinística.
-----------------	---

Tabla 2. Algoritmo genético vs. Métodos tradicionales (Fuente: Elaboración propia)

2.2.8. Aplicaciones de los algoritmos genéticos

Optimización: Se trata de un campo especialmente abonado para el uso de los Algoritmos Genéticos, por las características intrínsecas de estos problemas. No en vano fueron la fuente de inspiración para los creadores estos algoritmos. Los Algoritmos Genéticos se han utilizado en numerosas tareas de optimización, incluyendo la optimización numérica, y los problemas de optimización combinatoria.

Programación automática: Los Algoritmos Genéticos se han empleado para desarrollar programas para tareas específicas, y para diseñar otras estructuras computacionales tales como el autómata celular, y las redes de clasificación.

Aprendizaje máquina: Los algoritmos genéticos se han utilizado también en muchas de estas aplicaciones, tales como la predicción del tiempo o la estructura de una proteína. Han servido asimismo para desarrollar determinados aspectos de sistemas particulares de aprendizaje, como pueda ser el de los pesos en una red neuronal, las reglas para sistemas de clasificación de aprendizaje o sistemas de producción simbólica, y los sensores para robots.

Economía: En este caso, se ha hecho uso de estos Algoritmos para modelizar procesos de innovación, el desarrollo estrategias de puja, y la aparición de mercados económicos.

Sistemas inmunes: A la hora de modelizar varios aspectos de los sistemas inmunes naturales, incluyendo la mutación somática durante la vida de un individuo y el descubrimiento de familias de genes múltiples en tiempo evolutivo, ha resultado útil el empleo de esta técnica.

Ecología: En la modelización de fenómenos ecológicos tales como las carreras de armamento biológico, la coevolución de parásito-huésped, la simbiosis, y el flujo de recursos.

Genética de poblaciones: En el estudio de preguntas del tipo “*¿Bajo qué condiciones será viable evolutivamente un gene para la recombinación?*”.

Evolución y aprendizaje: Los Algoritmos Genéticos se han utilizado en el estudio de las relaciones entre el aprendizaje individual y la evolución de la especie.

Sistemas sociales: En el estudio de aspectos evolutivos de los sistemas sociales, tales como la evolución del comportamiento social en colonias de insectos, y la evolución de la cooperación y la comunicación en sistemas multi-agentes. Aunque esta lista no es, en modo alguno, exhaustiva, sí transmite la idea de la variedad de aplicaciones que tienen los Algoritmos Genéticos. Gracias al éxito en estas y otras áreas, los Algoritmos Genéticos han llegado a ser un campo puntero en la investigación actual.

2.3. LA COMPUTACIÓN EVOLUTIVA

La computación Evolutiva o en otras palabras Algoritmo Evolutivo es una técnica de resolución de problemas inspirada en la evolución de los seres vivos. Define una estructura de datos que admita todas las posibles soluciones a un problema.

Solucionar un problema consistirá en encontrar una solución óptima, y por tanto, los Algoritmos Evolutivos son en realidad métodos de Búsqueda ampliada. Es un método especial, en el que las soluciones de un determinado problema son capaces de multiplicarse (reproducirse) entre si, combinando sus características y generando nuevas soluciones. En cada etapa o ciclo se seleccionan las soluciones que mas se acercan a la solución optima buscada, desechando el resto de las soluciones. Las soluciones seleccionadas se reproducirán entre si, permitiendo la mutación o cambio durante dicha reproducción.

El termino Computación Evolutiva es el que agrupa a los Algoritmos Genéticos, la Programación Evolutiva y las Estrategias Evolutivas. En realidad todas estas técnicas son muy parecidas y comparten muchos aspectos, [Herrán, 2002].

La **Programación Evolutiva** (PE), fue desarrollada por Fogel, es una técnica de investigación a través de un espacio de pequeñas maquinas en estado finito. Se utiliza el término, Programación Evolutiva, para todo sistema basado en la evolución.

La programación evolutiva es un algoritmo probabilística que mantiene una población de individuos, $P(t) = \{x_1^t, \dots, x_n^t\}$ para ciclos o iteraciones de t . Cada individuo representa una solución potencial al problema. Cada solución x_i^t es evaluada para medir el nivel de su aptitud o adaptabilidad. Luego una nueva población es formada por selección individual de mejor aptitud, algunos miembros de la población sufren transformaciones por medio de los operadores genéticos que dan una nueva solución. Después de algunas generaciones el programa converge, se espera que el mejor individuo presente la solución óptima más próxima, [Michalewicz, 1995].

La estructura de un programa evolutivo según Michalewicz es el siguiente:

```

Procedure Evolution Program
Begin
  t = 0
  Initialize P(t)
  Evaluate P(t)
  While (not termination condition) do
  begin
    t = t + 1
    select P(t) from P(t-1)
    alter P(t)
    evalate P(t)
  End
End

```

La Programación Evolutiva se diferencia de los AG por la utilización de representaciones específicas mas apropiadas para el dominio de los problemas y porque solo usa el

operador genético de mutación, es decir que la recombinación no se necesita, puesto que las formas de mutación que se utilizan son muy flexibles pueden producir perturbaciones similares a la recombinación si se desea. Después de la inicialización todos los N individuos son seleccionados para ser padres siendo mutados para producir N hijos.

Estos hijos son evaluados junto con los padres y los N de los $2N$ individuos sobreviven utilizando una función probabilística basado en la función de adaptación. [Ucharico, 1999].

Las **Estrategias Evolutivas**⁵ (introducido por Rechenberg) son algoritmos los cuales emitan los principios de la evolución natural como un método para resolver problemas de optimización de parámetros, [Michalewicz, 1995].

Las Estrategias Evolutivas utilizan representaciones con vectores de valores reales. Un par de padres generan un hijo vía de recombinación, el cual posteriormente es perturbado vía mutación. El número de hijos es creado mayor que N , la supervivencia es determinística y se implementa de alguna de las dos maneras: La primera permite la supervivencia de los N hijos y reemplazar los padres con estos N hijos. La segunda permite sobrevivir los N mejores individuos entre padres e hijos. La mutación es el operador más importante porque cada variable puede ser mutada de acuerdo a una distribución de probabilidad. Las diferencias que existen entre las Estrategias Evolutivas y los AG son que en el campo natural las Estrategias Evolutivas su aplicación es de optimización paramétrica, mientras que los AG es la optimización de atributos. La clave de búsqueda de los AG son los operadores genéticos de cruzamiento, en las Estrategias Evolutivas son las de mutación. Por defecto el criterio de reemplazo en los AG es estocástico, en las Estrategias Evolutivas es determinístico. En ambos casos se trata de mantener un nivel de aleatoriedad vs. Determinismo. En las Estrategias Evolutivas el tamaño se fija de antemano, en los AG esta fijo el promedio (por una tasa de cruzamiento). Además en las Estrategias Evolutivas todos los individuos se someten a la mutación, en los AG solo unos individuos se someten a mutación de acuerdo a una frecuencia de mutación [Ucharico, 1999].

⁵ Las estrategias evolutivas, son algoritmos que se basan en el principio de la evolución, esto fue introducido por el investigador Michalewicz, en año 1995.

Los Algoritmos Genéticos son uno de los más novedosos y originales técnicas de resolución de problemas dentro de lo que se ha definido como Computación Evolutiva.

Por lo tanto, centraremos nuestro estudio en los Algoritmos Genéticos, la misma que nos ayudará a comprender, plantear y resolver el modelo propuesto en la presente tesis.

2.4. MÉTRICAS PARA PRUEBAS ORIENTADOS A OBJETOS

El Software Orientado a Objetos (OO) es fundamentalmente distinto del software que se desarrolla utilizando métodos convencionales. Las métricas para sistemas OO deben de ajustarse a las características que distinguen el software OO del software convencional. Estas métricas hacen hincapié en el encapsulamiento, la herencia, complejidad de clases y polimorfismo. Por lo tanto las métricas OO se centran en métricas que se pueden aplicar a las características de encapsulamiento, ocultamiento de información, herencia y técnicas de abstracción de objetos que hagan única a esa clase. Como en todas las métricas los objetivos principales de las métricas OO se derivan del software convencional: comprender mejor la calidad del producto, estimar la efectividad del proceso y mejorar la calidad del trabajo realizado a nivel del proyecto.

La prueba del software es un elemento crítico para la garantía de la calidad del software. El objetivo de la etapa de pruebas es garantizar la calidad del producto desarrollado. Además, esta etapa implica:

- Verificar la interacción de componentes;
- Verificar la integración adecuada de los componentes;
- Verificar que todos los requisitos se han implementado correctamente;
- Identificar y asegurar que los defectos encontrados se han corregido antes de entregar el software al cliente;
- Diseñar pruebas que sistemáticamente saquen a la luz diferentes clases de errores, haciéndolo con la menor cantidad de tiempo y esfuerzo.

La prueba no es una actividad sencilla, no es una etapa del proyecto en la cual se asegura la calidad, sino que la prueba debe ocurrir durante todo el ciclo de vida:

podemos probar la funcionalidad de los primeros prototipos; probar la estabilidad, cobertura y rendimiento de la arquitectura; probar el producto final, etc. Lo que conduce al principal beneficio de la prueba: proporcionar feedback mientras hay todavía tiempo y recursos para hacer algo.

La prueba es un proceso que se enfoca sobre la lógica interna del software y las funciones externas. La prueba es un proceso de ejecución de un programa con la intención de descubrir un error. Un buen caso de prueba es aquel que tiene alta probabilidad de mostrar un error no descubierto hasta entonces. Una prueba tiene éxito si descubre un error no detectado hasta entonces.

La prueba no puede asegurar la ausencia de defectos; sólo puede demostrar que existen defectos en el software.

Para el presente trabajo solo se aplicaran las métricas de prueba de punto función y la segunda prueba de Halstead.

2.4.1. Métricas orientadas a la Función

Son medidas indirectas del software y del proceso por el cual se desarrolla. En lugar de calcularlas las LDC, las métricas orientadas a la función se centran en la funcionalidad o utilidad del programa.

Las métricas orientadas a la función fueron el principio propuestas por Albercht quien sugirió un acercamiento a la medida de la productividad denominado método del punto de función. Los puntos de función que obtienen utilizando una función empírica basando en medidas cuantitativas del dominio de información del software y valoraciones subjetivos de la complejidad del software.

La métrica del punto de función (PF) se puede utilizar como medio para predecir el tamaño de un sistema obtenido a partir de un modelo de análisis. Para visualizar esta métrica se utiliza un diagrama de flujo de datos.

Los puntos de función se calculan rellenando la tabla como se muestra a continuación (ver Figura 7):

Calculo de métricas de punto de función.

Parámetro de medición	FACTOR DE PONDERACIÓN				=	Cuenta
	Cuenta	Simple	Medio	Complejo		
Numero de entradas de usuario	<input type="text"/>	X	3	4	6	<input type="text"/>
Numero de salidas de usuario	<input type="text"/>	X	4	5	7	<input type="text"/>
Numero de peticiones de usuario	<input type="text"/>	X	3	4	6	<input type="text"/>
Numero de archivos	<input type="text"/>	X	7	10	15	<input type="text"/>
Numero de interfaces externas	<input type="text"/>	X	5	7	10	<input type="text"/>
Cuenta = Total	—————→					<input type="text"/>

Figura 7. Métrica de punto de Función

Se determinan 5 características del ámbito de la información y los cálculos aparecen en la posición apropiada de la tabla. Los valores del ámbito de información están definidos de la siguiente manera.

- **Números de entrada de usuario:** se cuenta cada entrada del usuario que proporcione al software diferentes datos orientados a la aplicación. Las entradas deben ser distinguidas de las peticiones que se contabilizan por separado;
- **Número de salida del usuario:** se encuentra cada salida que proporciona el usuario información orientada a la aplicación. En este contexto las salidas se refieren a informes, pantalla, mensajes de error. Los elementos de datos individuales dentro de un informe se encuentran por separado;
- **Números de peticiones al usuario:** una petición esta definida como una entrada interactiva que resulta de la generación de algún tipo de respuesta en forma de salida interactiva. Se cuenta cada petición por separado;
- **Número de archivos:** se cuenta cada archivo maestro lógico, o sea una agrupación lógica de datos que puede ser una parte en una gran base de datos o un archivo independiente;
- **Numero de interfaces externas:** se cuentan todas las interfaces legibles por la maquina por ejemplo: archivos de datos, en cinta o discos que son utilizados para transmitir información a otro sistema.

Cuando han sido recogidos los datos anteriores se asocian el valor de complejidad a cada cuenta. Las organizaciones que utilizan métodos de puntos de función desarrollan criterios para determinar si una entrada es denominada simple, media o compleja. No obstante la determinación de la complejidad es algo subjetivo.

Para calcular los puntos de función se utiliza la siguiente relación.

$$PF = CUENTA_TOTAL * [0.65 + 0.01 * \sum (f_i)]$$

Donde CUENTA _ TOTAL es la suma de todas las entradas de PF obtenidas de la tabla anterior.

f_i donde i puede ser de uno hasta 14 los valores de ajuste de complejidad basados en las respuestas a las cuestiones señaladas de la siguiente tabla.

Evaluar cada factor en escala 0 a 5.

0	1	2	3	4	5
Sin influencia	Incidental	Moderado	Medio	Significativo	Esencial

Tabla 3. Ajuste de complejidad

f_i :

1. ¿Requiere el sistema copias de seguridad y recuperación fiables?
2. ¿Se requiere comunicación de datos?
3. ¿Existen funciones de procesamiento distribuido?
4. ¿Es crítico el rendimiento?
5. ¿Será ejecutado el sistema en un entorno operativo existente y frecuentemente utilizado?
6. ¿Requiere el sistema entrada de datos interactivo?
7. ¿Requiere la entrada de datos interactivo que las transiciones de entrada se llevan acabo sobre múltiples o variadas operaciones?
8. ¿Se actualizan los archivos maestros en forma interactiva?
9. ¿Son complejas las entradas, las salidas, los archivos o peticiones?
10. ¿Es complejo el procesamiento interno?
11. ¿Se ha diseñado el código para se reutilizable?
12. ¿Están incluidos en el diseño la conversión y la instalación?

13. ¿Se ha diseñado el sistema para soportar múltiples instalaciones en diferentes organizaciones?
14. ¿Se ha diseñado la aplicación para facilitar los cambios y fácil uso por el usuario?

Los valores constantes de la ecuación anterior y los factores de peso aplicados en las encuestas de los ámbitos de información han sido determinados empíricamente.

Una vez calculado los puntos de función se usan de forma analógica a las LDC como medida de la productividad, calidad y otros productos del software.

Productividad = PF / persona-mes

Calidad = Errores / PF

Costo = Dólares / PF

Documentación = Pags. Doc / PF

2.4.2. Métricas de Halstead

La teoría de la ciencia del software propuesta por Halstead es probablemente la medida de complejidad mejor conocida y minuciosamente estudiada. La ciencia del software propuso la primera ley analítica y cuantitativa para el software de computadora.

Utiliza un conjunto de medidas primitivas que pueden obtenerse una vez que se han generado o estimado el código después de completar el diseño, entre estas medidas podemos mencionar:

- n1: número de operadores diferentes que aparecen en el programa;
- n2: número de operandos diferentes que aparecen en el programa;
- N1: número total de veces que aparece el operador;
- N2: número total de veces que aparecen el operando.

Halstead utiliza medidas primitivas para desarrollar expresiones para la longitud global del programa; volumen mínimo potencial para un algoritmo; el volumen real (número de bits requeridos para especificar un programa); el nivel del programa (una medida de la complejidad del software); nivel del lenguaje (una constante para un lenguaje dado); y

otras características tales como el esfuerzo de desarrollo, tiempo de desarrollo e incluso el número esperado de fallos en el software.

Las métricas de Halstead, serían:

Longitud global del programa

$$N = n_1 \log_2 n_1 + n_2 \log_2 n_2$$

Volumen del programa

$$V = N \log_2 (n_1 + n_2)$$

Volumen compacto (ya que V varía dependiendo del lenguaje)

$$L = 2 / n_1 \times n_2 / N_2$$

La mayoría de las métricas para pruebas se concentran en el proceso de prueba, no en las características técnicas de las pruebas mismas. En general, los responsables de las pruebas deben fiarse en las métricas de análisis, diseño y código para que sirvan de guía en el diseño y ejecución de los casos de prueba.

El esfuerzo de las pruebas también se puede estimar utilizando métricas obtenidas de las medidas de Halstead. Usando la definición del volumen de un programa, V, y nivel de programa, NP, el esfuerzo de la ciencia del software puede calcularse como:

$$NP = 1 / [(n_1/2) \times (N_2/n_2)]$$

n1: no de operadores diferentes

n2: no de operandos diferentes

N2: no total de Operandos

2.5. PRUEBA DE HIPÓTESIS

Una hipótesis estadística es una suposición hecha con respecto a la función de distribución de una variable aleatoria. Para establecer la verdad o falsedad de una hipótesis estadística con certeza total, será necesario examinar toda la población. En la mayoría de las situaciones reales no es posible o práctico efectuar este examen, y el camino más aconsejable es tomar una muestra aleatoria de la población y en base a ella, decidir si la hipótesis es verdadera o falsa.

En la prueba de una hipótesis estadística, es costumbre declarar la hipótesis como verdadera si la probabilidad calculada excede el valor tabular llamado el nivel de significación y se declara falsa si la probabilidad calculada es menor que el valor tabular. La prueba a realizar dependerá del tamaño de las muestras, de la homogeneidad de las varianzas y de la dependencia o no de las variables.

Si las muestras a probar involucran a más de 30 observaciones, se aplicará la prueba de Z, si las muestras a evaluar involucran un número de observaciones menor o igual que 30 se emplea la prueba de t de student. La fórmula de cálculo depende de si las varianzas son homogéneas o heterogéneas, si el número de observaciones es igual o diferente, o si son variables dependientes.

Para determinar la homogeneidad de las varianzas se toma la varianza mayor y se divide por la menor, este resultado es un estimado de la F de Fisher. Luego se busca en la tabla de F usando como numerador los grados de libertad (n-1) de la varianza mayor y como denominador (n-1) de la varianza menor para encontrar la F de Fisher tabular. Si la F estimada es menor que la F tabular se declara que las varianzas son homogéneas. Si por el contrario, se declaran las varianzas heterogéneas. Cuando son variables dependientes (el valor de una depende del valor de la otra), se emplea la técnica de pruebas pareadas.

Como en general estas pruebas se aplican a dos muestras, se denominarán a y b para referirse a ellas, así entenderemos por:

- na al número de elementos de la muestra a;
- nb al número de elementos de la muestra b;
- \bar{x}_b al promedio de la muestra b;
- s^2_a la varianza de la muestra a;
- Y así sucesivamente.

Entonces se pueden distinguir 6 casos a saber:

1. Caso de muestras grandes ($n > 30$)
2. Caso de $n_a = n_b$ y $s^2_a = s^2_b$
3. Caso de $n_a = n_b$ y $s^2_a <> s^2_b$
4. Caso de $n_a <> n_b$ y $s^2_a = s^2_b$

5. Caso de $n_a \neq n_b$ y $s_a^2 \neq s_b^2$
6. Caso de variables dependientes

2.5.1. Prueba de t student

Técnicamente se puede describir la prueba t de Student como aquella que se utiliza en un modelo en el que una variable explicativa (var. independiente) dicotómica intenta explicar una variable respuesta (var. dependiente) dicotómica. Es decir en la situación: dicotómica explica dicotómica.

La prueba t de Student como todos los estadísticos de contraste se basa en el cálculo de estadísticos descriptivos previos: el número de observaciones, la media y la desviación típica en cada grupo. A través de estos estadísticos previos se calcula el estadístico de contraste experimental. Con la ayuda de unas tablas se obtiene a partir de dicho estadístico el p-valor. Si $p < 0,05$ se concluye que hay diferencia entre los dos tratamientos.

Las hipótesis o asunciones para poder aplicar la t de Student son que en cada grupo la variable estudiada siga una distribución Normal y que la dispersión en ambos grupos sea homogénea (hipótesis de homocedasticidad = igualdad de varianzas). Si no se verifica que se cumplen estas asunciones los resultados de la prueba t de Student no tienen ninguna validez.

Por otra parte no es obligatorio que los tamaños de los grupos sean iguales, ni tampoco es necesario conocer la dispersión de los dos grupos.

La distribución t es continua, tiene forma de campana y es simétrica respecto al cero como la distribución z. La distribución t es dispersa y es más plana en centro que la distribución z, pero se acerca a ella cuando el tamaño de muestra crece.

El estadístico de prueba para el caso de una muestra está dado por:

$$t = \frac{\bar{X} - \mu}{s / \sqrt{n}}$$

Para una prueba de dos colas con la distribución t , se rechaza la hipótesis nula cuando el valor del estadístico de prueba es mayor que

$$t_{n-1, \alpha/2}$$

o si es menor que

$$t_{n-1, \alpha/2}$$

Para una prueba de cola izquierda con la distribución t , se rechaza la hipótesis nula cuando el valor del estadístico de prueba es menor que

$$t_{n-1, \alpha/2}$$

2.6. LENGUAJES PROGRAMACIÓN

Para la implementación del modelo con algoritmos genéticos, se tomará en cuenta tres lenguajes de programación las cuales están Orientados a Objetos de los cuales se escogerá uno de ellos el más conocido y mas apto para el desarrollo del modelo, por lo cual la programación será orientada a objetos con esto se facilitará el manejo de los operadores genéticos.

Cabe hacer notar que estos lenguajes son los más capaces para desarrollar diferentes tipos de aplicaciones, no solamente el de los AGs sino que estos lenguajes se adaptan a cualquier área de estudio.

Además hay que tener en cuenta también los requerimientos de hardware para el lenguaje seleccionado. Los siguientes lenguajes son los más usados en este tipo de aplicaciones, entre los cuales podemos mencionar a los siguientes.

2.6.1. Java

Es un lenguaje de programación orientado a objetos desarrollado por Sun Microsystems a principios de los años 90. El lenguaje en sí mismo toma mucha de su sintaxis de C y C++, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel, que suelen inducir a muchos errores, como la manipulación directa de punteros o memoria.

La implementación original y de referencia del compilador, la máquina virtual y las bibliotecas de clases de Java fueron desarrolladas por Sun Microsystems en 1995. Desde entonces, Sun ha controlado las especificaciones, el desarrollo y evolución del lenguaje a través del Java Community Process, si bien otros han desarrollado también implementaciones alternativas de estas tecnologías de Sun, algunas incluso bajo licencias de software libre.

Entre noviembre de 2006 y mayo de 2007, Sun Microsystems liberó la mayor parte de sus tecnologías Java bajo la licencia GNU GPL, de acuerdo con las especificaciones del Java Community Process, de tal forma que prácticamente todo el Java de Sun es ahora software libre (aunque la biblioteca de clases de Sun que se requiere para ejecutar los programas Java todavía no es software libre).

La primera característica, orientado a objetos (“OO”), se refiere a un método de programación y al diseño del lenguaje. Aunque hay muchas interpretaciones para OO, una primera idea es diseñar el software de forma que los distintos tipos de datos que usen estén unidos a sus operaciones. Así, los datos y el código (funciones o métodos) se combinan en entidades llamadas objetos. Un objeto puede verse como un paquete que contiene el “comportamiento” (el código) y el “estado” (datos). El principio es separar aquello que cambia de las cosas que permanecen inalterables.

Frecuentemente, cambiar una estructura de datos implica un cambio en el código que opera sobre los mismos, o viceversa. Esta separación en objetos coherentes e independientes ofrece una base más estable para el diseño de un sistema software.

El objetivo es hacer que grandes proyectos sean fáciles de gestionar y manejar, mejorando como consecuencia su calidad y reduciendo el número de proyectos fallidos. Otra de las grandes promesas de la programación orientada a objetos es la creación de entidades más genéricas (objetos) que permitan la reutilización del software entre proyectos, una de las premisas fundamentales de la Ingeniería del Software. Un objeto genérico “cliente”, por ejemplo, debería en teoría tener el mismo conjunto de comportamiento en diferentes proyectos, sobre todo cuando estos coinciden en cierta medida, algo que suele suceder en las grandes organizaciones.

En este sentido, los objetos podrían verse como piezas reutilizables que pueden emplearse en múltiples proyectos distintos, posibilitando así a la industria del software a construir proyectos de envergadura empleando componentes ya existentes y de comprobada calidad; conduciendo esto finalmente a una reducción drástica del tiempo de desarrollo. Podemos usar como ejemplo de objeto el aluminio. Una vez definidos datos (peso, maleabilidad, etc.), y su “comportamiento” (soldar dos piezas, etc.), el objeto “aluminio” puede ser reutilizado en el campo de la construcción, del automóvil, de la aviación, etc.

La reutilización del software ha experimentado resultados dispares, encontrando dos dificultades principales: el diseño de objetos realmente genéricos es pobremente comprendido, y falta una metodología para la amplia comunicación de oportunidades de reutilización. Algunas comunidades de “código abierto” (open source) quieren ayudar en este problema dando medios a los desarrolladores para diseminar la información sobre el uso y versatilidad de objetos reutilizables y bibliotecas de objetos.

Sun define tres plataformas en un intento por cubrir distintos entornos de aplicación. Así, ha distribuido muchas de sus APIs (Application Program Interface) de forma que pertenezcan a cada una de las plataformas:

- Java ME (Java Platform, Micro Edition) o J2ME — orientada a entornos de limitados recursos, como teléfonos móviles, PDAs (Personal Digital Assistant), etc.;
- Java SE (Java Platform, Standard Edition) o J2SE — para entornos de gama media y estaciones de trabajo. Aquí se sitúa al usuario medio en un PC de escritorio;
- Java EE (Java Platform, Enterprise Edition) o J2EE — orientada a entornos distribuidos empresariales o de Internet.

Las clases en las APIs de Java se organizan en grupos disjuntos llamados **paquetes**. Cada paquete contiene un conjunto de interfaces, clases y excepciones relacionadas. La

información sobre los paquetes que ofrece cada plataforma puede encontrarse en la documentación de ésta.

El conjunto de las APIs es controlado por Sun Microsystems junto con otras entidades o personas a través del programa JCP (Java Community Process). Las compañías o individuos participantes del JCP pueden influir de forma activa en el diseño y desarrollo de las APIs, algo que ha sido motivo de controversia.

La sintaxis de Java se deriva en gran medida de C++. Pero a diferencia de éste, que combina la sintaxis para programación genérica, estructurada y orientada a objetos, Java fue construido desde el principio para ser completamente orientado a objetos.

Todo en Java es un objeto (salvo algunas excepciones), y todo en Java reside en alguna clase (recordemos que una clase es un molde a partir del cual pueden crearse varios objetos).

2.6.2. C++

Es un lenguaje de programación diseñado a mediados de los años 1980 por Bjarne Stroustrup. La intención de su creación fue el extender al exitoso lenguaje de programación C con mecanismos que permitan la manipulación de objetos. En ese sentido, desde el punto de vista de los lenguajes orientados a objetos, el C++ es un lenguaje híbrido.

Posteriormente se añadieron facilidades de programación genérica, que se sumó a los otros dos paradigmas que ya estaban admitidos (programación estructurada y la programación orientada a objetos). Por esto se suele decir que el C++ es un lenguaje *multiparadigma*.

Una particularidad del C++ es la posibilidad de redefinir los operadores (sobrecarga de operadores), y de poder crear nuevos tipos que se comporten como tipos fundamentales.

C++ permite trabajar tanto a alto como a bajo nivel.

El nombre **C++** fue propuesto por Rick Mascitti en el año 1983, cuando el lenguaje fue utilizado por primera vez fuera de un laboratorio científico. Antes se había usado el

nombre "C con clases". En C++, la expresión "C++" significa "incremento de C" y se refiere a que C++ es una extensión de C.

Los objetos en C++ son abstraídos mediante una Clase. Según el paradigma de la programación orientada a objetos un objeto consta de:

- Métodos o funciones;
- Atributos o Variables Miembro.

Como se notar C++ es un lenguaje orientado a objetos y además de soportar las sobrecargas y herencia. Que a continuación se detalla:

Herencia: Existen varios tipos de herencia entre clases en el lenguaje de programación C++. Estos son:

Herencia Simple.- La herencia en C++ es un mecanismo de abstracción creado para poder facilitar y mejorar el diseño de las clases de un programa. Con ella se pueden crear nuevas clases a partir de clases ya hechas, siempre y cuando tengan un tipo de relación especial.

En la herencia, las clases derivadas "heredan" los datos y la función miembro de las clases base, pudiendo las clases derivadas redefinir estos comportamientos (polimorfismo) y añadir comportamientos nuevos propios de las clases derivadas.

Para no romper el principio de encapsulamiento (ocultar datos cuyo conocimiento no es necesario para el uso de las clases), se proporciona un nuevo modo de visibilidad de los datos/funciones: "protected". Cualquier cosa que tenga visibilidad protected se comportará como pública en la clase Base y en las que componen la jerarquía de herencia, y como privada en las clases que NO sean de la jerarquía de la herencia.

Antes de utilizar la herencia, nos tenemos que hacer una pregunta, y si tiene sentido, podemos intentar usar esta jerarquía: Si la frase <clase Gen> ES-UN <clase Cromosoma> tiene sentido, entonces estamos ante un posible caso de herencia donde clase Cromosoma será la clase base y clase Gen la derivada.

Herencia Múltiple.- La herencia múltiple es el mecanismo que permite al programador hacer clases derivadas a partir, no de una sola clase base, sino de varias.

Sobrecarga de Operadores.- La sobrecarga de operadores es una forma de hacer polimorfismo. Es posible definir el comportamiento de un operador del lenguaje para que trabaje con tipos de datos definidos por el usuario. No todos los operadores de C++ son factibles de sobrecargar, y, entre aquellos que pueden ser sobrecargados, se deben cumplir condiciones especiales. En particular, los operadores sizeof y :: no son sobrecargables.

2.6.3. Visual Basic .NET

Es un lenguaje de programación orientado a objetos que se puede considerar una evolución de Visual Basic implementada sobre el framework .NET. Su introducción resultó muy controvertida, ya que debido a cambios significativos en el lenguaje VB.NET no es compatible hacia atrás con Visual Basic, cosa que causó gran división en la comunidad de desarrolladores de Visual Basic.

La gran mayoría de programadores de VB.NET utilizan el entorno de programación Microsoft Visual Studio .Net en alguna de sus versiones (Visual Studio .NET, Visual Studio .NET 2003 o Visual Studio .NET 2005), aunque existen otras alternativas, como SharpDevelop (que además es libre).

Como pasa con todos los lenguajes de programación basados en .NET, los programas escritos en VB.NET requieren el Framework .NET para ejecutarse.

Para esta versión se añadieron varias novedades, incluyendo:

- Soporte para LINQ (Language Integrated Query)* **Expresiones Lambda**;
- Literales XML.

Si Visual Basic .NET debe considerarse una mera versión de Visual Basic, o si debe considerarse como un nuevo lenguaje de programación es un tema que ha traído mucha discusión, y que aún la trae.

La sintaxis básica es prácticamente la misma entre VB y VB.NET, con la excepción de los añadidos para soportar nuevas características como el control estructurado de excepciones, la programación orientada a objetos, o los Genéricos.

Las diferencias entre VB y VB.NET son profundas, sobre todo en cuanto a metodología de programación y librerías, pero ambos lenguajes siguen manteniendo un gran parecido, cosa que facilita notablemente el paso de VB a VB.NET.

Para desarrollar en VB.NET existen algunas alternativas a Visual Studio, quizás la más notable sea SharpDevelop.

SharpDevelop: SharpDevelop es un entorno de programación integrado que permite programar en C# y en VB.NET.

Este es un entorno publicado bajo licencia LGPL, lo que implica que es libre y que disponemos del código fuente.

MonoDevelop: MonoDevelop es una implementación de SharpDevelop para programar usando Mono, una implementación libre de .NET que funciona en distintos sistemas operativos.

A partir de la introducción en el mercado de la versión 2005 de Visual Studio Microsoft publicó lo que se conoce como ediciones Express de distintos programas. Las versiones Express son versiones limitadas pero gratuitas, pensadas para usos no profesionales.

Visual Basic Express Edition es una versión de Visual Studio limitada. Esta versión permite sólo programar en VB.NET, y además limita el tipo de proyectos que se pueden desarrollar. Visual Web Developer Express Edition permite programar páginas ASP.NET en VB.

Gracias a Mono y al esfuerzo que ha realizado la gente de este proyecto para implementar una versión compatible 100% con .NET que incluye la inmensa mayoría de la librerías y una implementación multiplataforma de Windows.Forms, resulta posible programar para distintos sistemas operativos usando VB.NET.



3

ANÁLISIS Y DISEÑO DEL MODELO

3. ANÁLISIS Y DISEÑO DEL MODELO

3.1. INTRODUCCIÓN

La designación de recursos en cualquier organización deja de ser un problema trivial al menos cuando los recursos son limitados y las necesidades van aumentando. Por lo general, la solución a este tipo de problemas trae aparejada una serie de condiciones de eficiencia, tiempo y oportunidad que deben ser tenidas en cuenta, más allá de que la designación esté correcta. El proceso de designación de recursos de infraestructura aparece cuando hay que realizar cierto número de trabajos o tareas existiendo diversas formas o alternativas de realizarlas, y no se dispone de recursos y medios suficientes para realizar cada actividad de la forma más eficaz.

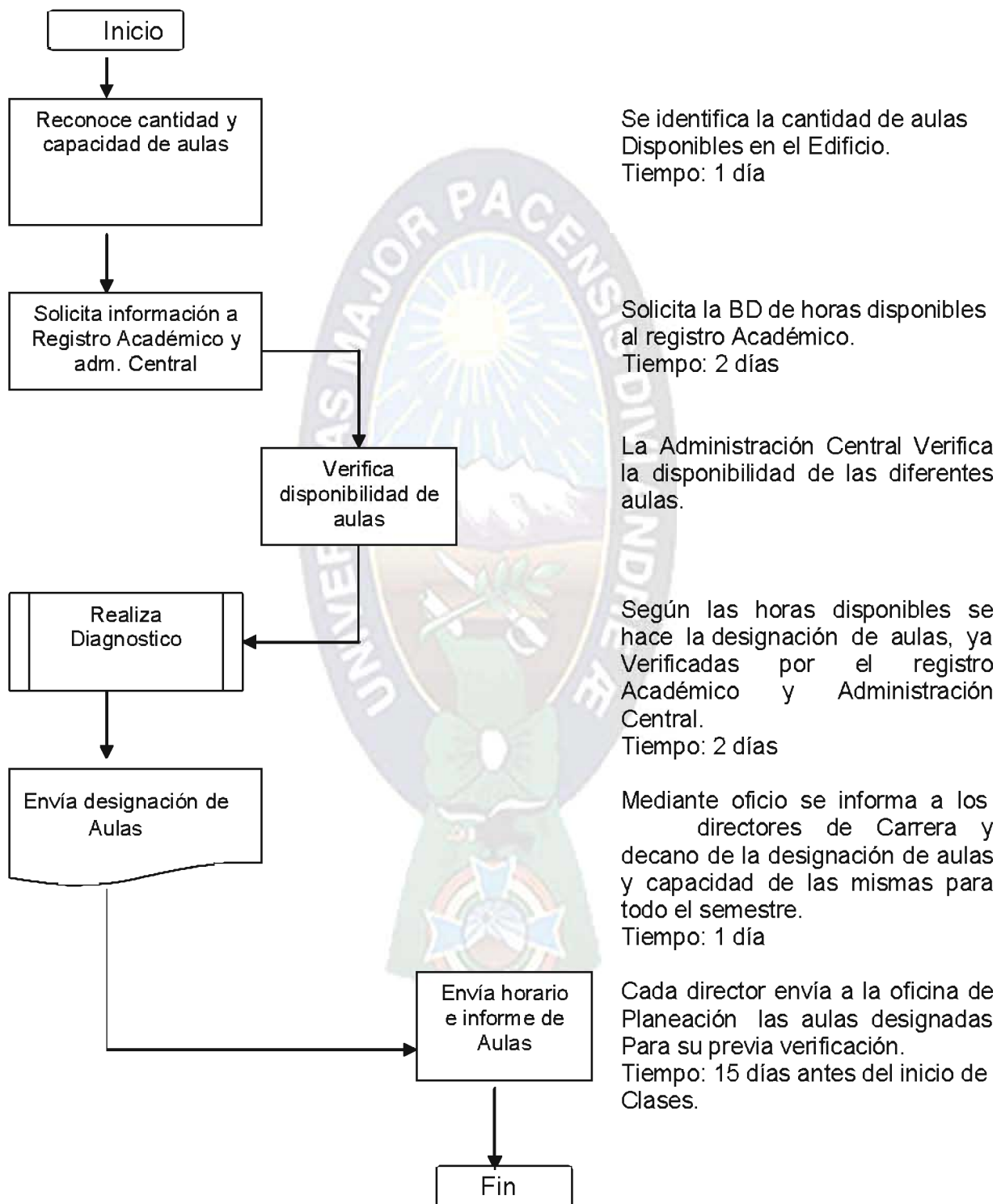
Básicamente los sistemas de designación de aulas se basan en designar una determinada área o espacio, la cual pueda ser ocupada o no, y una vez ocupada no puede ser modificada, ya que puede afectar a todo el sistema, es decir una vez realizada la designación no se puede modificar ni hacer ningún cambio en las designaciones, este cambio causaría perjuicio a estudiantes como a docentes.

Actualmente las instituciones usan softwares, los cuales realizan la designación de aulas o espacios áulicos de una manera no tan eficaz, en un rango del 1 al 10, un 6.5 sería la eficacia de los sistemas actuales, es decir que los programas desarrollados solo realizan una designación arbitraria, no se toman en cuenta los tipos o maneras de designar un determinado espacio áulico, tampoco se toman en cuenta los aspectos de cómo se puede optimizar el empleo de los distintos tipos de software, ya sea distribuyéndolos por módulos de tal manera que pueden realizar un determinado trabajo o proceso, para ello se busca diferentes formas de optimizar estos procesos.

Con los datos obtenidos y las referencias establecidas en el problema del presente trabajo, se puede ver el proceso que siguen los métodos tradicionales. A continuación se presenta el procedimiento básico que sigue la designación de aulas en una institución académica.

Procedimiento: Designación de aula

Jefe de Oficina o Kardex	Adminis - tración Central	Director de Carrera	Descripción
--------------------------	---------------------------	---------------------	-------------



Como se puede observa en el diagrama de procedimientos la designación de aulas se lo realiza en forma secuencial, es decir paso a paso sin obviar ningún proceso, ya que esto daría malos resultados y mas perdida de tiempo, se podría decir que este tipo de proceso es un poco tedioso y que emplea mucho tiempo.

En este proceso es en la que se basan los modelos anteriores de designación de recursos, es decir para ello se utiliza una base de datos, en donde se realiza la designación de aulas a la sugerencia del plantel docente, de esta manera se daba como resultado, un aula, día y hora a la cual se designaba una materia con su respectivo docente y además ver si el aula estaba desocupada o vacía, pero no se percataban mucho de esto, de la cantidad de estudiantes que están inscritas en la materia, lo cual daba como resultado; la designación de muchos estudiantes a aulas reducidas y pocos estudiantes a las aulas más amplias, lo que daba como resultado la inconformidad de los estudiantes y reclamos de los docentes, los cuales no podían trabajar cómodamente.

3.2. MÉTODO CIENTÍFICO

El método científico es el conjunto de formas que se utilizan para la adquisición y elaboración de nuevos conocimientos. Se puede decir que es el camino planeado para descubrir las propiedades del objeto o área de estudio.

La aplicación del método científico en la presente tesis contempla la realización de las siguientes etapas:

- **Análisis y síntesis:** El análisis es la descomposición de un objeto en partes. El método analítico consiste en la separación de las partes de un todo para luego estudiarlas en forma individual, en la presente tesis, se realiza un análisis de criterios que intervienen en la evaluación de una población, verificando la aptitud de cada individuo, tales criterios van de acuerdo a una función objetivo. En cuanto a la Síntesis, se puede decir que es la reconstrucción de todo lo descompuesto por el análisis, este se localiza en los operadores de cruzamiento y mutación que presenta el Algoritmo Genético. Lo que se puede mencionar en cuanto al Análisis y Síntesis es que cuando se utiliza el análisis sin llegar a la síntesis, los conocimientos no se comprenden verdaderamente y cuando ocurre lo contrario el análisis arroja resultados ajenos a la realidad;

- **Inducción y Deducción:** Se utiliza el método de inducción en el presente trabajo, ya que mediante esta se parte de datos particulares para llegar a conclusiones generales, en este caso si el Algoritmo Genético se encuentra en una fase inicial en donde se van seleccionando los individuos de la población inicial, además de la actitud de cada individuo. También se utiliza el método deductivo por que mediante ella se parte de datos generales aceptados como validos para llegar a una conclusión de tipo particular, en este caso los operadores de cruzamiento y mutación en función al cromosoma seleccionado, una vez hecho esto se designa al cromosoma resultante como una solución candidata para la solución al problema planteado;
- **Observación:** Para la observación se trabaja con el área de investigación, en este caso los ambientes áulicos de una institución educativa y la manera de distribución realizada por la institución, una vez hecha el sondeo se realiza la recopilación de los datos que se utilizaran para dar solución al problema en cuestión.

3.3. MÉTODO INFORMÁTICO

Para la solución del problema planteado en esta tesis, se tomó una metodología que es parte de la Computación Evolutiva llamada Algoritmos Genéticos, la misma que nos ayudara a la resolución del problema. Para lo que se tomaran en cuenta las etapas de la misma, para interpretar de manera sencilla el comportamiento de los algoritmos genéticos y de cómo interactúan con las variables establecidas en la hipótesis, que se detallan a continuación:

- **Análisis del Problema:** El análisis del problema es la parte fundamental para la construcción del Prototipo, esto nos encamina hacia la solución, la cual se obtiene mediante la utilización de: los objetivos planteados, definición del problema, el área en estudio, la mala distribución de los espacios áulicos y la cantidad de aulas;
- **Diseño del Cromosoma:** El diseño del cromosoma es esencial para el desarrollo del presente trabajo, ya que es la estructura inicial para generar a la población y cual se trabajará para usar los operadores de Genéticos. La estructura del cromosoma estará constituida por un conjunto de ternas;

- **Elección de la Función de Aptitud:** Hallar al función aptitud es lo primordial por que es la que evaluara a cada cromosoma o individuo de la población creada, el cual será seleccionado para ser operado;
- **Selección de los Operadores:** En esta etapa se tomará en cuenta dos operadores esenciales de los Algoritmos Genéticos los son: el operador de Cruzamiento y Mutación, ha los cuales será sometido el cromosoma;
- **Desarrollo del Prototipo:** Una vez analizado y diseñado el cromosoma se pasa al desarrollo del prototipo con el AG, en el cual se tomara en cuenta cada etapa mencionada anteriormente, usando la programación Orientada a Objetos;
- **Evaluación de Resultados:** En esta etapa se probará el prototipo, la función y el comportamiento del AG, verificando el porcentaje de efectividad en la designación de espacios áulicos.

Se seguirá las etapas de la metodología del AG de la **Figura 8** para desarrollar el prototipo, que dará solución al problema planteado, en la presente tesis.

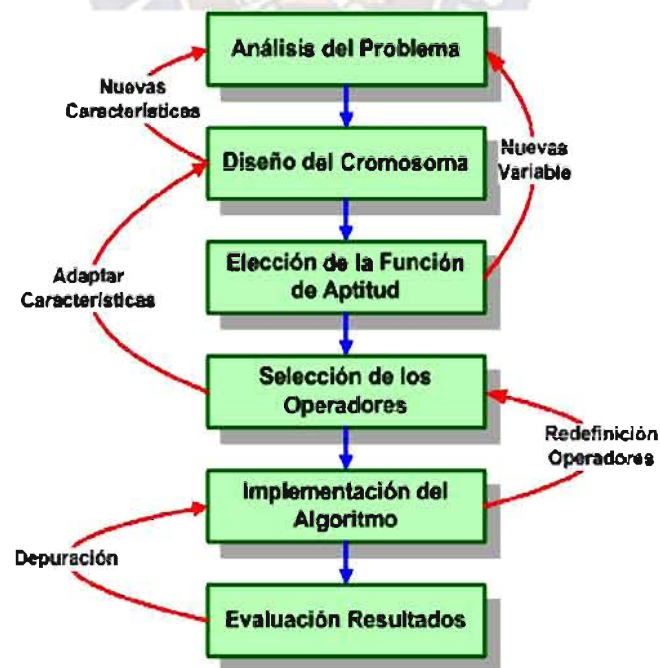


Figura 8. Etapas de la metodología del AG

3.4. IMPLEMENTACIÓN DEL PROTOTIPO

Después de haber hecho un pequeño énfasis en los modelos de designación tradicionales y los métodos a ser utilizados, se pasará a realizar un análisis del mismo, solo que se empleara la metodología de los AG con el fin de obtener una solución al tema planteado de esta investigación. Toda metodología a utilizar esta en base a este problema específico.

En este trabajo se propone una solución a la designación de aulas de una determinada infraestructura académica para ello se aplicarán los algoritmos genéticos. Para realizar la designación, se toma en cuenta las distintas materias que se dictan, el día en al que se dicta la materia, las horas de cada clase, los ambientes y algunas necesidades específicas para el dictado de clases. Este problema puede ser abstraído de la siguiente manera: se supone que existen n recursos (aulas) que son necesarios para designar a $[1, \dots, i, \dots, n]$ demandantes del recurso y que las características de los recursos son parecidas en sustancia pero no en características cuantificables. A medida que n aumenta, la explosión combinatoria se vuelve inmanejable si se quiere revisar todos los casos. Bajo estas condiciones, se puede pensar en esas combinaciones como un espacio de estados donde cada estado es una configuración posible de designación. De esta manera, y como revisar un espacio de combinaciones muy grande es dificultoso, conviene dirigir o conducir la búsqueda de la solución que permita resolver el problema de forma eficiente.

Partiremos considerando tres espacios vectoriales en R como ser: Au (aulas), Ds (días) y Hrs. (horas).

Donde:

$$\text{Au} = (a_1, a_2, \dots, a_i, \dots, a_N)$$

$$\text{Ds} = (d_1, d_2, \dots, d_j, \dots, d_D) \text{ donde } D \leq 6$$

$$\text{Hrs} = (h_1, h_2, \dots, h_k, \dots, h_M)$$

Además

$$N = \text{Número de aulas}$$

$$D = \text{Número de días}$$

$$M = \text{Número total de escalas de horas en un día}$$

De estos espacios vectoriales se obtendrá un conjunto de ternas ordenadas, denotado por:

$$Desig = \{(a_i, d_j, h_k) \mid a_i \in Au, d_j \in Ds, h_k \in Hrs\}$$

Donde **Desig** es el conjunto de ternas ordenadas (a_i, d_j, h_k)

La representación grafica se lo realiza en una coordenada cartesiana tridimensional, dado a que son tres espacios vectoriales, es decir básicamente sería un cubo dimensional.

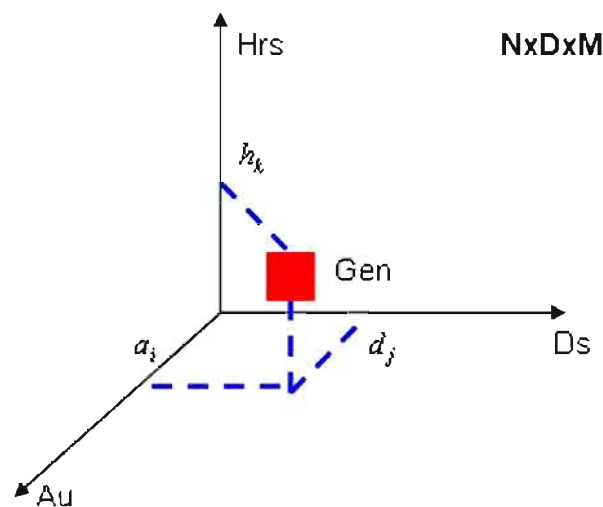


Figura 9. Representación del gen

Al trabajar con un modelo genético se deben definir los aspectos que están relacionados con la implementación como ser: el **cromosoma**, que representa a los individuos de una población, para luego aplicar los elementos y operadores de los Algoritmos Genéticos para dar solución al problema planteado.

La mayoría de los desarrollos de aplicaciones que utilizan AG, manejan representación binaria para los individuos. Esto significa que los cromosomas son cadenas de bits cuyos valores representan situaciones particulares en la resolución. Sin embargo, no existe limitación alguna para modelar y trabajar con cromosomas cuyos parámetros sean no

binarios. En este caso, para el modelo que nos ocupa, se deben considerar ciertos aspectos de la representación y la aplicación de los operadores no binarios.

3.4.1. Estructura de un cromosoma

Para establecer el formato del cromosoma se necesitan varias estructuras de datos que contengan la información relevante de los espacios áulicos que se designen. Para ello se define la siguiente estructura:

$$Desig = \left[\begin{array}{l} (a_1, d_1, h_1), (a_1, d_1, h_2), \dots, (a_1, d_1, h_M), (a_1, b_2, h_1), \dots, (a_1, b_2, h_M), \dots, (a_1, d_D, h_M), \dots, (a_2, d_1, h_1) \\ \dots, (a_2, d_D, h_M), \dots, (a_i, d_j, h_k), \dots, (a_N, d_1, h_1), \dots, (a_N, d_1, h_M), \dots, (a_N, d_2, h_M), \dots, (a_N, d_D, h_M) \end{array} \right]$$



Cada una de estas ternas representa un Gen

Como se observa en la estructura planteada del cromosoma se forman ternas las cuales son consideradas como gen, por tanto el conjunto de estas ternas (genes) forman un cromosoma.

Para nuestro caso el total de genes que tiene un cromosoma será de $(N \times D \times M) + 1$. De los cuales $(N \times D \times M)$ genes serán ocupados por las ternas y el adicional será por el valor de aptitud del cromosoma. Ya la función aptitud es el que evalúa la terna o gen del cromosoma.

Cada uno de los componentes de estas ternas, es decir a_i , d_j y h_k representan la ubicación del aula, día y hora respectivamente, para su mejor estudio es necesario crear una tabla de posiciones, donde se guardará cada terna en el orden creado.

Para designar un elemento de cada terna (gen) se ha considerado un conjunto de materias, denotadas por:

$$Mat = (m_1, m_2, \dots, m_x, \dots, m_{TM})$$

m_x = Código de materia

TM = Número Total de materias

Es sabido que una materia se designa dos veces por semana, lo cual el total de materias a ser considerados para la designación es $(TM * 2)$, el cual esta sujeta a esta restricción:

$$(TM * 2) \leq (N * D * M)$$

En otras palabras, una materia m_x es designada a un gen de un cromosoma $Desig_p$ de la siguiente manera:

$$Desig_p(a_i, d_j, h_k) = m_x$$

Por consiguiente el cromosoma representado por valores alfabéticos, cada terna (gen) toma dos posibles estados de designación, como estar ocupado o libre.

El cromosoma a ser utilizado es el siguiente:

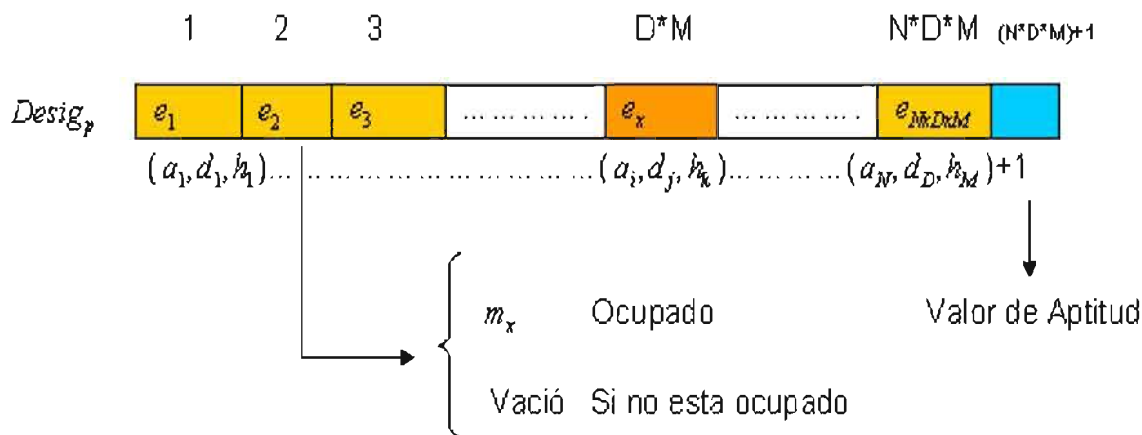


Figura 10. Representación del cromosoma

Como se observa en la **Figura 10** de la estructura del cromosoma, se observa que cada elemento e_i ; Representa a cualquier elemento del conjunto de materias o espacios vacíos en el caso que existiera.

De acuerdo con la modalidad de designación (mencionado en el Cáp. 1), tomaremos una tabla de sugerencias de horarios (TSH), donde un determinado docente con su respectiva materia sugerirá el horario a su conveniencia, pero esta tabla no es una solución óptima o viable ya que pueden existir, tales como, choque de horarios o que varios docentes soliciten el mismo horario, pero para nuestro caso tomará un papel muy importante en el momento de evaluar el cromosoma en cuestión.

La tabla de sugerencias es el siguiente:

	Cod. Materia	1° Día	1° Hora	2° Día	2° Hora
1					
.					
.					
.					
TM					

Tabla 4. Tabla de sugerencias

Básicamente ya tenemos todo definido o lo que es más importante la estructura del cromosoma y sus elementos.

Pasemos a definir el diagrama de proceso de un Algoritmo Genético el cual tomaremos para la resolución del problema planteado. En este diagrama se detalla básicamente las funciones, clases y procedimientos que aplicaremos para cada uno de los operadores genéticos, que componen al algoritmo genético.

En el prototipo a ser desarrollado seguirá la secuencia de este diagrama, donde se partirá desde la selección de la población inicial y se terminará en la mutación si el individuo o cromosoma a ser analizado no es apto, de tal manera que será verificado en un ciclo infinito hasta que sea el más apto, para luego ser designada.

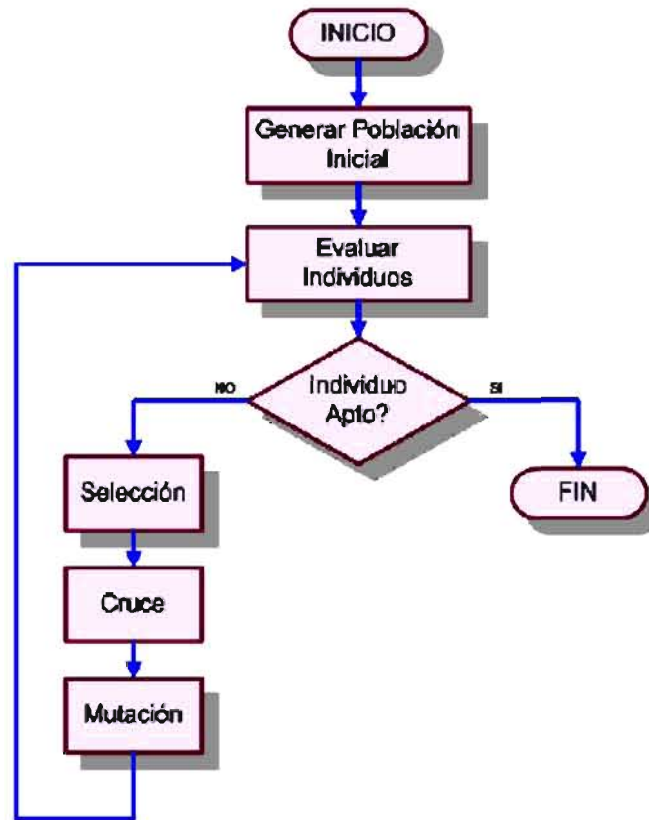


Figura 11. Diagrama de procesos del AG

La idea fundamental es representar en una misma estructura todas las características del problema, mediante una función de evaluación, la cual determinará si ese cromosoma está listo para ser verificado o simplemente es considerado una solución candidata. Luego de ser evaluado, se verifica o determina la función de aptitud, la cual no es una tarea sencilla, como a priori no se conoce una función “buena” puede ser que existan problemas de múltiples óptimos locales y de aislamiento del óptimo global.

Una vez establecido la estructura del funcionamiento del Algoritmo Genético o el diagrama de proceso, se pasará a realizar el pseudo-código en el cual nos basaremos para la codificación de los operadores del algoritmo genético, con lo cual se desarrollará el prototipo para la solución al problema.

```

BEGIN /* Algoritmo Genético */
    Generación = 0
    Generar P (población inicial).
    Computar la función de evaluación Eval(Desig).
    WHILE NOT Terminado DO
        BEGIN /* Producir nueva generación */
            Seleccionar dos padres de la población P Desig.
            Cruzar P( $Desig_p, Desig_q$ ).
            Mutar P( $Desig_p$ ).
            Computar la función de evaluación P Eval(Desig).
            Insertar los dos descendientes mutados en la nueva
            generación.
            Generación += 1
            IF la población ha encontrado una solución THEN
                Terminado:= TRUE
        END
    END
END

```

3.4.2. Población

Como se mencionó anteriormente la solución adecuada al problema es buscada en un conjunto de cromosomas, las cuales representaran una solución candidata. Este conjunto de cromosomas hace referencia a la población, a la cual denotaremos por el conjunto P, donde estarán todos los individuos a ser transformados y de los cuales se tomara un par de ellos los más aptos para operar con ellos, como se muestra a continuación:

$$P = \{Desig_1, Desig_2, \dots, Desig_p, \dots, Desig_{TP}\}$$

Donde:

$Desig_p$: Posible solución (cromosoma)

TP : Tamaño de la población

Para que la población sea generada, se hace mediante permutaciones, el cual nos da el número total de permutación (NTP) para un determinado caso como:

$$NTP = [(NxDxM)/(2_1!2_2!...2_{TM}!)]$$

Esta formula es para datos grandes, el cual da un número muy alto de permutaciones y si los datos son pequeños el número de permutación es considerable.

Además, se sabe que para definir un tamaño de población se debe tener mucho cuidado, ya que si es muy pequeño o muy grande acarrea problemas. En nuestro estudio tomaremos una población de 90 cromosomas (TP = 90).

La población inicial se genera aleatoriamente, la cual ira evolucionando con cada iteración que se realice.

3.4.3. Evaluación

Para dar una medición o valor a cada solución de la población, es necesario la construcción de la función objetivo, la misma que se forma de acuerdo a las condiciones o restricciones ya establecidas.

Para nuestro estudio la función objetivo será:

$$Eval(Desig_p) = \alpha + \sum_{s=1}^{TM} [\gamma(s) + \beta(s)]$$

$$\alpha = \begin{cases} TM & \text{Si } \forall \text{ docente con } m_x > 1 \Rightarrow d_j \neq d_v \wedge h_k \neq h_w, \\ & \text{Donde } d_j, h_k, d_v, h_w \in Desig_p(a_i, d_j, h_k) \text{ y } Desig_p(a_u, d_v, h_w) \\ 0 & \text{e.o.c.} \end{cases}$$

α = Función condicional de todos los docentes con varias materias, de modo que estas materias no tengan el mismo horario.

$$\gamma(s) = \begin{cases} 1 & \text{Si } \forall \text{Desig}_p(a_i, d_j, h_k) = \text{Desig}_p(a_u, d_v, h_w) \Rightarrow d_j \neq d_v \\ 0 & \text{e.o.c.} \end{cases}$$

s = Identifica una determinada materia

$\gamma(s)$ = Función condicional de una materia (materia que se habilita dos veces por semana, estos deben contarse en días distintos).

$$\beta(s) = \begin{cases} 2 & \text{Si } \text{Desig}_p(a_i, d_j, h_k) = \text{Desig}_p(a_u, d_v, h_w) \Rightarrow d_j, h_k \wedge d_v, h_w \in TSH \\ 1 & \text{Si } \text{Desig}_p(a_i, d_j, h_k) = \text{Desig}_p(a_u, d_v, h_w) \Rightarrow d_j, h_k \vee d_v, h_w \in TSH \\ 0 & \text{e.o.c.} \end{cases}$$

$\beta(s)$ = Función condicional según las sugerencias dadas (evalúa a las materias comprobando sus horarios con la tabla de sugerencias).

El máximo valor que debe alcanzar un cromosoma es:

$$\text{TotEval} = TM + TM + (TM * 2)$$

Esta formula significa que:

1º TM Todos los docentes con varias materias tienen distintos horarios, lo cual es muy importante para prevenir choques de horarios en una misma aula.

2º TM Si todas las materias, con sus correspondientes horarios no dictan en un mismo día.

3º (TM*2) Significa que todas las materias con sus dos respectivos horarios cumple con la tabla de sugerencias.

Es importante que las dos primeras condiciones alcancen su valor máximo, en vista de que evitan errores de incoherencia mencionados anteriormente, asegurando de esta

manera un 50% de efectividad en la solución (solución válida). Se considera que la última condición no necesita cumplir con su valor máximo como las dos anteriores, puesto que se puede o no complacer el horario que cada docente sugiere. Por tanto un valor aproximado que deberá esperarse, es que cumpla el 60% al 80% del total de evaluación (TotEval).

3.4.4 Selección

Una vez determinado el valor de aptitud de los cromosomas, procede a seleccionar aquellos individuos (cromosomas) con mayor aptitud, es decir aquellos que representen una solución candidata o aquel que se mantenga y nunca decrezca.

Para esto utilizaremos el operador de selección, basándonos en el criterio del elitismo, el cual consiste en seleccionar el mejor de los cromosomas y preservarlo hasta que surja otro mejor que este. Todo esto con el objetivo de no perder una solución posible. Luego se procede a la selección de individuos.

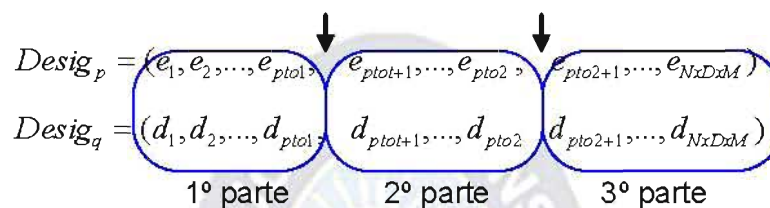
Para la selección de individuos utilizaremos el método de selección por torneo, ya es considerado uno de los mejores tipos de selección que, básicamente en este tipo de selección se realiza un torneo en el cual uno de los individuos resulta el vencedor. El cual pasará a ser una de las soluciones, y de la misma manera se procede con los demás llegando a obtener una cantidad determinada de la población inicial y para luego pasar al siguiente proceso.

3.4.5 Cruzamiento

En el conjunto de individuos seleccionados en el proceso anterior, se procederá a realizar el respectivo cruzamiento de los mismos. El operador de cruzamiento toma dos cromosomas aleatoriamente de la población o del conjunto de cromosomas seleccionados, para el caso nuestro $Desig_p$ y $Desig_q$ los cuales serán cruzados, para esto usaremos uno de los tipos de cruce, específicamente el de dos puntos (de emparejamiento), estos puntos de cruce serán fijos, es decir que no se procederá con los pasos estándares de esta técnica.

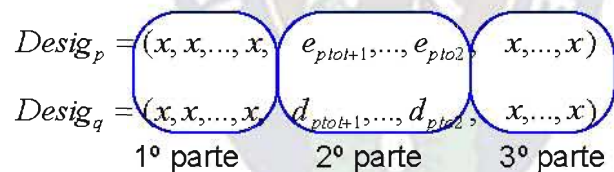
Este tipo de cruzamiento fue propuesto por [Michalewicz, 1996], el cual sigue el siguiente proceso:

- Obtener dos cromosomas $Desig_p$ y $Desig_q$, luego fijar dos puntos en cada Cromosoma, luego realizamos en los puntos fijados pto1 y pto2, de tal forma que obtenemos tres segmentos de cada cromosoma.



Donde e_i y d_i representan al conjunto de código de cada materia o espacios vacíos a utilizar.

- Tacharemos con una X las posiciones de datos a intercambiar, en este será la 1º y 3º, y en la 2º parte o segmento no realizar ninguna modificación mantener su contenido, veamos:



- Para obtener el primer descendiente, se toma el segundo cromosoma de $Desig_q$ y formar con los electos un vector auxiliar de la siguiente manera:

$$\begin{array}{l}
 d_{pto2+1}, \dots, d_{NxDxM}, d_1, d_2, \dots, d_{pto1}, d_{pto1+1}, \dots, d_{pto2} \\
 \text{3º parte} \qquad \qquad \qquad \text{1º parte} \qquad \qquad \qquad \text{2º parte}
 \end{array}$$

- Luego, pasamos a eliminar los elementos que existen en la 2º parte del cromosoma $Desig_p$ del vector auxiliar. Una vez eliminado, queda un vector (auxiliar) reducido el cual se pasará a dividir en dos partes (la división que se hará debe tomar la longitud a la cual pasará a reemplazar).

- Por ultimo, para obtener el primer cromosoma descendiente se toma la primera división del vector auxiliar, la cual ocupara la 3º parte del cromosoma descendiente, la segunda división pasará a ocupar la 1º parte del cromosoma y por ultimo la 2º parte del cromosoma será ocupado por la 2º parte del cromosoma de $Desig_p$.

Por ejemplo:

$$Desig_p = (1, 0, 0, 1, 0, 1, 1, 0, 1, 1)$$

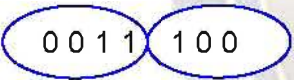
$$Desig_q = (1, 1, 0, 1, 0, 0, 0, 0, 1, 1)$$

$$Desig_p = (x, x, x, 1, 0, 1, x, x, x, x)$$

$$Desig_q = (x, x, x, 1, 0, 0, x, x, x, x)$$

Vector auxiliar 0 0 1 1 1 1 0  se eliminan los existentes en $Desig_p$.

Quedará:



$$0 0 1 1 \quad 1 0 0$$

Entonces el primer descendiente será:

$$Desig_{Desc1} = (1, 0, 0, 1, 0, 1, 0, 0, 1, 1)$$

Para obtener el segundo descendiente de estos dos cromosomas debemos formar otro vector auxiliar, pero esta vez tomando en cuenta al cromosoma $Desig_p$, y se pasará a realizar los mismos pasos que se hicieron con el primer descendiente, pero de forma inversa al utilizar los cromosomas.

La razón por la que se tomó este tipo de cruzamiento es por que evita perder información o adicionar datos erróneos en un cromosoma descendiente, es decir que un cromosoma no sea viable para dar solución al problema. Se debe tener en cuenta que no todos los individuos de la población serán cruzados, si depende mucho de la probabilidad de cruzamiento.

3.4.6. Mutación

Este operador de mutación se utilizará para el cambio de un dato del gen de $Desig_p$, la posición de la misma se elegirá aleatoriamente. En este caso reemplazar cualquier dato resulta riesgoso ya que se tropieza con el mismo caso que con el anterior operador.

Para proceder con esta operación de mutación, se obtiene dos posiciones aleatoriamente e intercambian sus elementos o datos. Como se observa a continuación:

$$Desig_p = (e_1, e_2, \dots, e_{pos1}, \dots, e_{NxDim})$$

Guardar e_{pos1} , luego obtener la segunda posición en el mismo cromosoma

$$Desig_p = (e_1, e_2, \dots, \downarrow e_{pos1}, \dots, \downarrow e_{pos2}, \dots, e_{NxDim})$$

Por último, e_{pos2} se ubica en la primera posición obtenida y e_{pos1} se ubica en la segunda posición, por consiguiente en cromosoma mutado será:

$$Desig_p = (e_1, e_2, \dots, e_{pos2}, \dots, e_{pos1}, \dots, e_{NxDim})$$

De esta manera se evita que un cromosoma sufra de cambios considerables, que perjudiquen al mismo.

Al igual que el anterior operador se necesita de una cierta probabilidad de mutación lo cual determina cuantos individuos de la población serán mutados.

3.5. TECNOLOGÍA EMPLEADA

Para el desarrollo del prototipo emplearemos el lenguaje de programación Visual Basic .Net, por su facilidad de manejo en cuanto a la interfaz y a la codificación de código fuente. La gran mayoría de programadores de VB.NET utilizan el entorno de programación Microsoft Visual Studio .Net en alguna de sus versiones (Visual Studio .NET, Visual Studio .NET 2003 o Visual Studio .NET 2005), aunque existen otras alternativas, como SharpDevelop (que además es libre). Así de esta manera podremos

trabajar a la vez con Visual Studio .NET 2005 ya que vienen a ser uno mismo, el cual nos facilitará un entorno mas fácil de manipular.

Visual Basic .NET posee gran capacidad en el manejo de funciones, procedimientos y clases, además de contar con librerías externas por ser un lenguaje de alto nivel y el diseño orientado a objetos, el entorno desarrollado que posee permite la manipulación de varios componentes que permiten un desarrollo de aplicaciones con mayor facilidad en diferentes áreas.

3.6. INTERFAZ DEL PROTOTIPO

El prototipo desarrollado presenta una interfaz grafica (ver **Figura 12**) diseñada en función del análisis y diseño ya presentados, opera bajo la plataforma Windows, se presenta una pantalla principal, el cual permite al usuario u operador llevar acabo las operaciones propias del modelo o prototipo.

El menú principal del sistema para la Designación dinámica de infraestructura académica consta de tres opciones: entrar al prototipo, información sobre el mismo y salir del prototipo.

En el capítulo siguiente se verá la prueba y ejecución del prototipo.



Figura 12. Interfaz del Prototipo



4

PRUEBAS Y RESULTADOS

4. PRUEBAS Y RESULTADOS

En este capítulo, se realizará el análisis de los resultados, aplicando algunas métricas de pruebas al prototipo desarrollado.

4.1. ANÁLISIS DE RESULTADOS

Para el análisis de resultados se consideran varios parámetros los cuales influyen de gran manera en el comportamiento de los Algoritmos Genéticos y en el tiempo de obtención de los resultados. Entre los parámetros introducidos están:

- El número de individuos que conforman la población;
- La probabilidad de cruzamiento;
- La probabilidad de mutación;
- Los valores de entrada para un determinado caso de designación tales como.

Número de Aulas (N).

Número de Días (D).

Intervalos o Escalas de horas (M).

Número de Materias (TM).

Todos estos valores de entrada estarán escritos en un documento TXT, el cual se lo cargara en el prototipo. Cabe señalar que se tomaran parámetros fijos como: una población de 90 cromosomas (TP), se opto por este tamaño de población por la efectividad del algoritmo genético, también se considera una probabilidad de cruzamiento en un 0.95 y una probabilidad de mutación de 0.1, haciendo de esta forma que exista una mayor cantidad de generaciones de predecesores. Por lo cual, se observó la conveniencia de solo hacer variar los parámetros de entrada como: N, D, M y TM.

4.1.1. Métricas orientadas a la función

Para medir el punto función se hace uso del diagrama de flujo de datos, el cual se evalúa para determinar las medidas clave necesarias para el cálculo de la métrica de punto función.

Se realizaron varias corridas del algoritmo con diferentes parámetros, para verificar la funcionalidad del modelo, el cual arrojará diferentes resultados de acuerdo a los parámetros introducidos, para lo cual se analiza dos casos:

Caso 1:

Valores de entrada: $N = 2$

$D = 3$

$M = 3$

$TM = 7$

TASA DE EFECTIVIDAD %	NRO. DE ITERACIONES	TIEMPO
51 - 59	0 - 80	0.2 - 1 seg.
61 - 69	4 - 25	1 - 60 seg.
≥ 70	10 - 1000	1 - 70 seg.

Tabla 5. Resultados del caso 1

Caso 2:

Valores de entrada: $N = 4$

$D = 5$

$M = 4$

$TM = 25$

TASA DE EFECTIVIDAD %	NRO. DE ITERACIONES	TIEMPO
51 - 59	0 - 80	0.2 - 1 seg.
61 - 69	100 - 23400	1 - 52 min.
≥ 70	> 4000	> 10 min.

Tabla 6. Resultados del caso 2

La tasa de efectividad se consideró a partir del 51%, ya los individuos que alcanzan este nivel se consideran soluciones factibles, de tal forma que se presenta una designación sin errores, es decir cumple en su totalidad las dos condiciones primeras de la función objetivo. Por tanto los individuos que sobre pasen el 51% llegan a tomar la tercera restricción, en otras palabras, el 51% mide el nivel de satisfacción de la tabla de sugerencias propuesta por los docentes.

Los resultados obtenidos se miden en intervalos ya que no se puede asegurar el tiempo o la iteración exacta en la que se obtuvo el resultado, todo se debe al comportamiento aleatorio que demuestran los AGs. Es decir, en una ejecución realizada puede obtenerse la solución en la primera o segunda generación como también puede encontrarse después de muchas generaciones, por lo tanto no es predecible el tiempo de parada.

Una vez verificado la funcionalidad del modelo se hará uso de la métrica de punto función en el cual se evalúan las siguientes medidas clave que son necesarias para el cálculo de la métrica de punto de función:

- Número de entradas del usuario = 2;
- Número de salidas del usuario = 3;
- Número de consultas del usuario = 1;
- Número de archivos = 2;
- Número de interfaces externas = 2.

Además se usará la siguiente ecuación a la cual debe ajustarse la cuenta total:

$$PF = c\text{-total} \times (0,65 + 0,01 \times \Sigma Fi)$$

Donde F_i ($i = 1$ a 14) son “los valores de ajuste de complejidad”. Para el propósito de nuestro calculo, asumimos que ΣFi es 43 (un producto moderadamente complejo).

A continuación calcularemos la cuenta total de las medidas clave del punto función, para luego reemplazar los datos obtenidos en la ecuación, la cuenta total es la suma de todas las entradas clave del punto función, que se detalla en la siguiente tabla.

Factor de ponderación

Parámetro de medición	Cuenta		Simple	Media	Compleja			
Numero de entradas de usuario.	2	x	3	4	6	=	6	
Numero de salidas del usuario.	3	x	4	5	7	=	12	
Numero de consultas del usuario.	1	x	3	4	6	=	3	
Numero de archivos	2	x	7	10	15	=	14	
Numero de interfaces externas	1	x	5	7	10	=	5	
Cuneta Total								40

Tabla 7. Calculo de PF sin ajustar

Por tanto el ajuste de punto función es:

$$PF = 40 \times [0,65 + (0,01 \times 43)] = 43$$

Basándose en el valor previsto de PF obtenido del modelo de análisis, se puede estimar el tamaño global de implementación de las funciones de interacción, que son básicamente seguras.

El fin de las métricas del software es intentar estimar la funcionalidad y la utilidad del sistema antes de comenzar (solo con los requisitos de usuarios). Su objetivo es poder ver si el sistema va a ser complejo en su realización. Una vez calculado los puntos de función se usaremos de forma analógica a las LDC como medida de la productividad, calidad y otros productos del software (solo estimación). Esto solo en el caso de que el software desarrollado sea implementado.

Productividad	=	PF/Personas-mes	=	40/5	=	8
Calidad	=	Errores/PF	=	50/40	=	1.25
Costo	=	Dólares/PF	=	5000/40	=	125 \$us.
Documentación	=	Pg.Doc /PF	=	77/40	=	1.9

4.2. PRUEBAS DEL PROTOTIPO

Como ya se mencionó en el anterior capítulo el prototipo fue desarrollado en el lenguaje de programación orientado a objetos Visual Basic .Net, como se había mencionado este lenguaje posee interfaz amigable y es de fácil manipulación por el usuario y permite el manejo de estructuras dinámicas.

A continuación se ejecutará una de las soluciones que se obtienen con el prototipo para un determinado caso.

En la **Figura 12** del capítulo anterior se muestra la interfaz inicial del prototipo, la cual cuenta con tres botones (Entrar, Acerca de... y Salir del sistema), una vez dentro del sistema o prototipo se observa el ingreso de los parámetros, como se ve en la **Figura 13**.

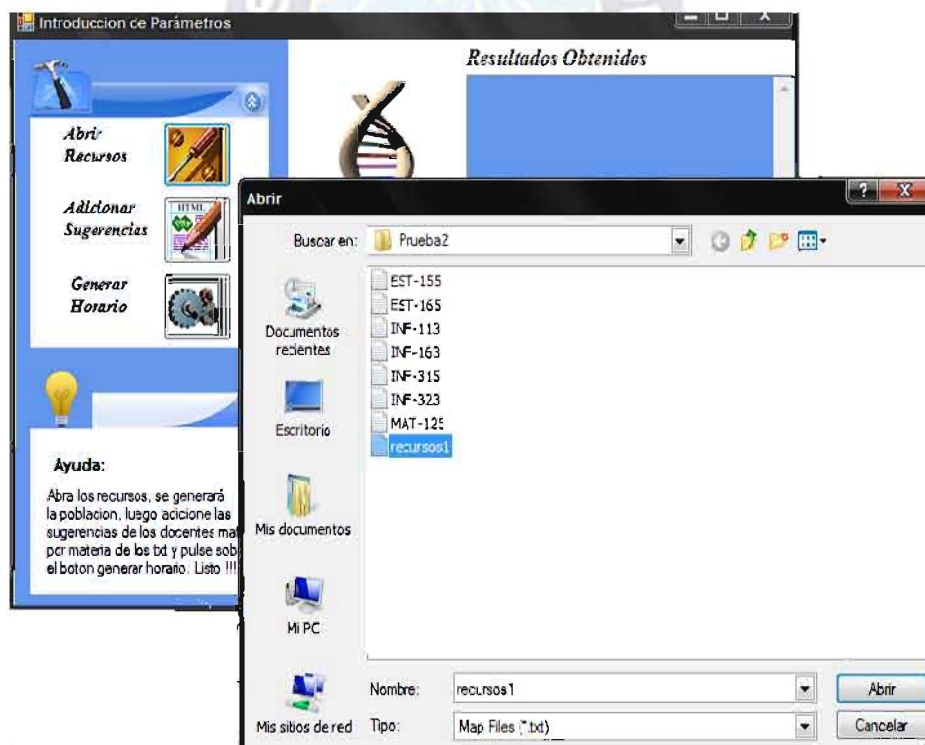


Figura 13. Introducción de parámetros

Como se mencionó anteriormente los parámetros de entrada se realizaron en un TXT, la misma que tiene un formato especial para ser procesada, la cual es agregada en el prototipo como se muestra en la **Figura 13**.

De acuerdo a los parámetros introducidos anteriormente se obtiene lo siguiente: El cuadro en la parte derecha muestra la población generada, luego en el mismo cuadro se agrega el horario sugerido por los docentes en sus distintas materias y horarios que tienen disponibilidad de tiempo.

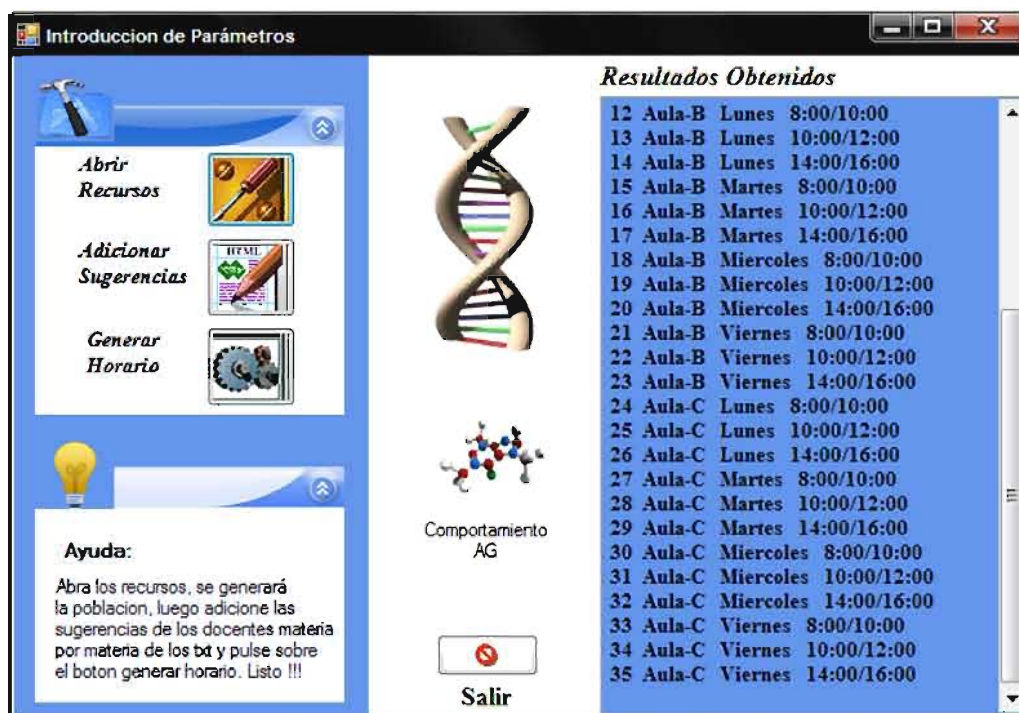


Figura 14. Generación de la Población Inicial

La población es generada una vez cargado los recursos de la infraestructura académica y los parámetros de entrada, es decir se genera un cromosoma con todas las combinaciones posibles de todos los parámetros introducidos en el documento recursos.txt, (Ver **Figura 15**)

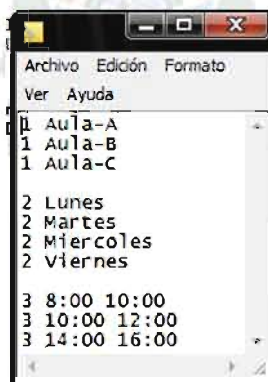


Figura 15. Formato de Parámetros

Luego se hace clic sobre el botón generar horario donde se observa el horario generado, esto después de varias iteraciones y de operaciones especificadas en los capítulos anteriores, el horario generado es el siguiente, done solo son designadas las aulas A y B como se observar en la **Figura 16**, las materias designadas no presentan choques, además de que cada materia cumple con las condiciones establecidas y sugerencias de los docentes.

Lunes	Martes	Miercoles	Jueves	Viernes	Sabado	<u>Aula-B</u>
Vacio	Vacio	INF-113	Vacio	Vacio	Vacio	8:00-10:00
INF-315	Vacio	EST-165	Vacio	INF-315	Vacio	10:00-12:00
Vacio	Vacio	Vacio	Vacio	Vacio	Vacio	12:00-14:00
INF-163	EST-155	INF-323	Vacio	MAT-125	Vacio	14:00-16:00
Vacio	Vacio	Vacio	Vacio	Vacio	Vacio	16:00-18:00
Vacio	Vacio	Vacio	Vacio	Vacio	Vacio	18:00-20:00
Vacio	Vacio	Vacio	Vacio	Vacio	Vacio	20:00-22:00

Lunes	Martes	Miercoles	Jueves	Viernes	Sabado	<u>Aula-C</u>
MAT-125	Vacio	Vacio	Vacio	Vacio	Vacio	8:00-10:00
Vacio	INF-113	Vacio	Vacio	Vacio	Vacio	10:00-12:00
Vacio	Vacio	Vacio	Vacio	Vacio	Vacio	12:00-14:00
INF-163	INF-323	EST-165	Vacio	EST-155	Vacio	14:00-16:00
Vacio	Vacio	Vacio	Vacio	Vacio	Vacio	16:00-18:00
Vacio	Vacio	Vacio	Vacio	Vacio	Vacio	18:00-20:00
Vacio	Vacio	Vacio	Vacio	Vacio	Vacio	20:00-22:00

Figura 16. Horario Óptimo

Además de las pruebas efectuadas anteriormente, es decir pruebas de caja negra, también aplicaremos las pruebas o métricas de Halstead, con estas dos técnicas de prueba básicamente se verificaría la mayoría de los casos de prueba del software.

4.2.1. Prueba de Halstead

Primeramente se utilizará lo que es la teoría de Halstead para luego pasar la prueba, Halstead se aplica para especificar las medidas primitivas para desarrollar expresiones para la longitud global del programa, volumen mínimo potencial para un algoritmo, el

volumen real (número de bits requeridos para especificar un programa), el nivel del programa (una medida de la complejidad del software), nivel del lenguaje (una constante para un lenguaje dado), y otras características tales como el esfuerzo de desarrollo, tiempo de desarrollo e incluso el número esperado de fallos en el software.

Para este propósito, utilizaremos un segmento de código interno, el cual deberá ser el más relevante del prototipo desarrollado.

```

Sub Aptitud()
  For Each m As Materia In materias 'Horario sugerido
    If m.h1.Ds.nombre = "null" And m.h1.Hrs.hora_ini = "null" And
      m.h1.Hrs.hora_fin = "null" Then
      TextBox1.AppendText(m.codigo + " " + m.h1.Au.id.ToString + "/" +
        m.h1.Ds.nombre.ToString + "/" + m.h1.Hrs.hora_ini.ToString + "/" + "/" +
        m.h1.Hrs.hora_fin.ToString)
      For Each m2 As Materia In materias2 'horario sin choques
        If m2.codigo.Equals(m.codigo) Then
          m.h1.Au = m2.h1.Au
          m.h1.Ds = m2.h1.Ds
          m.h1.Hrs.hora_ini = m2.h1.Hrs.hora_ini
          m.h1.Hrs.hora_fin = m2.h1.Hrs.hora_fin
        End If
      Next
      TextBox1.AppendText(_enter)
      TextBox1.AppendText("asignado:")
      TextBox1.AppendText(_enter)
      TextBox1.AppendText(m.codigo + " " + m.h1.Au.id.ToString + "/" +
        m.h1.Ds.nombre.ToString + "/" + m.h1.Hrs.hora_ini.ToString + "/" + "/" +
        m.h1.Hrs.hora_fin.ToString)
      TextBox1.AppendText(_enter)
    End If
    If m.h2.Ds.nombre.Equals("null") And m.h2.Hrs.hora_ini.Equals("null") And
      m.h2.Hrs.hora_fin.Equals("null") Then
      TextBox1.AppendText(m.codigo + " " + m.h2.Au.id.ToString + "/" +
        m.h2.Ds.nombre.ToString + "/" + m.h2.Hrs.hora_ini.ToString + "/" + "/" +
        m.h2.Hrs.hora_fin.ToString)
      For Each m2 As Materia In materias2
        If m2.codigo.Equals(m.codigo) Then
          m.h2.Au = m2.h2.Au
          m.h2.Ds = m2.h2.Ds
          m.h2.Hrs.hora_ini = m2.h2.Hrs.hora_ini
          m.h2.Hrs.hora_fin = m2.h2.Hrs.hora_fin
        End If
      Next
      TextBox1.AppendText(_enter)
      TextBox1.AppendText("asignado:")
      TextBox1.AppendText(_enter)
      TextBox1.AppendText(m.codigo + " " + m.h2.Au.id.ToString + "/" +
        m.h2.Ds.nombre.ToString + "/" + m.h2.Hrs.hora_ini.ToString + "/" + "/" +
        m.h2.Hrs.hora_fin.ToString)
      TextBox1.AppendText(_enter)
    End If
  Next
End Sub

```

Segmento de código de la Aptitud del AG

Calculemos las medidas preventivas necesarias con las que trabaja Halstead para el cálculo de sus métricas:

n_1 = número de operadores diferentes en el programa	= 14
n_2 = número de operandos distintos en el programa	= 22
N_1 = número total de veces que aparecen los operadores	= 110
N_2 = número total de veces que aparecen los operandos	= 200

Aplicando a las ecuaciones de Halstead se tiene:

Longitud global del programa

$$N = n_1 \log_2 n_1 + n_2 \log_2 n_2 \Rightarrow N = 14 \log_2 14 + 22 \log_2 22 = 151$$

Volumen del programa

$$V = N \log_2 (n_1 + n_2) \Rightarrow V = 151 \log_2 (14 + 22) = 781$$

Ahora pasaremos a aplicar la prueba con Halstead. Usando la definición del volumen de un programa, V , y nivel de programa, NP , el esfuerzo de la ciencia del software puede ser calculada como:

$$NP = 1 / [(n_1/2) \times (N_2/n_2)] \Rightarrow NP = 1 / [(14/2) \times (110/22)] = 0.03$$

Como se puede observar, el resultado obtenido demuestra el esfuerzo de la ciencia del software, la cual no es muy grande debido a que el segmento de código seleccionado no es tan complejo. Pero se ve claramente que el programa en si es de gran tamaño.

4.3. VALIDACIÓN DEL PROTOTIPO

Para esta parte se tomará el porcentaje de la efectividad que presenta el prototipo en la designación de infraestructura académica y el valor que se espera obtener es el 100%, pero como es de saber que nada es perfecto por diferentes razones, por lo que se ha tomado un margen de error de un 5%, claro que puede ser mas, pero para el caso el margen de error es adecuado. Este margen de error del 5% ocurre a que la designación realizada no satisficará a todos en general, puesto que siempre habrá alguien que no este de acuerdo con la designación realizada.

Para probar al prototipo y ver la efectividad se realizó 30 pruebas para tener una cantidad considerable que muestre el porcentaje de efectividad de cada caso analizado.

Nro.	Efectividad del Prototipo %	Valor esperado %
1	95	95
2	90	95
3	92	95
4	90	95
5	95	95
6	95	95
7	95	95
8	90	95
9	95	95
10	95	95
11	95	95
12	95	95
13	95	95
14	90	95
15	95	95
16	92	95
17	93	95
18	95	95
19	90	95
20	95	95
21	91	95
22	93	95
23	95	95
24	93	95
25	95	95
26	90	95
27	95	95
28	93	95
29	95	95
30	95	95
x	93,4	95
s	2,061134606	
t	1,06295	

Tabla 8. Resultados del prototipo

Para comprobar la hipótesis planteada se tomará en cuenta las siguientes hipótesis:

H₀: La designación de infraestructura con Algoritmos Genéticos tienen un % \leq 93

H₁: La designación de infraestructura con Algoritmos Genéticos tienen un % $>$ 93

Donde $\mu = 93$, además es de saber que la población presenta una distribución normal en el muestreo de la media, por lo que una distribución t student con n-1 grados de libertad. Usando un nivel de significancia de 0.05, por consiguiente se tiene que la zona de rechazo es por debajo de -1.680 como se observa en al **Figura 17**.

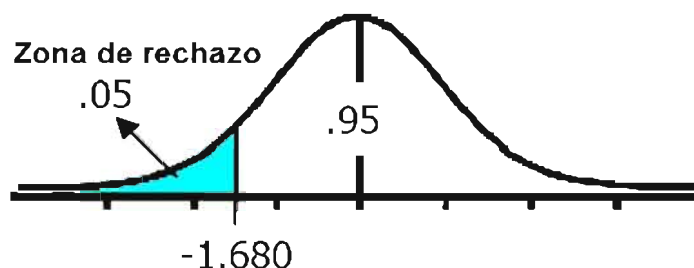


Figura 17. Prueba del estadístico t student [TAHA, 2000]

Se calcula el estadístico $t = (\bar{x} - \mu_n) / (s / \sqrt{n})$

Se obtuvo $t = 1,06295$, como se observa en al **Figura 17**, el valor obtenido no se encuentra en la zona de rechazo por lo cual se acepta H_1 y se rechaza H_0 , dado a que el valor promedio obtenido se aproxima al valor esperado que se quería obtener mediante el prototipo, por tanto queda demostrada la hipótesis planteada en la presente investigación.

Haciendo esta prueba de hipótesis podemos asegurar que con el uso de algoritmos genéticos siempre se obtendrá un porcentaje mayor a 93 por ciento, el cual se aproxima al resultado esperado que es el 95 por ciento con un margen de error del 5%. Por otro lado cabe mencionar que las instituciones que utilizan métodos tradicionales para la designación de infraestructura o de ambientes siempre estarán por debajo de 93 por ciento.



5

CONCLUSIONES Y RECOMENDACIONES

5. CONCLUSIONES Y RECOMENDACIONES

5.1. CONCLUSIONES

La utilización de algoritmos genéticos para la asignación de aulas ha mostrado un desempeño altamente eficaz en relación a la optimización de la designación de aulas. Esta eficacia se debe a que el modelo planteado pondera diversos aspectos durante el proceso de resolución del problema. El cromosoma utilizado contiene solamente la vinculación entre el espacio físico y el curso que se pretende designar, este tipo de representación flexibiliza el cómputo de la función de aptitud, manteniéndose por separado estructuras relacionadas con la capacidad, el acondicionamiento y la ubicación del aula.

En presente trabajo de investigación se demostró la aplicabilidad de los Algoritmos Genéticos para la designación de infraestructura académica, para ello se uso algunos modelos matemáticos, con lo cual se fundamenta una base teórica muy distinta a las demás. Otro aspecto importante del trabajo es la delimitación de un área de interés, el cual reduce el tiempo en la búsqueda de un determinado espacio áulico tomando esto en consideración es posible concluir que una adecuada delimitación de una región o área de interés mejora el rendimiento de un sistema de búsqueda.

Existen otros métodos para resolver problemas de designación de espacios áulicos, por ejemplo el recocido simulado, la búsqueda tabú, etc., algunos de los cuáles pueden ser rápidos en términos de encontrar buenas soluciones, sin embargo un AG se aproxima más a la designación que es realizada manualmente, no obstante, cuando esta designación es hecha a mano se puede tratar de balancear diferentes condiciones u objetivos, algunas de estas condiciones no pueden ser expresadas fácilmente en la función de evaluación. A pesar de esta desventaja un AG produce más de un resultado para una sola ejecución del mismo (sin incrementar el número

de generaciones) a diferencia de los métodos mencionados anteriormente que producen un único resultado.

Al inicio de este trabajo se planteó el objetivo de crear una herramienta que fuera útil en la designación de Infraestructura Académica de una determinada institución, aplicando para ello los Algoritmos Genéticos. Con la construcción del Sistema para la Designación de Infraestructura Académica, se considera que este objetivo fue alcanzado satisfactoriamente, aunque se presentó un problema de sobre especificación en el caso de las salas de cómputo ya que las materias a impartir rebasan en mucho el número de espacios disponibles y esto es solucionado utilizando condiciones que no pudieron ser representadas en la función de evaluación, por lo que se decidió incluir estas materias en la asignación de cursos en aulas, dejando al criterio de la persona que realiza la asignación de horarios manualmente, la adaptación de los resultados de este sistema a sus necesidades.

El Algoritmo Genético implementado en este trabajo de Tesis, reduce eficazmente el tiempo que toma elaborar una correcta asignación de horarios, ya que una asignación hecha manualmente puede tomar desde días hasta semanas, lo cual no es conveniente, sin embargo el AG propuesto solo tarda unos minutos en elaborar una asignación de cursos mejor a la actual.

5.2. RECOMENDACIONES

Como consecuencia del presente trabajo surgen algunos tópicos que pueden ser aplicadas a futuros trabajos relacionados a este tema en particular. En este sentido se plantean las siguientes recomendaciones:

Realizar un estudio sobre el tamaño de la población que debe utilizar un problema, ya que influye en el comportamiento de los Algoritmos Genéticos y en la ejecución del problema.

Emplear un modelo de interpretación del problema, ya que en el presente trabajo se utilizó tres espacios vectoriales aulas, días y horas respectivamente.

Se recomienda buscar otras formas de diseño de la estructura de cromosomas en este problema de tal manera que minimice el tamaño total del cromosoma.

Completar el trabajo para una institución real utilizando Algoritmos Genéticos, en vista de que no se tomo todas las exigencias que pueden existir en una determinada institución, ya que el objetivo era probar que si era posible utilizar esta metodología.





BIBLIOGRAFÍA

BIBLIOGRAFÍA

[C. Darwin, 1959] *Jhon Murray* "On the origin of species by of natural selection or the Preservations of favores races in the struggle for life" 2da. Edición, 1959.

[Coello, 2002] COELLO, C., Introducción a los algoritmos Genéticos. Universidad de tulane, Nueva Orleáns, EUA. 2002.

[DeJong, 2007] DeJong J. *Computación Evolutiva*. 4ta. Edición, 2000, New York: IEEE Press.

"Evaluación del comportamiento de los Algoritmos Genéticos" **[Fecha de visita: 30/03/2009]**

<http://www.geocities.com/SiliconValley/9802/3d5ca400.htm>

[Gastón Crevillén y David Díaz, 2001] Nexus 7, "Home page sobre Algoritmos Genéticos", 2001, **[Fecha de visita: 25/04/2009]**

<http://www.cinefantastico.com/nexus7/ia/ageneticos.htm>

[Gonzáles, 2003] Gonzáles F. Algoritmo Genético Simple. Presentaciones de Clase, 2da. Edición Bogota, Colombia. 2003

[Guervós, 2003] Guervós J. J. Merelo Informática Evolutiva. **[Fecha de visita: 15/03/2009]**

<http://kal-el.ugr.es/~jmerelo>

Hernández, Fernández & Baptista, METODOLOGIAS DE LA INVESTIGACION, 3ra. Edición, 2003, Ed. McGraw, 706 p.

[Herrán, 2002] HERRAN, M. 2002. Computación Evolutiva. **[Fecha de visita: 15/10/2008]**.

http://www.geocities.com/SiliconValley/Vista/749/ce_c.htm

[Jesús Alfonso López, 2000] Tripod, "Home page sobre Algoritmos Genéticos", 2000,

[Fecha de visita: 5/04/2009]

http://members.tripod.com/jesus_alfonso_lopez/AgIntro.html

[John Holland, 1999] Raúl Archuleta, "Home Page sobre Algoritmos Genéticos", 1999,

[Fecha de visita: 3/04/2009]

<http://www.geocities.com/archuleta>

[Koza, 1992] Koza, J.R. "Genetic Programming. On the Programming of Computers by Means of Natural Selection", The MIT Press, 1992, 819 p.

[Larrañaga, 2000] Larrañaga P., Algoritmos Genéticos y Computación Evolutiva,

[Fecha de visita: 3/11/2008].

http://www.Algoritmos_geneticos_y_computacion_evolutiva.htm

[Michalewicz, 1995] Michalewicz M. Genetic Algorithms + Data structures. Springer. 3ra. Edición. 1995

[Mitchell, 1997] Mitchell M. An Introduction to Genetic Algorithms. Massachusetts. 3ra. Edición. 1997

"Operadores genéticos" **[Fecha de visita: 15/03/2009]**

<http://www.iamnet.com/users/jcontre/genetic/operadores.htm>

[TAHA, H. 2000] H. Taha, *Investigación Operativa*. Alfa Omega. México. 5ta. Edición, 2000.

[Ucharico, 1999] Ucharico M. Modelo Difuso de Arranque para los Algoritmos Genéticos. Tesis de Licenciatura. Universidad Mayor de San Andrés. Informática. 1999

[Wikipedia, 2005] Algoritmos Genéticos. **[Fecha de visita: 15/10/2008].**

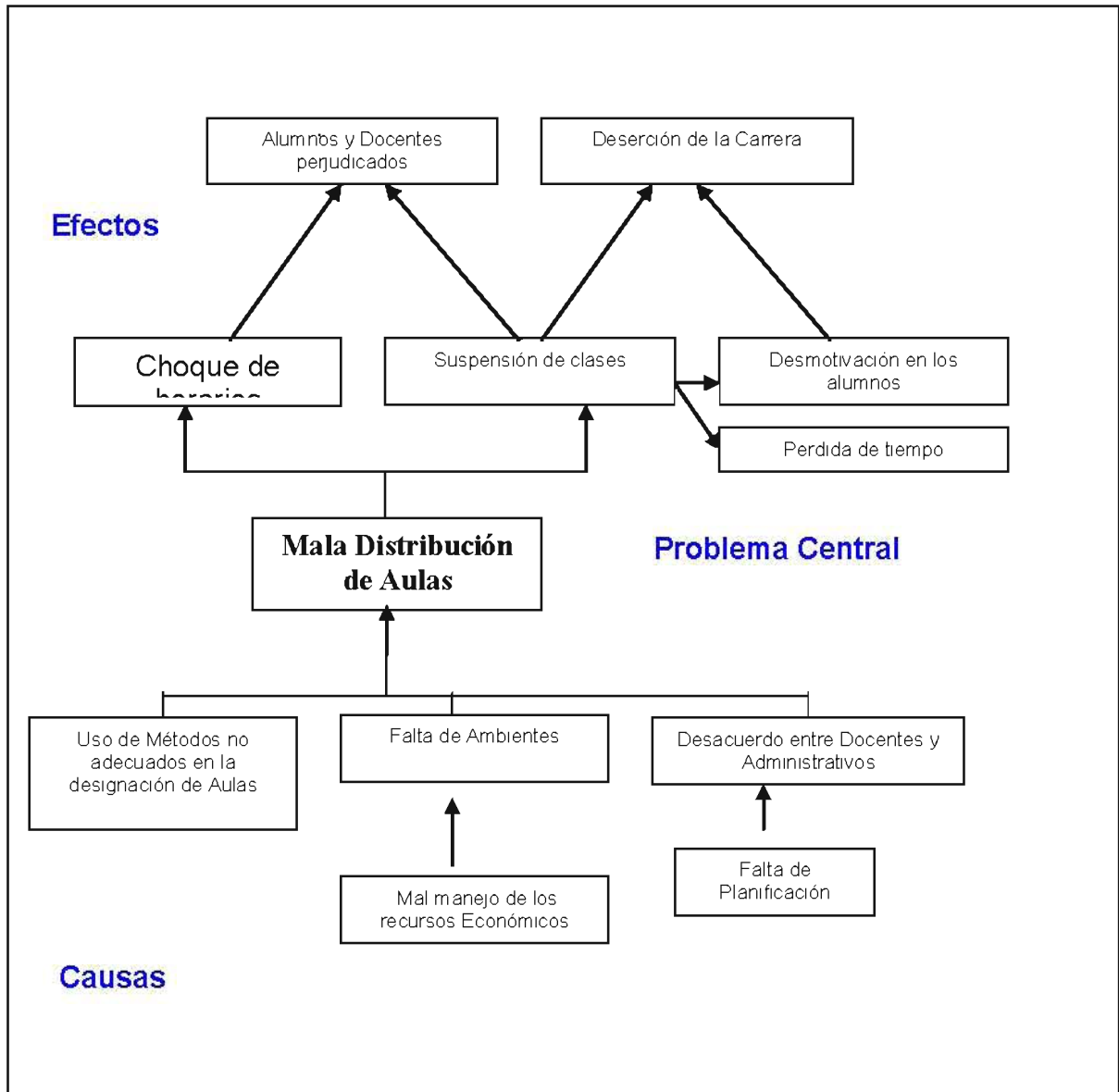
http://es.wikipedia.org/wiki/Algoritmo_geneticos



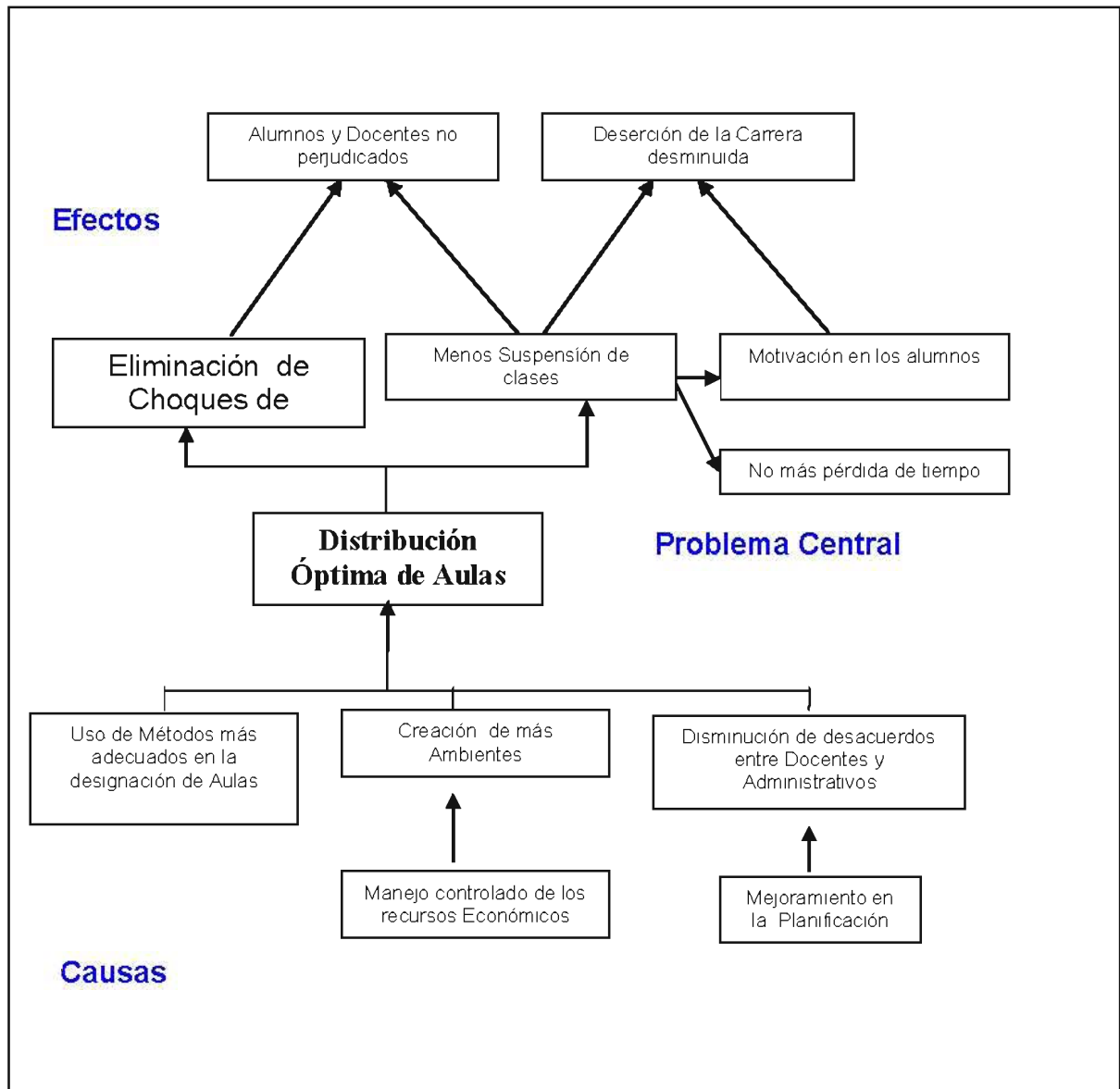
ANEXOS

Anexo A

ARBOL DE PROBLEMAS



ARBOL DE OBJETIVOS



Anexo B

Algoritmos Genéticos Paralelos

Nociones de Paralelismo

Podemos definir el **procesamiento en paralelo** como la ejecución concurrente (o simultánea) de instrucciones en una computadora. Dicho procesamiento puede ser en la forma de eventos que ocurran:

1. durante el mismo intervalo de tiempo
2. en el mismo instante
3. en intervalos de tiempo traslapados

La motivación mas obvia del paralelismo es el incrementar la eficiencia de procesamiento.

Existen muchas aplicaciones que demandan grandes cantidades de tiempo de procesamiento y que, por ende, resultan beneficiadas de contar con arquitecturas en paralelo.

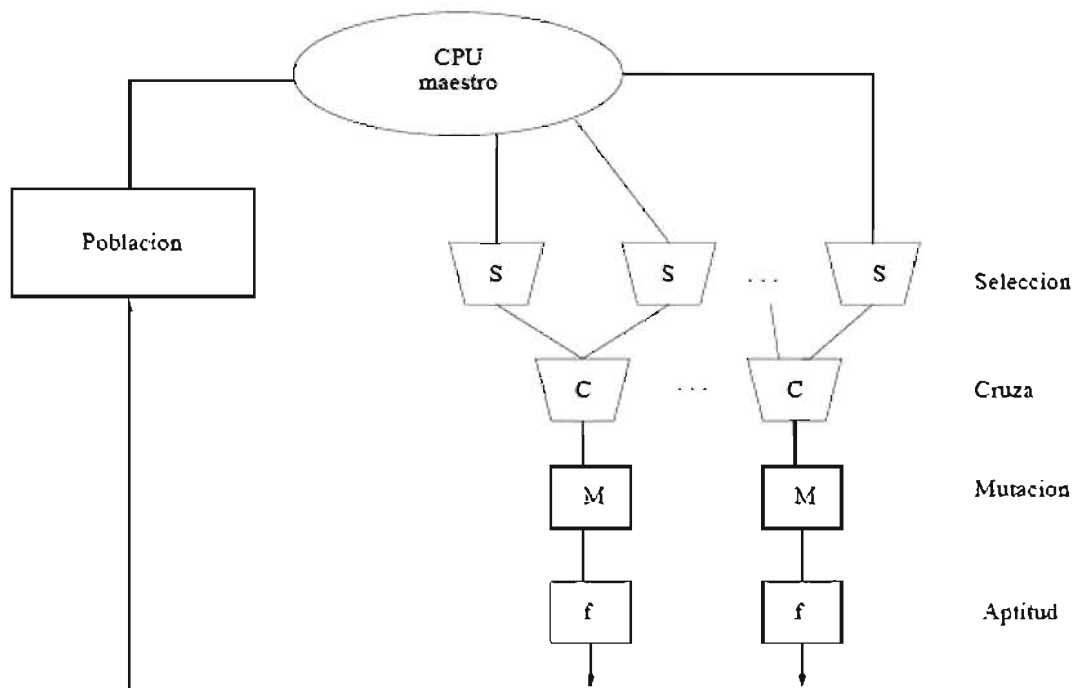
Una de las frecuentes confusiones respecto al paralelismo es que se cree que al contar con una computadora que tenga procesadores trabajando en el mismo problema, este podrá resolverse. Veces más rápido. Esto es falso.

Al usar varios procesadores para una misma tarea, debemos tomar en cuenta que existirán: Problemas de comunicación entre ellos. Conflictos al intentar acceder la memoria.

Algoritmos ineficientes para implementar el paralelismo del problema. Paralelismo del problema.

AGs Paralelos

Una vez revisados algunos conceptos básicos de paralelismo, procederemos a analizar los esquemas más comunes de paralelización de un algoritmo genético.



Esquema de paralelismo global de un algoritmo genético

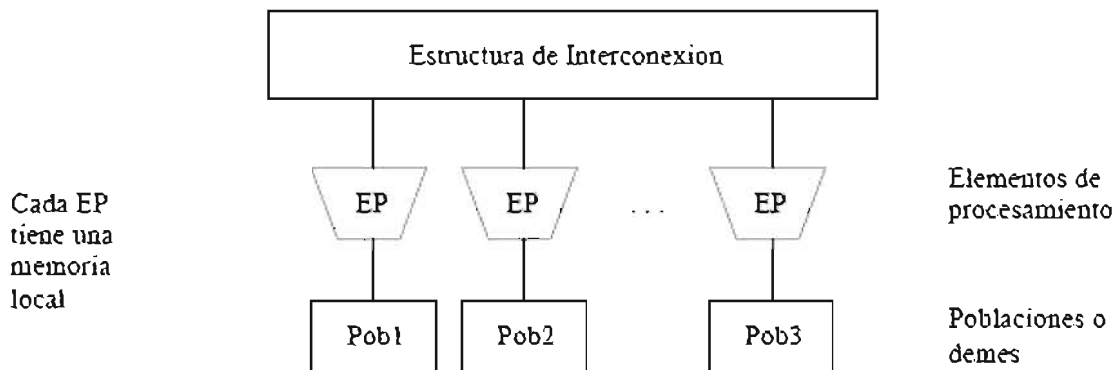
Paralelización global

El método más simple de paralelizar un AG es la llamada paralelización global. En este caso, sólo hay una población, como en el AG convencional, pero la evaluación de los individuos y los operadores genéticos se paralelizan de forma explícita.

Puesto que sólo hay una población, la selección considera a todos los individuos y cada individuo tiene oportunidad de aparearse con cualquier otro (o sea, hay apareamiento aleatorio). Por lo tanto, el comportamiento del AG simple permanece sin cambios.

La paralelización global es un método relativamente fácil de implementar y puede obtenerse un incremento significativo de velocidad si los costos de comunicación no dominan los costos de procesamiento. Una observación importante es que no debe confundirse el concepto de paralelismo implícito de un AG con el de paralelismo explícito.

A la paralelización global también se le conoce como AG panmítico, pues se cuenta con un solo depósito de material genético (*gene pool*), o sea con una **sola** población.



Esquema de funcionamiento de un algoritmo genético paralelo de grano grueso.

Los AGs panmíticos son útiles cuando el costo de evaluar la función de aptitud es elevado (por ejemplo, una simulación). En el AG panmítico no se requieren nuevos operadores ni nuevos parámetros y la solución encontrada será la misma que la producida con un AG convencional (o sea, serial).

Es importante hacer notar que aunque el paralelismo global normalmente es **síncrono** (o sea, que el programa se detiene y espera a recibir los valores de aptitud de toda la población antes de proceder a producir la siguiente generación), puede implementarse también de forma **asíncrona**, aunque en ese caso, su funcionamiento ya no resultará equivalente al de un AG convencional.

Además de paralelizarse la evaluación de la función de aptitud, en el paralelismo global es posible incluir también los operadores genéticos, pero dada la simplicidad de éstos, no suelen paralelizarse, pues los costos de comunicación dispararían cualquier mejora en el desempeño del programa.