UNIVERSIDAD NACIONAL DE INGENIERÍA

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

“SISTEMAS DE COMUNICACIÓN UTILIZANDO ALGORITMOS GENÉTICOS PARA OPTIMIZAR PROCESOS”

TESINA

PARA OPTAR EL TÍTULO DE SEGUNDA ESPECIALIZACIÓN PROFESIONAL EN LA ESPECIALIDAD DE TELEMÁTICA

PRESENTADO POR

OCTAVIO ALVAREZ MAURICIO

ASESOR

M. SC. JOSÉ P. MIGUEL CAÑAMERO

LIMA-PERÚ

2014

Le dedico esta Tesina a:

A mis padres Paulina y Griseldo.

A Dios.

El autor

AGRADECIMIENTOS

Yo deseo expresar mi sincero agradecimiento al señor M. Sc. José P. Miguel Cañamero, asesor de la Tesina, por su apoyo continuo y ayuda en la asesoría del tema. Por haber tenido paciencia y cooperación por toda investigación, sin ello, no habría sido posible llevé a cabo la disertación.

Mis agradecimientos a los señores Dr. Arturo Rojas Moreno, M. Sc. Ing. José Díaz Zegarra y M. Sc. Ing. Franz Peralta Alférez, para cada uno de los ingenieros, por haber dedicado su tiempo en la revisión y asesoría de la Tesina de “Sistemas de Comunicación Utilizando Algoritmos Genéticos Para Optimizar Procesos”.

Mis agradecimientos finales a todas las personas que han colaborado en la elaboración de la presente Tesina con su apoyo directo o indirecto a la investigación.

El autor

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN

EL desarrollo de la investigación de la tesina de “Sistemas de Comunicación Utilizando Algoritmos Genéticos Para Optimizar Procesos”, ha permitido llegar a las siguientes conclusiones:

Hay deficiencia en el tiempo de procesos de tareas por procesador del sistema; que están trabajando con algoritmos tradicionales. Por ejemplo, retardo en la cola de espera, retardo en tiempo de proceso de tareas, retardo en tiempo de respuesta, que ocasionan un efecto de cuello de la botella, falta de memoria, etcétera.

El problema central que se propone resolver está orientado al crecimiento y, evolución de una red de comunicaciones de una manera económica y escalable que lleve a un rendimiento óptimo. Por consiguiente, existe la necesidad de estudiar los procesos del sistema de comunicaciones en la red, de tal manera que se aplique otra alternativa, como “algoritmos genéticos” para optimizar procesos en el servidor, mejorar procesos del servidor web y, para mejorar la atención a los clientes / usuarios.

Se ha elegido la alternativa de “los algoritmos genéticos” para solucionar el retardo de procesos de tareas en el procesador del servidor. Se usan los algoritmos genéticos ya que por su evolución natural sirven para optimizar procesos de tareas del procesador.

Se ha implementado un simulador prototipo para sistema con algoritmos genéticos, que sirve optimizar los procesos en comunicaciones. Antes de procesar los datos se evaluarán procesos de tareas, número de proceso, probabilidad de cruce, probabilidad de mutación, en cola, atendidos, tiempo consumido, resultados de procesos óptimos. El tiempo de procesamiento de datos es menor que el tiempo de procesamiento de tareas convencional en el procesador, para mejor uso del recurso de la computadora.

ABSTRACT

The development of the thesis Research of "Communication Systems Using Genetic Algorithms to Optimize Processes", it has allowed to reach the following conclusions:

There is a deficiency in the process time of tasks per processor in the system they are working with traditional algorithms. For example, retard in wait queue, retard in time in processing of tasks, retard answer time, causing an effect of the bottleneck, out of memory, and so on.

The central problem is proposed to solve is oriented to growth and development of communication network in an economical and scalable leading to optimal performance. Therefore, there is a need to study the processes of the communication system on the network, so that alternative applies, as "genetic algorithms" to optimize server processes, improve web server processes to improve care customers / users.

It has chosen the alternative of "genetic algorithms" to solve the process retard of tasks in the server processor. Genetic algorithms are used as for its natural evolution they serve for to optimize, task processes of the processor.

We have implemented a prototype simulator for system with genetic algorithms, to optimize for the communications process. Before the data to process task processes are evaluated: number of process, crossover probability, mutation probability, in queue, served, time-consuming, optimal process results. The data processing time is less than the processing time for conventional data processor, for better use of computer resource.

INTRODUCCIÓN

El presente trabajo de investigación Tesina se ha denominado “Sistemas de Comunicación Utilizando Algoritmos Genéticos Para Optimizar Procesos”, ha identificado procesos de tareas en el procesador del servidor, en la cual hay deficiencia en el tiempo de procesos de tareas de procesador del sistema; que están trabajando con algoritmos tradicionales. Por ejemplo, retardo en la cola de espera, retardo en tiempo de proceso de tareas, retardo en tiempo de respuesta, que ocasionan un efecto de cuello de la botella, falta memoria, etcétera.

Para navegar por Internet, no sólo se necesita una computadora, un módem y algunos programas, sino también una gran dosis de paciencia. El ciberespacio es un mundo de acceso o espera al sistema. Un usuario puede pasar varios segundos o minutos esperando a que se cargue una página o varios minutos tratando de bajar un programa / datos de la web a su computadora del cliente / usuario.

Hoy en día, un sistema de comunicación ha mejorado la velocidad de transmisión de datos, debido a nuevas tecnologías de la información y comunicación (hardware, software, aplicaciones de comunicaciones, comunicaciones ópticas, nanotecnología electrónica, etcétera).

Por lo expuesto, el planteamiento del problema visto antes, se ha elegido la alternativa de “algoritmos genéticos”, para solucionar la mejora de procesos de tareas en el procesador del servidor.

Para obtener los resultados de trabajo de investigación, se ha utilizado la metodología del **sistema de información** (SI) y experimentalmente se ha implementado un simulador prototipo para un sistema de aplicación con algoritmos genéticos, para optimizar los procesos en comunicaciones. Antes de procesar los datos se evaluarán procesos de tareas tales como: número de proceso, probabilidad de cruce, probabilidad de mutación, en cola, atendidos, tiempo consumido, resultados de procesos óptimos. El tiempo de procesamiento de datos es menor que el tiempo de procesamiento de datos que es convencional en el procesador, mejor uso del recurso de la computadora.

CAPÍTULO 1

GENERALIDADES

1.1 Antecedentes del problema

Al inicio de la era de la computación e informática, las computadoras eran grandes y caras. Debido a su escasez y costo, éstas funcionaban de forma independiente entre ellas.

A partir de 1970, surgen las primeras minicomputadoras que competirían con las grandes computadoras, como asistencia técnica y por su precio, con ello se extendió su uso. Los grandes sistemas centralizados fueron dejando paso lentamente a sistemas mucho más descentralizados, y organizados por varias computadoras o a los sistemas multiprocesadores. Pronto surgieron nuevas necesidades de interconexión de los equipos, como computadoras personales y las utilizan para estaciones de trabajo en la red, se desarrollaron además la **red de área local** (LAN), como Ethernet o Token ring, etcétera.

Hoy en día, la Internet es la red de mayor tamaño y la más utilizada, y continúa un impresionante ritmo de crecimiento. Además, la Internet es la base de muchos proyectos nuevos para procesos de sistemas de aplicaciones en el servidor web.

Aunque los actuales sistemas en la red solucionan parte de las necesidades actuales de comunicación entre computadoras, aún presentan importantes limitaciones, y no son aplicables a una gran cantidad de problemas. Por ello, surge la necesidad de crear sistemas de comunicación para optimizar procesos.

Los procesos de sistemas de aplicaciones están basados en las ideas básicas de transparencia, eficiencia, flexibilidad, escalabilidad y fiabilidad. Sin embargo, estos aspectos son parte contrario de algunos casos; y por lo tanto, los procesos de sistemas de aplicaciones han de cumplir en su diseño el compromiso de que todos los puntos anteriores sean solucionados de manera aceptable.

1.2 Descripción de la realidad problemática

Hoy en día, en los procesos de tareas del servidor se presentan deficiencias en la realización de tareas específicas de: asignación de procesos al sistema, que están utilizando algoritmos tradicionales tanto para el sistema operativo como para la

comunicación. Por ejemplo, el tiempo de espera para procesar datos, el tiempo de respuesta, falta memoria, etcétera.

a) Delimitaciones

Se hace una breve descripción de delimitación de la investigación de la Tesina de la Segunda Especialización, una vez implementado el simulador prototipo utilizando algoritmos genéticos para optimizar procesos de tareas en el servidor web, se obtiene las siguientes limitaciones:

- Existe un tiempo limitado para el proceso de asignación de tareas por simulador.
- Los valores generados por el nuevo requerimiento son aleatorios.
- Existe un valor limitado para las iteraciones en los algoritmos genéticos.
- Existe un valor limitado para probabilidad de cruzamiento.
- Existe un valor limitado para mutación.
- Existe un valor limitado de número de alelos.

1.3 Planteamiento del problema

Hoy en día, en un sistema de comunicaciones, se han identificado procesos de tareas en el procesador del servidor, en el cual hay deficiencias en tiempo de los procesos de datos que están trabajando con algoritmos tradicionales.

Por ejemplo, retardo en la cola de espera, retardo en tiempo de proceso de tareas, retardo en tiempo de respuesta, que ocasionan un efecto de cuello de botella y falta de memoria.

El problema central que se propone está orientado al crecimiento y, evolución de red de comunicaciones de una manera económica y escalable que lleve a un rendimiento óptimo. Por consiguiente, existe la necesidad de estudiar multiprocesos en paralelo del sistema de comunicaciones de red, de tal manera que se aplique otra alternativa como los algoritmos genéticos para optimizar procesos en el procesador, mejorar procesos en el servidor web y, mejorar servicios a los clientes / usuarios.

Alternativas de solución al problema son:

a) Redes neuronales.

b) Algoritmos genéticos.

Se ha elegido la alternativa de algoritmos genéticos, para mejorar los tiempos de proceso de las tareas en el procesador del servidor. Los algoritmos genéticos, por su evolución natural sirven para optimizar proceso de tareas en el procesador del servidor. Se desarrollará un simulador prototipo para optimizar procesos de tareas en el servidor. Los algoritmos genéticos son descritos en el marco teórico.

1.4 Justificación e importancia

La presente tesina titulada “Sistemas de Comunicación Utilizando Algoritmos Genéticos Para Optimizar Procesos”, permitirá mejorar el tiempo de procesamiento de datos para optimizar procesos de datos / tareas de eventos concurrentes en el procesador del sistema, y las estaciones de trabajo de computadoras.

La implementación del simulador prototipo con algoritmos genéticos para sistema de aplicación de procesos de tareas (datos) en el procesador permitirá: recepción de paquetes equitativos en la computadora, no habrá paquetes almacenados mucho tiempo en computadoras, disminuirá el tiempo de retardo de procesos datos en la computadora, disminuirán costos de recursos computacionales del servidor, disminuirá tiempo de respuesta de mensajes del sistema, entonces, mejorará el uso de recursos de la computadora, etcétera.

Las pruebas se realizaron en el Laboratorio de Computación de la Sección de Posgrado y Segunda Especialización de la Facultad de Ingeniería Eléctrica y Electrónica de la Universidad Nacional de Ingeniería.

1.5 Objetivos

a) Objetivo general

Desarrollar un simulador prototipo que utiliza algoritmos genéticos para optimizar procesos de tareas en el procesador del servidor, a fin de mejorar el/los tiempo(s) utilizado(s) del procesador del sistema y mejorar la atención a los clientes / usuarios.

b) Objetivos específicos

- Analizar procesos de tareas clientes / usuarios en el servidor.
- Aplicar algoritmos genéticos a un simulador propuesto.
- Implementar simulador prototipo utilizando algoritmos genéticos para optimizar el procesamiento de datos en el procesador del servidor.
- Analizar resultados de procesos de datos.
- Disminuir tiempo de espera de las colas del servidor.
- Disminuir tiempo de retardo de procesos de datos en la computadora.
- Disminuir costos de recursos computacionales del servidor.
- Mejorar tiempo de respuesta de mensajes del sistema.

1.6 Hipótesis

1. Nuestra propuesta, que constituye la hipótesis del trabajo de investigación, es:
   - Es posible agilizar el procesamiento de datos a través de algoritmos genéticos.

- Es posible la construcción de un sistema que pruebe la utilización de algoritmo Genético en esta aplicación.

2. En qué medida el sistema de aplicación con algoritmos genéticos en procesos del servidor, mejorará el/los tiempo(s) utilizado(s) de procesos de datos en procesador del servidor y mejorará la calidad de servicios para atender a los clientes / usuarios, tales como:

a) Variable independiente

Simulador por computadora utilizando algoritmos genéticos para optimizar procesos en el procesador del servidor de: rendimiento, carga, recursos, tiempo de respuesta, velocidad de transferencia, etcétera.

b) Variable dependiente

Optimización de procesos de tareas concurrentes en el procesador del servidor.

1.7 Metodología

Metodología de investigación experimental es un propósito definido para guiar paso a paso el desarrollo de la tesina de titulación, sobre cómo realizar investigaciones científicas.

Metodología de investigación del **sistema de información** (SI) y experimental se incluye cliente / servidor utilizando lenguaje de programación C ++ de Borland Developer Studio 2006, para desarrollar el simulador prototipo de aplicación utilizando algoritmos genéticos para optimizar procesos de tareas de eventos concurrentes en el procesador de la tesina propuesta, a continuación algunas consideraciones:

1. Se ha elegido la metodología del sistema de información experimental para procesos en el procesador del servidor, lo cual es desarrollada en el punto de trabajo considerando la secuencia de pasos que se van a seguir y los detalles correspondientes.
2. La metodología de algoritmos genéticos es la base principal para implementación del simulador prototipo del sistema para optimizar procesos y datos en el procesador del servidor.
3. La población es un conjunto de individuos (cromosomas) de algoritmos genéticos. En el sistema se toman cromosomas fuertes para optimizar procesos de tareas en el procesador; para ser implementada y probado en el servidor.

CAPÍTULO 2

MARCO TEÓRICO

2.1 Internet

2.1.1 Evolución de la Internet

La evolución de Internet empezó a finales de 1960. La Unión Soviética había lanzado el satélite Sputnik en 1957, y en plena guerra fría los Estados Unidos de América querían asegurar su posición a la cabeza de la tecnología militar. Entonces, el **Departamento de Defensa** (DoD) de los Estados Unidos, se dio cuenta de que la tecnología empleada por la red telefónica, llamada conmutación de circuitos, era demasiado frágil para resistir el más mínimo ataque y mucho menos la tan temida guerra nuclear. Si se destruía una conexión entre dos centrales importantes o resultaba una central fuera de servicio, alguna parte de las telecomunicaciones de defensa del país podrían quedar inutilizadas.

Posteriormente, como en el caso de muchas otras tecnologías, Internet y las redes de conmutación de paquetes se desarrollaron inicialmente gracias al financiamiento y apoyo del gobierno de los Estados Unidos de América. La **Oficina de Proyectos de Investigación Avanzada** (ARPA) de los Estados Unidos, fue una de las primeras instituciones que adoptó la teoría de conmutación de paquetes. ARPA creó lo que fue llamado ARPAnet como una red importante de computadoras del gobierno capaces de resistir daños a la red producidos por una guerra o una catástrofe severa. Con la colaboración de varias compañías y universidades los esfuerzos iníciales de ARPA culminaron en septiembre de 1969, cuando fue entregada a la **Universidad de California en Los Ángeles** (UCLA) una minicomputadora Honeywell 516. Este sistema fue el primero de cuatro conmutadores de paquetes, también conocido como **procesadores de mensajes con interfases** (IMP).

Otros tres conmutadores de paquetes fueron instalados en la Universidad de Utah, la Universidad de California en Santa Bárbara y el Instituto de Investigación de Stanford. Muy pronto, estas computadoras intercambiaron paquetes entre sí a través de líneas telefónicas, con lo que nació la “madre de Internet”: ARPAnet. La figura 2.1 muestra los cuatro sitios originales de ARPAnet [18].

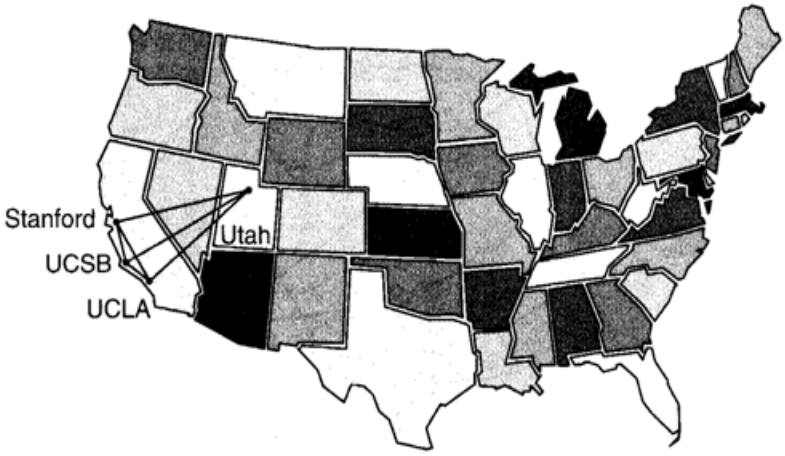


Figura 2.1. La ARPAnet original.
Fuente [18].

Hoy en día, la tecnología de Internet es sofisticado en cuanto a hardware y software, corporativos en el ámbito mundial, proliferación de clientes, mensajes, comunicaciones, etcétera. Esta variedad de computadoras son mostradas en la figura 2.2.

Figura 2.2. Redes de computadoras en la Internet.
Fuente [13].

i) Desarrollo de la Internet 2

Los objetivos mencionados anteriormente son llevados a cabo mediante actividades de desarrollo y prueba de nuevos protocolos y aplicaciones para Internet 2. Estos desarrollos son hechos en comités llamados **grupos de trabajo** (WG). Cada WG pertenece a alguna área técnica del desarrollo de Internet 2 de ingeniería, Middleware (interfaz software que provee funcionalidades rutinarias en una conexión típica Internet).

Entre estas, por ejemplo, se pueden mencionar las autenticaciones del usuario y aplicaciones. Cada una de estas áreas posee un director de área responsable de sus actividades. Los miembros de estos grupos de trabajo pueden ser tanto miembros de Internet 2 como empresas de apoyo externo (por ejemplo, las empresas de apoyo económico).

Si un miembro de Internet 2 tiene alguna idea a desarrollar entonces se debe contactar al director de área apropiada.

Los actuales grupos de trabajo por área, son:

Ingeniería. IPv6, Measurement, Multicast, Network Management, Routing, Security, Topology.

Middleware. MACE-Architecture, MACE-DIR (Directories), HEPKI-TAG (PKI Technical), HEPKI-PAG (PKI Policy).

Aplicaciones. Las artes e iniciativa de las humanidades, imagen digital, iniciativa del vídeo digital, almacenamiento en la red, iniciativa de ciencia de la salud, canal de investigación, conferencia en vídeo (iniciativa subcomité de vídeo digital), voz en IP.

Las dos primeras áreas tienen labores que son transparentes al cliente / usuario y que solo sirven para ofrecer un mejor servicio a las aplicaciones del área de aplicaciones. A partir de los nombres de los grupos de trabajo del área de aplicaciones uno puede deducir a grandes rasgos de qué se trata. Por ejemplo, en el grupo de trabajo de red de almacenamiento, se desarrolla la **infraestructura de almacenamiento distribuido en Internet 2** (**I2-DSI**: Internet 2 Distributed Storage Infrastructure). El objetivo de esto es el almacenar datos replicados a través de la red y cuando un cliente intente acceder a los datos entonces el sistema le provea los datos que se encuentran en el servidor más cercano (on) a él, manteniendo así el tráfico lo más local posible.

a) Red Internet 2

La red Internet 2 está compuesta por redes principales o troncales en USA, a los cuales se conectan los llamados gigaPoPs y troncales internacionales los cuales a su vez se conectan gigaPoPs o nodos en particular a las universidades. Un gigaPoP es una red regional (con ancho de banda del orden de los gigabits por segundo) conectada a la Internet 2. Por ejemplo, en USA el MIT, la Universidad de Boston y la Universidad de Harvard conforman la capacidad **Gigabit en presencia de punto** (gigaPoP) llamado BOS [19].

A continuación en la figura 2.3 se muestra el esquema generalizado de la Internet 2 siguiente:

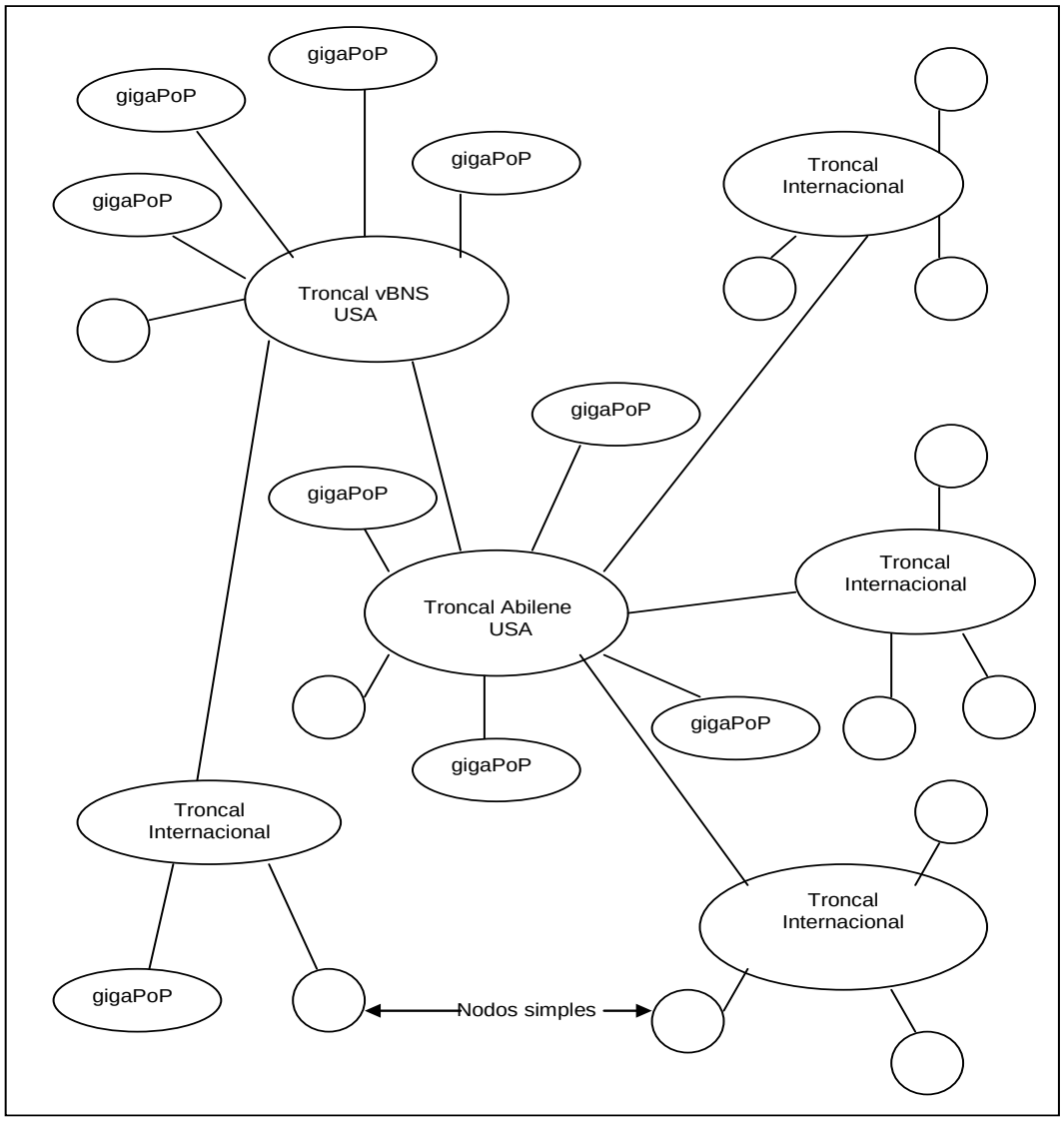


Figura 2.3. Esquema generalizado de la Internet 2.
Fuente [19].

En la figura 2.3 se puede visualizar que actualmente existen dos grandes troncales en USA (aunque hoy en día el troncal Abilene es mucho mayor en ancho de banda, 2,4 Gbps o superior), de los cuales se distribuyen enlaces hacia troncales en otros países. Uno de estos troncales internacionales es la **Red Universitaria Nacional** (RUN).

Para conexión a la Internet 2, se es necesario algunos nuevos dispositivos o equipos y algunas nuevas conexiones por lado de los clientes / usuarios de las respectivas universidades conectadas a la Internet 2. Los troncales son los responsables de encaminar el flujo de datos por Internet 2 o Internet comercial según correspondan.

b) Internet 2 desplazará al internet comercial actual

El proyecto Internet 2 reemplazará paulatinamente a la Internet actual. Su objetivo es unir a las instituciones con los recursos para desarrollar nuevas tecnologías y posibilidades que posteriormente puedan extrapolarse a la Internet global. Las

universidades mantendrán y continuarán teniendo un crecimiento sustancioso en el uso de las conexiones existentes de la Internet, que podrán seguir obteniendo de sus proveedores comerciales.

Aún más, el sector privado se beneficiará con las aplicaciones y tecnología desarrolladas por los miembros de Internet 2. Hoy en día, las universidades e institutos de investigaciones han hecho inversiones y esfuerzos considerables encaminados a conectar la mayor parte de sus instalaciones a la Internet comercial; dicha inversión y esfuerzo no puede ser despreciado.

c) Arquitectura de Internet 2

La red Internet 2 consiste en múltiples infraestructuras interconectadas, cada uno con arquitecturas independientes. Las infraestructuras son utilizadas por Internet 2 para proporcionar servicios de red a los clientes / usuarios. La red Internet 2 proporciona las siguientes clases de servicios: servicio IP y servicio del circuito. El servicio IP es una red IP que sustituye y engrandece la red de Abilene; y el servicio de circuito proporciona los circuitos punto a punto entre los puertos en la red Internet 2. Estos circuitos proporcionan ancho de banda garantizado y, comportamiento determinado desde el punto de vista de límites del Jitter, retardo y pérdida de paquetes. Se proporcionan tres diversos servicios del circuito: 1) Servicio estático del circuito. 2) Servicio dinámico del circuito. 3) Servicio de prueba HOPI.

Hay cuatro infraestructuras arquitectónicas marcadas en la red Internet 2, según las siguientes:

1. Infraestructura base, consistiendo en los segmentos de fibra interconectados con el equipo de señales.
2. Red IP.
3. Infraestructura de conmutación multiservicio.
4. Infraestructura HOPI testbed.

De la figura 2.4 es un diagrama de la arquitectura total de la red Internet 2. En el despliegue inicial, cada servicio será proveído por una infraestructura particular. Observar que este diagrama no muestra todos los nodos y conectores, sin embargo, muestra como los diferentes servicios interactúan arquitectónicamente. También demuestra algunas de las semejanzas y de las diferencias entre los servicios [13]. Se muestra en la figura 2.4 arquitectura de Internet 2 siguiente:

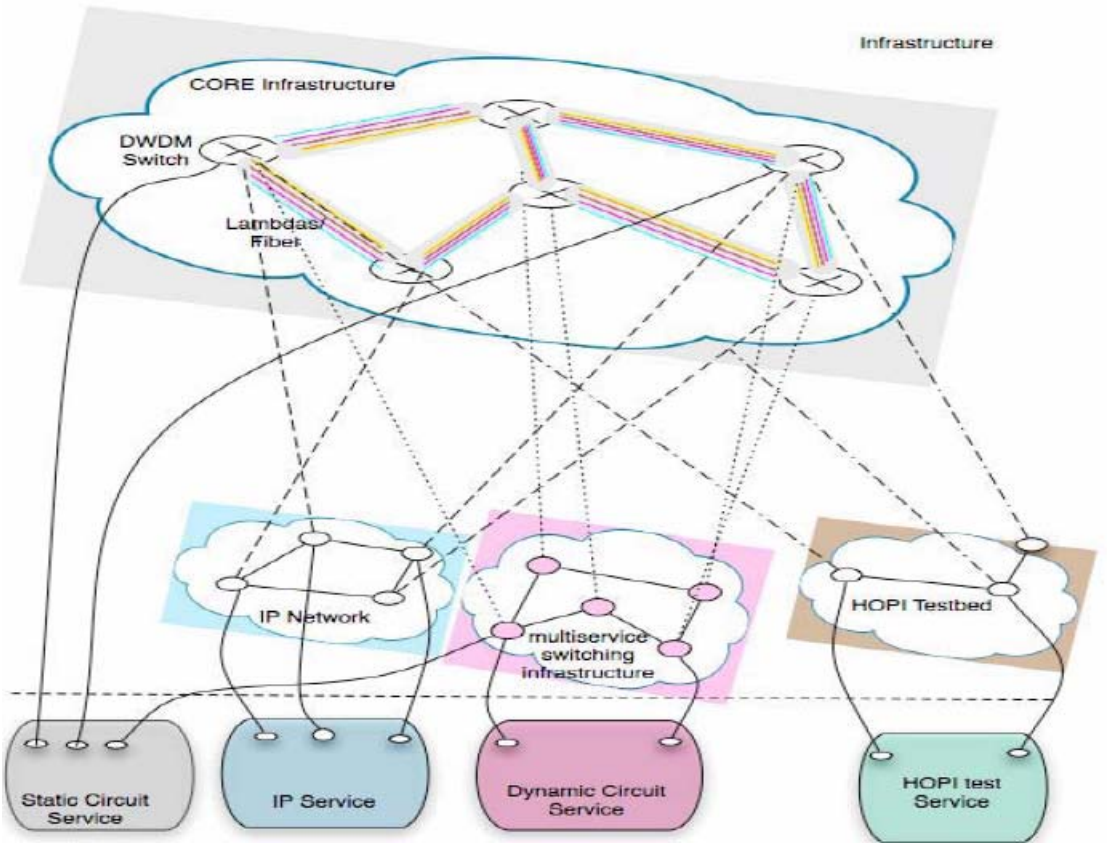


Figura 2.4. Arquitectura de Internet 2.
Fuente [19].

d) Instituciones educativas que no son miembros de la Internet 2

La participación en la Internet 2 está abierta para cualquier universidad que se comprometa a proveer facilidades para el desarrollo de aplicaciones avanzadas en su campus. La inversión financiera requerida para cumplir con estas obligaciones puede ser más de lo que muchas instituciones puedan permitirse por ahora. Sin embargo, la Internet 2 tiene la intención de acelerar la transmisión de nuevas posibilidades a la comunidad mayor del sistema de redes. El costo de la tecnología utilizada y desarrollada por Internet 2 descenderá a un nivel alcanzable para cualquier institución que actualmente tenga una conexión básica a la Internet.

2.2 Sistema de comunicación de datos

2.2.1 Introducción

Se hace una breve descripción del sistema de comunicación de datos. Al intercambio de información entre computadoras se le llama comunicación entre computadoras y al conjunto de esas computadoras que se interconectan se les denomina red de computadoras. El estudio de las comunicaciones contempla la denominada transmisión de señales de tal forma que esta sea eficaz y segura. En este trabajo veremos, algunas

de: transmisión de datos y codificación de señales, medios de transmisión, interfases y control del enlace de datos. Además, en principio se debe considerar:

- Existen diferencias entre el procesamiento de datos en las computadoras y comunicación de datos (la transmisión y sistema de conmutación de paquetes).
- Hay algunas diferencias fundamentales entre la transmisión de datos, de voz o de video.
- Las fronteras entre computadoras de monoprocesador o multiprocesador, así como entre **red de área local** (LAN), **red de área metropolitana** (MAN) y **red área amplia / extensa** (WAN) son cada vez más difusas.

El objetivo principal de todo sistema de comunicación es el de intercambiar información entre dos o más computadoras. En todos los sistemas de comunicaciones tienen por transmisión de información de usuario / cliente de origen hasta el destino o viceversa.

2.2.2 Sistema de comunicación de datos a través de redes

a) Introducción

En esta sección, se hace una breve descripción sobre el sistema de comunicación de datos a través de la web, transmisión de datos o conmutación de paquetes entre una computadora a otra.

Por ejemplo, a veces no es práctico que dos dispositivos de comunicación se conecten directamente mediante un enlace punto a punto. Esto es debido a alguna de las siguientes circunstancias:

- Los dispositivos están muy alejados. En este caso no será justificado, por ejemplo, utilizar un enlace dedicado entre cada dos dispositivos que puedan estar separados por miles de kilómetros.
- Hay un conjunto de dispositivos que necesitan conectarse entre ellos en instantes de tiempo diferente, [9] como se muestra en la figura 2.5.

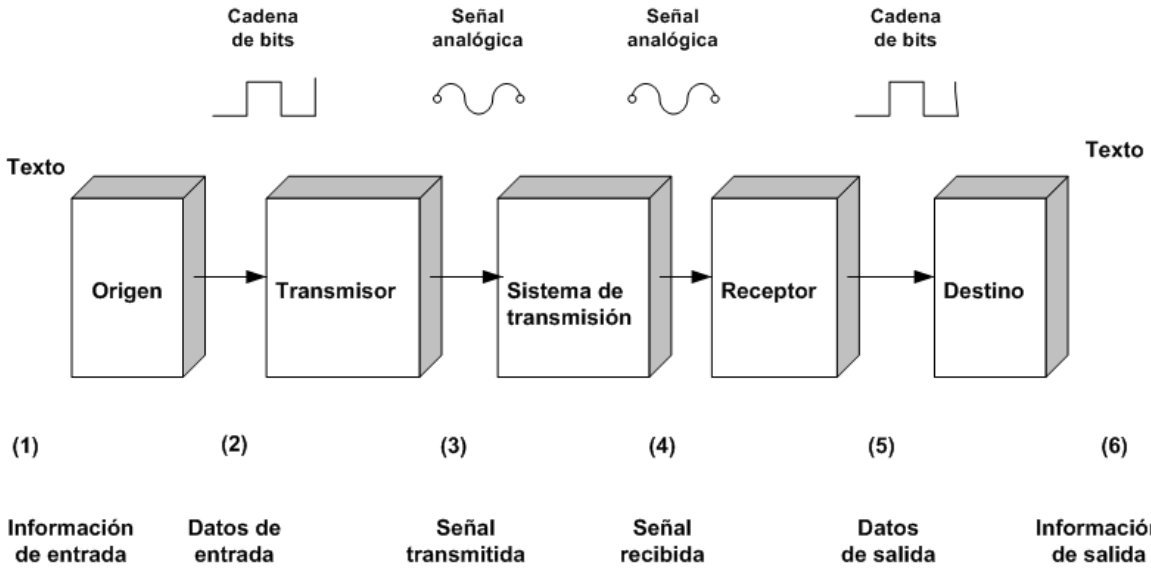


Figura 2.5. Modelo simplificado para comunicaciones de datos. Fuente [9].

b) Ventajas y desventajas en sistemas de comunicación de datos

i) Ventajas

- Fácil transmisión de datos en hardware y software heterogéneos.
- Fácil codificación de datos para transmisión (datos, audio, vídeo).
- La ventaja principal de la transmisión digital es la inmunidad al ruido. Las señales analógicas son más susceptibles que los pulsos digitales a la amplitud no deseada, frecuencia y variaciones de fases.
- Se prefieren a los pulsos digitales por su mejor procesamiento y multicanalizaciones que las señales analógicas. Los pulsos digitales pueden guardarse fácilmente, mientras que las señales analógicas no pueden.
- Los sistemas digitales utilizan la regeneración de señales, en vez de la amplificación de señales, por lo tanto producen un sistema más resistente al ruido que su contraparte analógica.
- Las señales digitales son más sencillas de medir y evaluar.
- Los sistemas digitales están mejores equipados para evaluar un rendimiento de error (por ejemplo, detección y corrección), que los sistemas analógicos.
- Tecnología digital. Tecnología de bajo costo de integración a escala grande (LSI: large scale integration) / integración a escala muy grande (VLSI: very large scale integration).
- Integridad de datos. Mayores distancias sobre líneas de más baja calidad.
- Utilización de la capacidad. Enlaces económicos de alto ancho de banda y mayor grado de multiplexación del canal.

- Seguridad y privacidad. Encriptación.
- Integración. Puede tratar de forma similar datos analógicos y digitales.
- Movilidad (inalámbrica).
- Facilidad de instalación (inalámbrica).
- Flexibilidad (inalámbrica).
- Escalabilidad (inalámbrica).
- Alta velocidad de transmisión de datos (fibras ópticas).
- Mayor ancho de banda (aproximadamente 4 órdenes de magnitud en fibra óptica).
- Baja atenuación (aproximadamente dos órdenes de magnitud en fibra óptica).
- Inmunidad a ruido eléctrico (fibra óptica), etcétera.

ii) Desventajas

- No es fácil decodificar los datos recibidos.
- Transmisión de datos a corta distancia (inalámbrica).

c) Tipos de redes

Para estructurar los conceptos de redes de computadoras se ha desarrollado un amplio conocimiento que cubre desde la abstracción de la arquitectura hasta la visión sistémica, pasando por los fundamentos tecnológicos y la modelación matemática. Las redes pueden analizarse desde las ópticas siguientes:

- Arquitectura, es decir, la definición abstracta de sus servicios y protocolos.
- Características físicas y lógicas, como topología, sistemas de transmisión, acceso, conmutación y medios de transmisión.
- Modelos matemáticos del comportamiento de la red, para evaluar parámetros de calidad [9].

Una clasificación de tipos de redes actuales, atendiendo a su cobertura, son:

i) Redes de área local (LAN).

ii) Redes de área metropolitana (MAN).

iii) Redes de área extensa / amplia (WAN).

iv) Redes inalámbricas.

v) Red Internet.

Vi) Red de área personal.

vii) Redes sociales.

d) La arquitectura de protocolo control de transmisión / protocolo de Internet

A finales de 1960, el Departamento de Defensa de los Estados Unidos creó la ARPANET

para poder investigar la comunicación de paquetes. La **Agencia de Proyectos Avanzados de Investigación Para la Defensa** (DARPA: Defense Advanced Research Projects Agency) fue el organismo que promovió el desarrollo de **Oficina de Proyectos Avanzados de Investigación Para Red** (ARPANET), red que llegó a interconectar en 1969 bases militares, centros de investigación, universidades y laboratorios gubernamentales.

El **Protocolo Control de Transmisión / Protocolo de Internet** (TCP/IP) se adoptó en 1982, como estándar en ARPANET, que fue referenciado por TCP/IP. Se ha suministrado implementaciones con las versiones 4.1 y 4.2 BSD de UNIX, lo cual facilitó su expansión.

TCP/IP se adoptó en 1983 como el estándar en ARPANET. A su vez, una segunda red llamada MILNET, dedicada exclusivamente a aspectos militares, se separó de ARPANET. Así empezó a formarse la red INTERNET. Actualmente el TCP/IP se considera como el conjunto de protocolos abiertos, no específicos de un fabricante determinado, y es el más extendido, con lo que se ha convertido en un estándar de facto, compatible para la mayoría de los fabricantes de sistemas operativos más extendidos.

e) Sistema de comunicaciones en la Internet

El sistema de comunicaciones en la Internet está formado por múltiples redes de computadoras interconectadas entre sí a través de elementos denominados enrutadores, receptores, módem, etcétera. En el contexto de Internet, el término enrutador o encaminador. El puente tiene una interfaz con la red de paso de testigo y otro con la red de conmutaciones de paquetes. Se muestra en la figura 2.6.

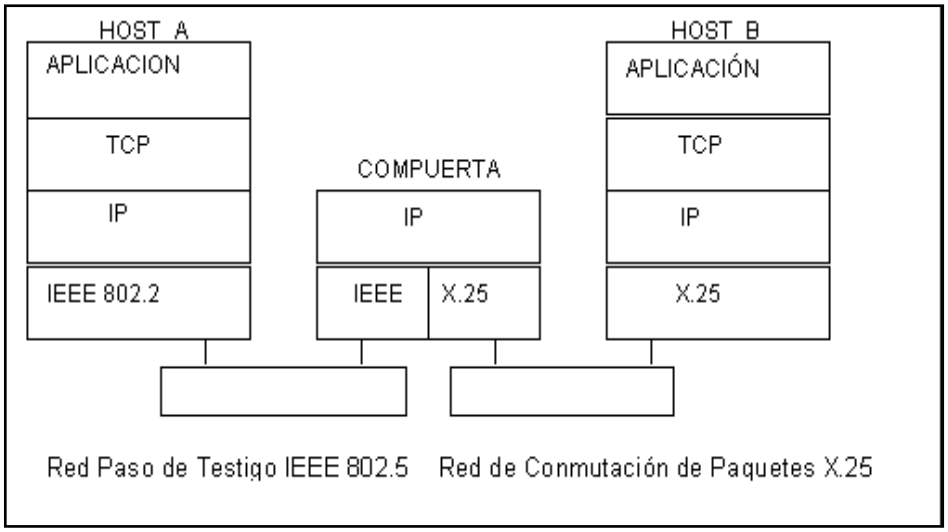


Figura 2.6. Comunicación de dos nodos en una red de Internet.
Fuente [9].

2.3 Sistema de transmisión de datos

El objetivo principal de todo sistema de transmisión de datos es el de intercambiar información entre dos o más entidades. Todos los sistemas de comunicaciones transmiten datos del usuario / cliente de origen al destino y viceversa. Esto se ilustra en las figuras 2.7, 2.8, 2.9.

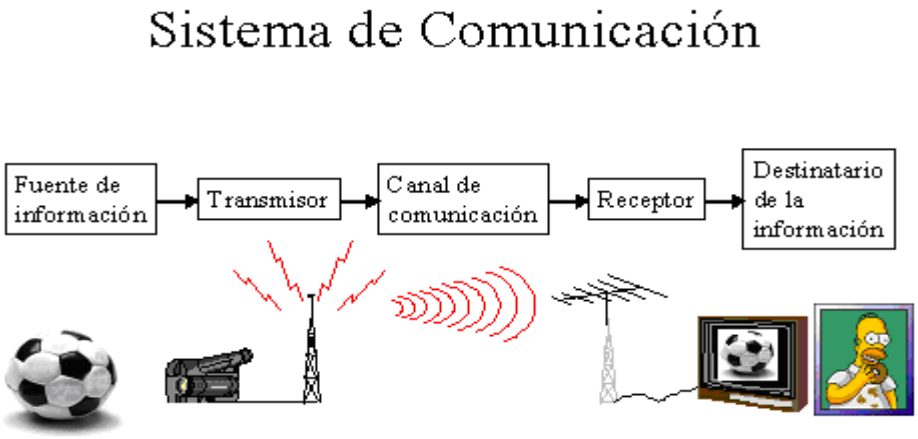


Figura 2.7. Sistema de transmisión y comunicación de datos.
Fuente [11].

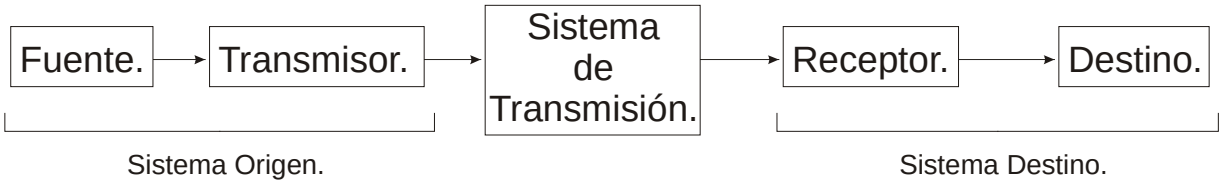


Figura 2.8. Sistema de transmisión de datos.
Fuente [7].

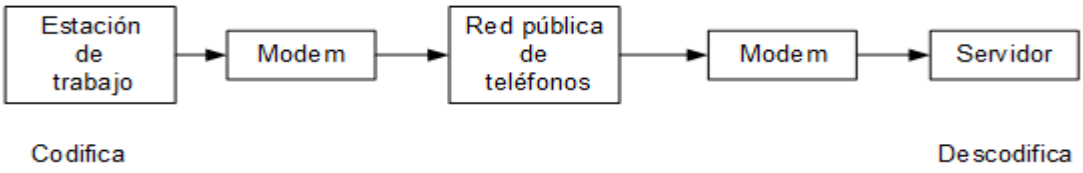


Figura 2.9. Sistema de comunicaciones de telefonía.
Fuente [8].

Se define **estación de trabajo** (Workstation) a cualquier elemento cuya arquitectura permite la comunicación con él y al exterior. Este término solo se aplica a computadoras [6]. A continuación se explica los elementos principales mostrados en las figuras anteriores:

a) **La fuente** (u origen). Es el dispositivo que genera los datos para transmitir, por ejemplo, teléfonos o computadoras personales.

b) **El transmisor**. Es común que los datos generados por la fuente no sean transmitidos

de forma directa y como fueron creados, sino que el transmisor transforma y codifica la información, generando señales electromecánicas susceptibles de ser transmitidas a través de algún sistema de transmisión. Por ejemplo, un módem que convierte las cadenas de bits generadas por una computadora personal y las transforma en señales analógicas que pueden ser trasmitidas a través de la red de teléfonos.

c) **El sistema de transmisión**. Este puede ser una línea sencilla de transmisión o bien, una red compleja que conecta el origen al destino, esto es, el medio físico por donde se envía la señal.

d) **El receptor**. Este acepta la señal proveniente del sistema de transmisión y la transforma de forma que pueda ser manejada por el dispositivo de destino, por ejemplo, un módem es capaz de captar la señal analógica en la red o línea de transmisión y la convertirá en una serie de bits.

e) **Destino**. Es el que toma los datos del receptor.

Las tareas principales que deben realizar cualquier sistema de comunicación, son:

1) Utiliza sistema de transmisión e implementación de la interfaz.
2) Generación de la señal.
3) Sincronización de tiempo necesario en el intercambio de información.
4) Gestión del intercambio de información.
5) Detección y corrección de errores en la transmisión de información.
6) Control del flujo de datos.
7) Direccionamiento. Es decir, detectar dónde está el receptor y el transmisor.
8) Encaminamiento, esto es, definir hacia dónde se dirigirán los datos.
9) Recuperación de datos ante errores en la transmisión.
10) Formato de los mensajes.
11) Seguridad en el proceso de transmisión.
12) Gestión de la red.

2.3.1 Transmisión de datos

Todos los formatos de información considerados: voz, datos, imágenes, video, etcétera; se pueden representar mediante señales electromecánicas. Dependiendo del medio de transmisión y del entorno donde se realicen las comunicaciones, se pueden utilizar señales analógicas o digitales para realizar el transporte de datos. Toda señal está constituida por una serie de frecuencias constituyentes. Un parámetro clave en la caracterización de la señal es el ancho de banda, el cual definimos como el rango de frecuencias contenidas en una determinada señal. En términos generales, cuanto más

grande sea el ancho de banda mayor será la velocidad (capacidad) para transportar información [9]. El tipo de señales analógicas o digitales se pueden representar como:

- ✓ Uno de los problemas principales en el diseño de sistemas de comunicaciones reside en las dificultades para transmitir por líneas de comunicación; o en los defectos en estas mismas. Entre los obstáculos más importantes están la atenuación (debilitación de la señal), la distorsión de retardo y los distintos tipos de ruido que existen, en este caso existen diferentes tipos de ruido, entre ellos, el térmico, de intermodulación, diafonía e impulso. Más adelante detallaremos algunos de estos problemas. Las dificultades en la transmisión de señales analógicas causan efectos aleatorios que degradan la calidad de la información transmitida y puede afectar su inteligibilidad; cuando se utilizan señales digitales, los defectos en el envío de datos pueden introducir bits erróneos en la recepción.
- ✓ El diseñador de un sistema de comunicaciones debe tomar en cuenta cuatro factores determinantes el ancho de banda de la señal, la velocidad a la que se transmiten los datos, la cantidad de ruido presente en el proceso de envío de datos y finalmente la porción o tasa de errores que se pueden tolerar. El ancho de banda disponible está limitado por el medio a través del cual se transmite, así como por la necesidad de evitar interferencias con señales cercanas. Debido a que el ancho de banda es un recurso escaso, es conveniente maximizar la velocidad de transmisión para el ancho de banda del cual se dispone. Por su parte la velocidad de transmisión está limitada por el ancho de banda, la presencia de defectos en las líneas de transmisión como ruido y otros, por la tasa de errores que se tolera. El éxito en la transmisión de datos depende fundamentalmente de dos factores: la calidad de la señal transmitida y las características del medio de transmisión [8]. En la figura 2.10 se muestra tipos de señales para transmisión de datos.

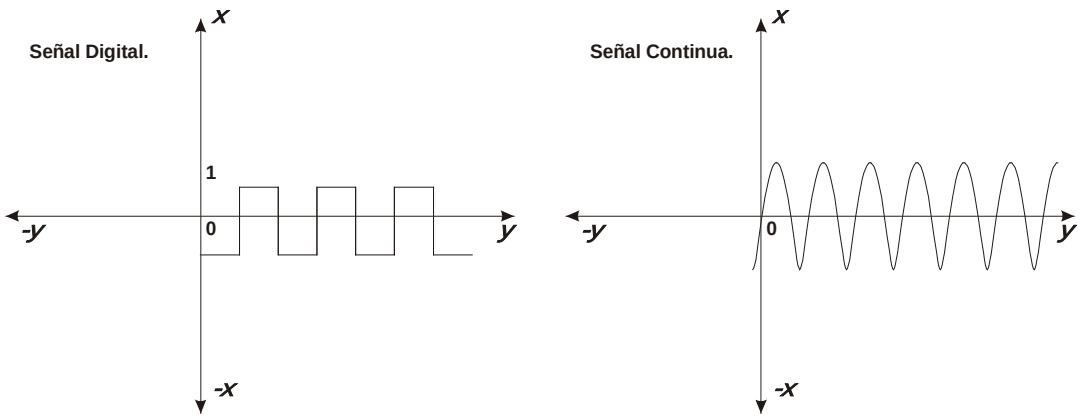


Figura 2.10. Tipos de señales para transmisión de datos.
Fuente [8].

2.4 Algoritmos genéticos

2.4.1 Introducción

Algoritmos genéticos (AGs) fue creado por John Holland (1975) [12]. Es una técnica de búsqueda aleatoria dirigida, que puede encontrar la solución óptima global en los espacios de búsqueda multidimensionales complejos. Un algoritmo genético es un modelo de evolución natural que los operadores emplean y está inspirado el proceso de evolución natural. Estos operadores, conocidos como los operadores genéticos, manipulan a los individuos en una población a lo largo de varias generaciones para mejorar su aptitud gradualmente.

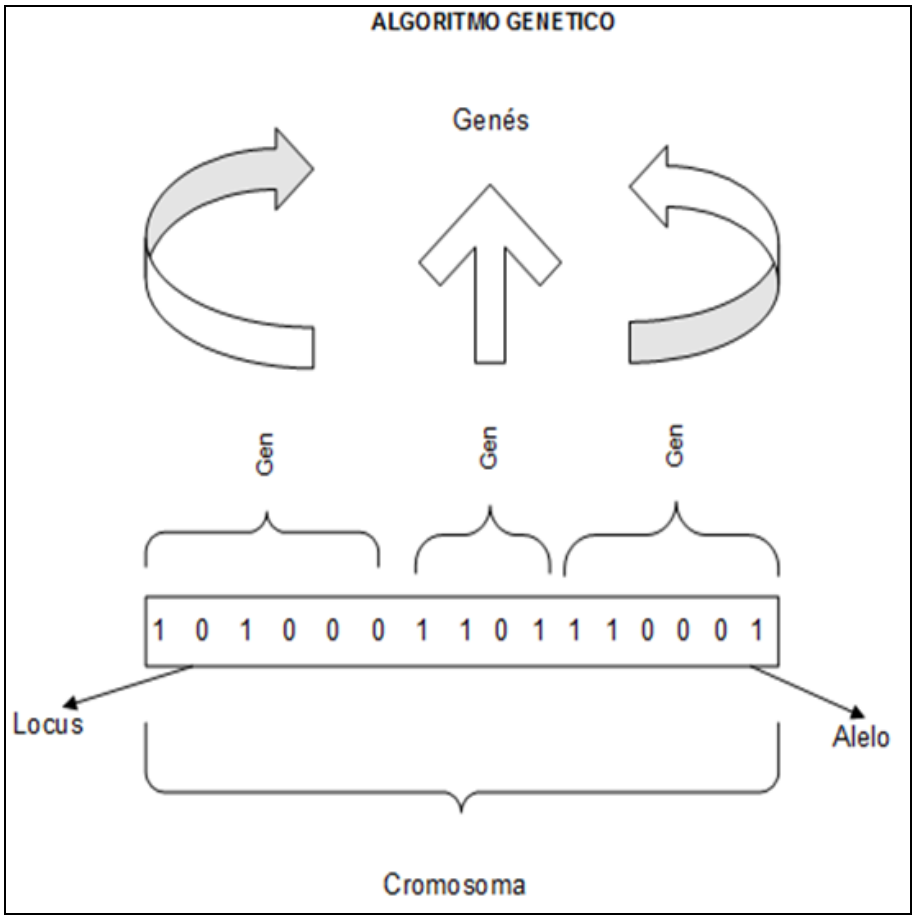


Figura 2.11. El cromosoma.
Fuente [2].

La evolución de una población de individuos se gobierna por "teorema del esquema" de la población, la cual se refiere a la similitud de los bits con ciertas posiciones de esos individuos. Por ejemplo, el esquema 1*0* que describe un conjunto de individuos cuyo primer y tercer bits son 1 y 0, respectivamente. Aquí, representa * los medios y cualquier valor binarios aceptable. En otros términos, los valores de bits en las posiciones marcadas * podría ser 0 o 1 en una cadena binaria. Un esquema se caracteriza por dos

parámetros: la longitud definida y el orden. La longitud definida es la longitud entre los primeros y últimos momentos con los valores fijos. El orden de un esquema es el número de bits con los valores especificados. Según el teorema del esquema [1], la distribución de un esquema a través de la población de una generación a la próxima depende de su orden, mientras se ha definido longitud y aptitud.

Los algoritmos genéticos no necesitan mucho conocimiento sobre el problema que van a perfeccionar y no trata directamente con los parámetros del problema. Trabajan con códigos que representan los parámetros.

Así, el primer problema en una aplicación de algoritmos genéticos es cómo codificar el problema bajo estudio, es decir, cómo representar los parámetros del problema.

Los algoritmos genéticos operan con una población de posibles soluciones, no sólo una posible solución, y el segundo problema es, por consiguiente, cómo crear la población inicial de posibles soluciones.

El tercer problema en una aplicación de algoritmos genéticos es cómo seleccionar o crear un conjunto adecuado de operadores genéticos.

Finalmente, como ocurre con otros algoritmos de búsqueda, los algoritmos genéticos tienen que saber la calidad de soluciones encontradas para mejorarlas más adelante. Hay una necesidad, por consiguiente, de una interfaz entre el ambiente del problema y los algoritmos genéticos propios [12].

2.4.2 Representación

Normalmente se representa la optimización de parámetros que van a ser perfeccionados en un formulario de una cadena de operadores genéticos que son adecuados para este tipo de representación. El método de representación tiene un gran impacto en la actuación de los algoritmos genéticos. Los esquemas de la representación de diferentes binarios podrían causar actuaciones diferentes lo que se refiere a la exactitud y tiempo del cómputo.

Hay dos métodos de representación común para los problemas de optimizaciones numéricos [6].

El método preferido es el método de representación de cadena binaria. La razón para este método que es popular es que el alfabeto binario ofrece el número máximo esquemático por binarios en comparación con otras técnicas codificadas. Varios binarios que sí codifican los esquemas pueden encontrarse en el texto; por ejemplo, el código uniforme y el código de escala de Gray.

El segundo método de la representación es utilizado para un vector de enteros o números reales, con cada entero o número real representando sólo el parámetro. Cuando

un esquema de representación binaria es empleado, un problema importante es decidir el número de bits utilizados en el código de parámetros que se va a optimizar. Cada parámetro debería codificarse con el número óptimo de bits que cubren todas las posibles soluciones, cuando se utilizan pocos o demasiados bits se podrían afectar el rendimiento del sistema.

2.4.3 Creación de la población inicial

Los individuos de la población inicial de un algoritmo genético suelen ser cadenas de ceros y unos generadas de forma completamente aleatoria, es decir, se va generando cada gen con una función que devuelve un cero o un uno con igual probabilidad [1].

Al inicio de la optimización, un algoritmo genético requiere un grupo de soluciones iníciales. Hay dos maneras de formar esta población inicial. **La primera** consiste en utilizar las soluciones aleatorias producidas al crear un generador del número aleatorio. Este método es preferido para los **problemas que no tiene conocimiento a priori que existe** o por evaluar la actuación de un algoritmo [12].

El segundo método **emplea el conocimiento a priori** para el problema de optimización dado. Utilizando este conocimiento, se obtiene un conjunto de requerimientos y se coleccionan soluciones que satisfacen esos requerimientos para formar una población inicial. En este caso, los algoritmos genéticos empiezan la optimización con un conjunto de soluciones aproximadamente conocidas y, por consiguiente, converge a una solución óptima en menos tiempo que con el método anterior.

2.4.4 Operadores genéticos

El diagrama de flujo de algoritmos genéticos simple se muestra en la figura 2.12. Hay tres operadores genéticos comunes de: selección, cruzamiento, y mutación. En la literatura pueden encontrarse muchas versiones de estos operadores. No es necesario emplear todos los operadores en un algoritmo porque cada una de las funciones es independiente de las otras. La opción o plan de operadores dependen del problema y la representación formal de los planes empleados. Por ejemplo, no puede utilizarse operadores diseñados para las cadenas binarias directamente en cadenas codificadas con enteros o números reales.

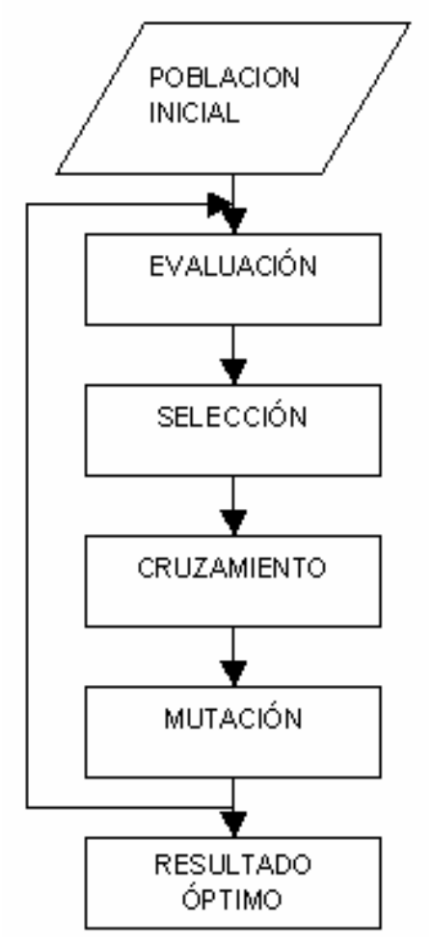


Figura 2.12. Diagrama de flujo de algoritmos genéticos simple.
Fuente [11].

a) La Selección

El objetivo del procedimiento de la selección es seleccionar individuos cuyos valores de aptitud (fitnees: mejor aptitud) son más altos que aquellos cuyos valores de aptitud son bajos. El procedimiento de la selección tiene una influencia significante en manejar la búsqueda hacia un área prometedora y las soluciones buenas son encontradas en un tiempo corto. Sin embargo, se debe mantener la diversidad de la población y evitar la convergencia prematura para alcanzar la solución óptima global. En los algoritmos genéticos hay dos procedimientos de selección principal: selección proporcional y selección de clasificación jerárquica, basada en [20]:

La selección proporcional normalmente se llama la "rueda de la ruleta", la selección que se asemeja al funcionamiento de una rueda de la ruleta. Es valorada la aptitud de individuos representantes de las anchuras de hendiduras en la rueda. Para seleccionar a un individuo para la próxima generación, los individuos en las hendiduras con anchuras

grandes que representan los valores de aptitud altos tendrán una oportunidad más alta de ser seleccionados después de un conjunto aleatorio de la rueda.

Una manera de prevenir la convergencia rápida es limitar el rango de ensayos asignados a un solo individuo, para que ningún individuo genere demasiados descendientes. La clasificación jerárquica tiene como base el procedimiento de la producción de idea planteada. Según este procedimiento, cada individuo genera un número esperado de descendientes que se basa en la línea de su valor de función de evaluación.

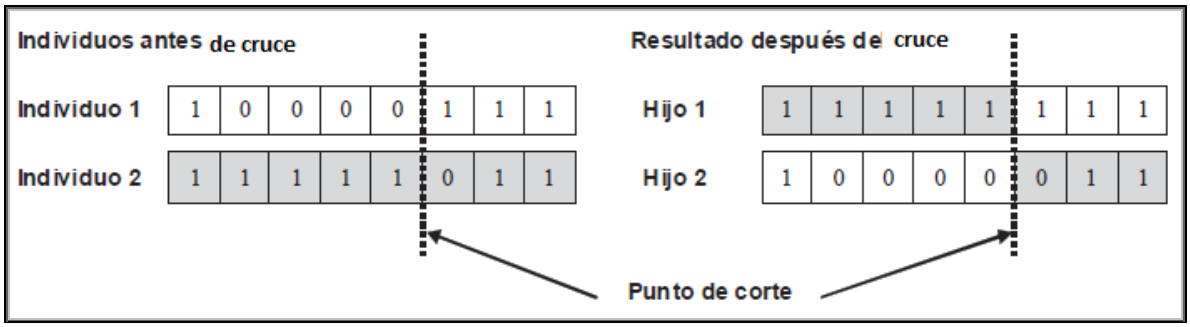
b) Cruzamiento

Se considera que esta operación es lo que hace a los algoritmos genéticos diferentes de otros algoritmos, como la programación dinámica. Se utiliza para crear dos nuevos individuos (los niños) de dos individuos existentes (los padres), que son escogidos de la población actual para la operación de selección. Hay varias maneras de hacer esto. Algunas operaciones de cruzamientos comunes son: un cruzamiento de punto, dos cruzamientos de puntos, cruzamiento de ciclo y cruzamiento uniforme.

El cruzamiento de un punto. Es la operación de cruzamiento más simple. Se selecciona dos individuos al azar como los padres de una familia de individuos formados por el procedimiento de la selección, y se cortará al azar un punto escogido. Las colas son las partes después del punto cortante, y dos nuevos individuos (los niños) se producen.

Veamos que esta operación no cambia los valores de los bits. Se muestran en las tablas de: 2.1, 2.2, 2.3 y 2.4 operaciones de cruzamientos diferentes:

Tabla 2.1. Un cruzamiento al punto.
Fuente [2]



De la tabla 2.1 de un cruzamiento al punto, el individuo 1 tiene ocho bits de ceros y unos, como la fórmula L-1 = L puntos de cruce, y aplicando la tabla 2.1 se tiene 8-1 = 7 puntos; entonces, se debe generar número aleatorio de 1 al 6. Está vez se ha tomado el número 5, es decir, del individuo 1 se toma los 5 bits primeros de 10000 y del individuo 2 se toma los últimos 3 bits de 011 y se une ambas cadenas; entonces se tiene la cadena

del hijo 1 de **11111111** de punto de cruzamiento. Del individuo 2 se toma los 5 primeros bits 11111 y del individuo 1 se toma los 3 últimas bits de 111 y se une ambas cadenas; entonces se tiene la cadena del hijo 2 10000011 de punto de cruzamiento. De esta manera se hace el cruzamiento de puntos de la población de algoritmos genéticos.

Tabla 2.2. Dos cruzamientos al punto.
Fuente [2].

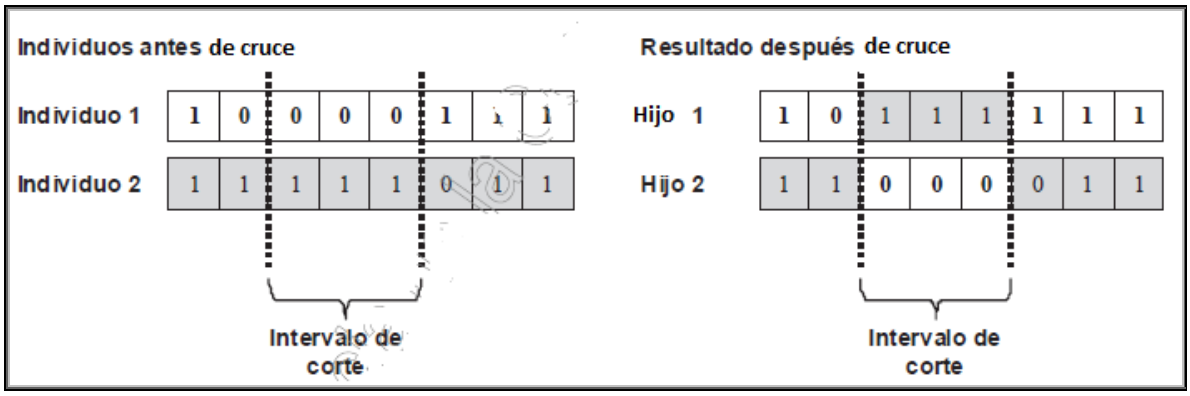
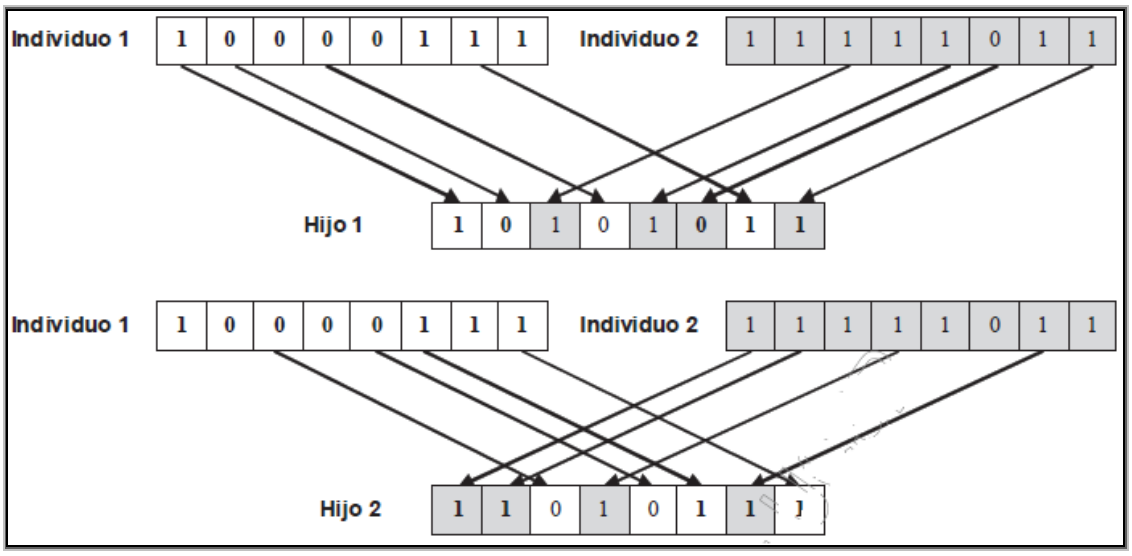


Tabla 2.3. El cruzamiento del ciclo.
Fuente [12].

Padre 1	12345678
Padre 2	abcdefgh

La cadena nueva 1	1b3de6g8
La cadena nueva 2	a2c45f7h

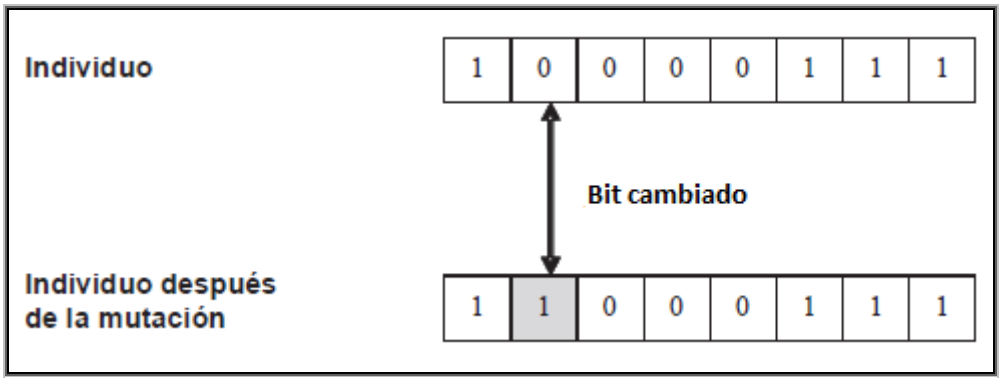
2.4 Tabla. El cruzamiento uniforme.
Fuente [2].



c) La Mutación

En este procedimiento, se verifican todos los individuos en la población, bit por bit, y los valores de bits se invierten al azar según una proporción especificada. El cruzamiento diferente es una operación monádica, es decir, una cadena de niño se produce de una sola cadena de padre. El operador de la mutación obliga al algoritmo a investigar las nuevas áreas. En el futuro, ayuda a los algoritmos genéticos a evitar la convergencia prematura y encuentra la solución óptima global. Un ejemplo se muestra en la tabla 2.5.

Tabla 2.5. Mutación de cromosomas.
Fuente [2].



Se puede tener uno o más operadores de mutación para nuestra representación; introducir variabilidad genética en la población a través de la mutación. Se muestra en la figura 2.12 la mutación del individuo.

Algunos aspectos importantes a tener en cuenta, son:

- Debe permitir alcanzar cualquier parte del espacio de búsqueda.
- El tamaño de la mutación debe ser controlado.
- Debe producir cromosomas válidos.

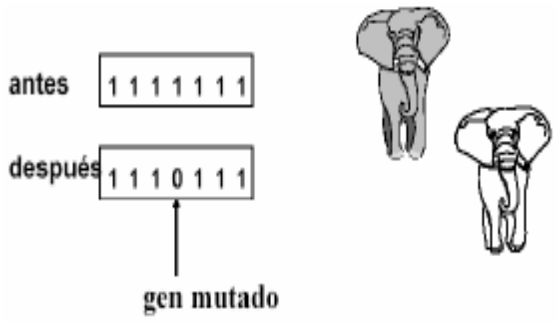


Figura 2.13. Mutación del individuo.
Fuente [12].

La inversión. Este operador adicional es empleado para varios problemas descritos en este tema, incluyendo el problema de la colocación celular, problemas del diseño y el

problema del viajante de comercio. También opera a un individuo en un momento dado. Se selecciona dos puntos al azar de un individuo y la parte de la cadena entre esos dos puntos se invierte, un ejemplo se muestra en la tabla 2.6 donde la inversión entre los bits 3 al 6.

Tabla 2.6. Inversión de un segmento de la cadena binaria.
Fuente [12].

La cadena vieja	1011001110
La cadena nueva	1000111110

2.4.5 Parámetros de control

Los parámetros de control importantes de los algoritmos genéticos simples incluyen el tamaño de la población y el número de individuos en la población; los cruzamientos tasan y proporcionan la mutación. Varios investigadores han estudiado el efecto de estos parámetros de rendimiento de algoritmos genéticos [17]. Las conclusiones principales han sido consideradas los medios de tamaño de población grandes para el manejo simultáneo de varias soluciones e incremento del tiempo de cómputo por la iteración; sin embargo, desde muchas muestras del espacio de búsqueda se ha visto que la probabilidad de convergencia a una solución óptima global es más alta que al usar un tamaño pequeño de la población. La proporción de cruzamiento determina la frecuencia de operación de cruzamiento. Es útil en la salida de optimización al descubrir una región prometedora. Una frecuencia baja de cruzamiento disminuye la velocidad de convergencia a un área. Si la frecuencia es demasiado alta, lleva a la saturación alrededor de una solución. La operación de mutación se controla por la proporción de la mutación. Una proporción de la mutación es alta si introduce la alta diversidad en la población y podría causar inestabilidad. Por otro lado, normalmente es muy difícil para los algoritmos genéticos encontrar una solución óptima global con una proporción demasiado baja de mutación.

2.4.6 Función de evaluación de aptitud

La unidad de evaluación de aptitud actúa como una interfaz entre algoritmos genéticos y el problema de optimización. Los algoritmos genéticos evalúan las soluciones para su calidad, según la información producida por esta unidad y no utilizando la información directa sobre la estructura. Se diseña el plan de problemas, los requisitos funcionales están especificados por el diseñador que tiene que producir una estructura de rendimiento que realiza las funciones deseadas dentro de las restricciones

predeterminadas. La calidad de una solución propuesta es usualmente calculada dependiendo del resultado de la solución que realiza las funciones deseadas y satisface las restricciones dadas. En el caso de algoritmos genéticos, este cálculo debe ser automático y el problema es cómo inventar un procedimiento que calcule la calidad de las soluciones.

Funciones de evaluación de la aptitud pueden ser simples o complejas, dependiendo del problema de optimización a la mano. Cuando una ecuación matemática no puede formularse para esta tarea, un procedimiento basado en reglas puede ser construido para uso como una función de aptitud o en algunos casos ambos se pueden combinar. Donde algunas limitaciones son muy importantes y no pueden ser violados, las estructuras o soluciones que lo hacen se pueden eliminar con antelación mediante el diseño de forma adecuada el esquema de representación. Alternativamente, se pueden dar bajas probabilidades utilizando funciones especiales de penalización.

2.4.7 Arquitectura de funcionamiento de algoritmos genéticos

La muestra es, en esencia, un subgrupo de la población. Es decir, es un subconjunto de elementos que pertenecen a ese conjunto definido en sus características a los que se denomina población.

Se muestra en la tabla 2.7, ahora se aplica sistema binario de $x = 2^y, y = 0,1,2,...,n$ y $f(x) = x^2$. Por ejemplo, x_1 = 16x0 + 8x1 + 4x1 + 2x0 + 1x0 = 12, esta muestra es un individuo / cromosoma, conjunto de individuos es la población total o universo.

Sea una función f(x) muy sencilla de:

$$f(x) = x^2$$

Sí se desea encontrar el valor de x, que hace que la **función f(x) alcance su valor máximo** (tabla 2.7), sin embargo, restringiendo a la variable x a tomar valores comprendidos entre 0 y 31. Aún más, a x sólo le vamos a permitir tomar valores enteros, es decir: 0, 1, 2, 3,..., 30, 31. Obviamente el máximo se tiene para x = 31, donde f vale 961. No se necesita saber algoritmos genéticos para resolver este problema, sin embargo, su sencillez hace que el algoritmo sea más fácil de entender [16].

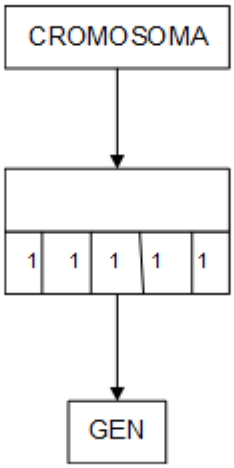
Lo primero que se debe hacer es encontrar una forma de codificar las posibles soluciones (posible valores de x). Una manera de hacerlo es con la codificación binaria. Con esta codificación un posible valor de x es

y = (0, 1, 1, 0, 0)

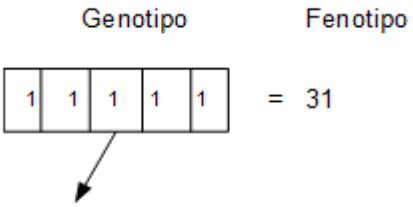
Muy sencillo: ahora se aplica sistema binario de $x = 2^y, y = 0,1,2,...,n$ y f(x) = x^2.

Por ejemplo, sea **y = 16x0 + 8x1 + 4x1 + 2x0 + 1x0 = 12.**

Observa que (0, 0, 0, 0, 0) equivale a x = 0 y que (1, 1, 1, 1, 1) equivale a x = 31.
Por ejemplo, se muestra en la figura 2.14 operación de cromosoma siguiente:



El fenotipo podría ser un entero.



$$1*2^4+1*2^3+1*2^2+1*2^1+1*2^0=16+8+4+2+1=31$$

Figura 2.14. Operación de cromosoma.
Fuente [16].

A cada posible valor de la variable x en representación binaria le vamos a llamar individuo. Una colección de individuos constituye la población y el número de individuos que la componen es el tamaño de la población.

Una vez que tenemos codificada la solución, debemos escoger un tamaño de población. Para este ejemplo ilustrativo escogemos 6 individuos.

Debemos partir de una población inicial. Una manera de generarla es aleatoriamente: coge una moneda y lánzala al aire; si sale cara, la primera componente del primer individuo es 0 y en caso contrario 1 cruz. Repite el lanzamiento de la moneda y tendremos la segunda componente del primer individuo (0 si sale cara y 1 si sale cruz).

Así hasta 5 veces y obtendrás el primer individuo. Repite ahora la secuencia anterior para generar los individuos de la población restantes. En total se tiene que lanzar 5x6 = 30 veces la moneda.

Nuestro siguiente paso es hacer competir a los individuos entre sí. Este proceso se conoce como selección. La tabla 2.7 resume el proceso.

Tabla 2.7. Selección de cromosomas.
Fuente [15].

(1)	(2)	(3)	(4)	(5)
1	(0, 1, 1, 0, 0)	12	144	6
2	(1, 0, 0, 1, 0)	18	324	3
3	(0, 1, 1, 1, 1)	15	225	2
4	(1, 1, 0, 0, 0)	24	576	5
5	(1, 1, 0, 1, 0)	26	676	4
6	(0, 0, 0, 0, 1)	1	1	1

Cada fila en la tabla 2.7 está asociada a un individuo de la población inicial. El significado de cada columna de la tabla es el siguiente:

(1) = Número que les asignamos al individuo.

(2) = Individuo en codificación binaria.

(3) = Valor de x.

(4) = Valor de f(x).

En cuanto a la columna (5) es la comparación de pareja a cada individuo. Una manera de realizar el proceso de selección es mediante un torneo entre dos. A cada individuo de la población se le asigna una pareja y entre ellos se establece un torneo: el mejor genera dos copias y el peor se desecha. La columna (5) indica la comparación de pareja a cada individuo, lo cual se ha realizado aleatoriamente. Existen muchas variantes de este proceso de selección, aunque este método nos vale para ilustrar el ejemplo.

Observa que el mejor individuo es fila 5 con f = 676. Se calcula la media de f y se obtiene fmed =324.3.

Después de realizar el proceso de selección, la población que tenemos es la mostrada en la columna (2) de la tabla 2.8. Se observa, por ejemplo, que en el torneo entre el individuo

1 y el 6 de la población inicial, el primero de ellos ha recibido dos copias, mientras que el segundo es desechado.

Tabla 2.8. Cruce de cromosomas.
Fuente [16].

(1)	(2)	(3)	(4)
1	(0, 1, 1, 0, 0)	12	144
2	(0, 1, 1, 0, 0)	12	144
3	(1, 0, 0, 1, 0)	18	324
4	(1, 0, 0, 1, 0)	18	324
5	(1, 1, 0, 1, 0)	26	676
6	(1, 1, 0, 1, 0)	26	676

Tras realizar la selección, se realiza el cruce. Una manera de hacerlo es mediante el cruce 1X: se forman parejas entre los individuos aleatoriamente de forma similar a la selección. Dados dos individuos pareja se establece un punto de cruce aleatorio, que no es más que un número aleatorio entre 1 y 4 (la longitud del individuo menos 1). Por ejemplo, en la pareja 3-4 el punto de cruce es 3, lo que significa que un hijo de la pareja conserva los tres primeros bits del padre y hereda los dos últimos de la madre, mientras que el otro hijo de la pareja conserva los tres primeros bits de la madre y hereda los dos últimos del padre. La población resultante se muestra en la columna (2) de la tabla 2.9 [16].

Tabla 2.9. Población tras cruce.
Fuente [16].

(1)	(2)	(3)	(4)
1	(0, 1, 0, 1, 0)	10	100
2	(1, 1, 1, 0, 0)	28	784
3	(0, 1, 1, 1, 0)	14	196
4	(1, 0, 0, 0, 0)	16	256
5	(1, 1, 0, 1, 0)	26	676
6	(1, 0, 0, 1, 0)	18	324

En la columna (3) se tiene el valor de x; en la siguiente se tiene el valor de f(x) correspondiente.

Veamos ahora el valor máximo de f(x) es 784 (para el individuo 2), mientras que antes de la selección y el cruce era de 676. Además, fmed ha subido de 324.3 a 389.3.

¿Qué quiere decir esto?

Simplemente que los individuos después de la selección y el cruce son mejores adaptados que antes de estas transformaciones.

El siguiente paso es volver a realizar la selección y el cruce tomando como población inicial la de la tabla 2.9. Esta manera de proceder se repite tantas veces como número de iteraciones que se fijen. En realidad los algoritmos genéticos no les garantizan en los primeros pasos de obtención del óptimo, sin embargo, si está bien construido el sistema de aplicación y los procesos, les proporcionará una solución razonable óptima. Se puede obtener el óptimo, sin embargo, el algoritmo no le confirma que lo sea, es decir, uno debe quedarse con última iteración. Así que quédate con la mejor solución de la última iteración. También es buena idea ir guardando la mejor solución de todas las iteraciones anteriores y al final se quedará con la mejor solución de las exploradas.

Solucionar el problema consiste en encontrar la solución óptima, y por tanto, los algoritmos genéticos son en realidad un método de búsqueda. Sin embargo, un método de búsqueda muy especial, en el que las soluciones al problema son capaces de reproducirse entre sí, combinando sus características y generando nuevas soluciones.

a) Consideraciones para algoritmos genéticos

En problemas reales en los que se aplican los algoritmos genéticos, existe la tendencia a la homogenización de la población, es decir, a que todos los individuos de la misma sean idénticos. Esto impide que el algoritmo siga explorando nuevas soluciones, con la cual se puede quedar estancado en un mínimo local no muy bueno.

Existen técnicas para contrarrestar esta "deriva genética". El mecanismo más elemental, aunque no siempre suficientemente eficaz, es introducir una mutación tras la selección y el cruce. Una vez que ha realizado la selección y el cruce escoge un número determinado de bits de la población y los alteras aleatoriamente. En nuestro ejemplo consiste simplemente en cambiar alguno(s) bit(s) de 1 a 0 o de 0 a 1.

En conclusión, este capítulo trata de conceptos básicos que comprende un conjunto de conocimientos relacionados con algoritmos genéticos; esta teoría que permitirá conocer la estructura, comportamiento de individuos y la población en la aplicación proceso de tarea en la red, para optimización de procesos de tareas en servidor web.

Tiene la capacidad de resolver problemas con un grado de dificultad muy elevado con eficiencia y exactitud. Reducen el costo computacional, es decir, el tiempo de cálculo y el consumo de recursos es menor.

Una vez que se llega a comprender los algoritmos genéticos, la implementación no es complicada y pueden ser aplicables en varias áreas.

Se puede ver la gran ventaja que da al trabajar con algoritmos genéticos, por su sencillez de operadores genéticos.

2.5 Algoritmos genéticos asíncronos combinados para una red heterogénea de computadoras

2.5.1 Introducción

Algoritmos genéticos (AGs) han demostrado ser una muy buena herramienta en la búsqueda de óptimos globales, debido a su relativa sencillez en implementación y a la robustez de su aplicación [10]; sin embargo, los algoritmos genéticos necesitan generalmente de mucho tiempo de procesamiento para llegar a buenos resultados. Una solución a este problema es la paralelización dividiendo el problema global en subproblemas que son asignados a varios procesadores de un sistema computacional. De esta forma, estos procesadores realizarán la computación del problema asignado obteniendo resultados parciales que serán transmitidos a los otros procesadores, colaborando todos para llegar a la solución global del problema [4].

Desde que se presentaron las primeras ideas acerca de la paralelización de algoritmos genéticos hasta nuestros días; se han realizado numerosas implementaciones en este campo, destacándose las implementaciones asíncronas donde cada subpoblación es procesada en paralelo por otro procesador de un sistema computacional que intercambia migrantes con sus procesadores vecinos, siguiendo una determinada política migratoria, sin interrumpir el procesamiento en espera de comunicación con algún otro procesador. Este tipo de implementación tiene una doble ventaja para sistemas computacionales heterogéneos:

a) Elimina los tiempos muertos producidos por barreras de sincronización, que pueden ser extremadamente perjudiciales cuando se trabaja con una red de computadoras cuyo tráfico no se puede controlar totalmente y con máquinas distintas cuyo balanceamiento de carga es sumamente difícil de optimizar [3].
b) El procesamiento independiente de las subpoblaciones de cada computadora ha demostrado tener importantes ventajas sobre procesos, entre procesos simples paralelos del procesamiento de población en el servidor, dando los reportes rápidos de los procesos [15].

No es de extrañar entonces la gran cantidad de trabajos dedicados a la implementación de algoritmos genéticos paralelos en un contexto asíncrono.

El presente trabajo se ha organizado de acuerdo a la sección 2.5.2 que describe el algoritmo implementado. Asimismo se presenta la plataforma computacional utilizada y algunos resultados significativos utilizando funciones de prueba [5].

2.5.2 Algoritmos genéticos asíncronos combinados

En esta sección, se postula la utilización de diversos procesadores, posiblemente heterogéneos, procesando subpoblaciones de tamaños proporcionales al desempeño relativo de los procesadores (y por consiguiente de tamaños diferentes). Cada procesador aplicará a su población los operadores genéticos tradicionales de selección, cruzamiento y mutación [8], así como un operador de optimización local similar al propuesto [15].

Los individuos de cada subpoblación podrán migrar de un procesador a otro conforme a una política migratoria definida por los siguientes parámetros:

El intervalo de migración. Que establece en cada generación realizará la migración de una cierta cantidad de individuos desde una subpoblación a la otra.

La tasa de migración. Que indica cuantos individuos han de comunicarse a la otra subpoblación cuando se cumple el intervalo de migración.

El criterio de selección de los migrantes. Que determina la política que se aplicará para selección de individuos que han de migrar. Por ejemplo, se puede establecer que los migrantes sean elegidos al azar. Sin embargo, es posible ayudar a las otras subpoblaciones seleccionando los individuos mejor adaptados, es decir, de **mejor aptitud** (fitness). Esta última opción es la más natural desde el punto de vista biológico, pues en la naturaleza, las migraciones involucran grandes esfuerzos a los migrantes, por la cual serán los mejores los que terminan la travesía.

La topología de comunicación. Que define el sentido de las migraciones, es decir, él o los procesadores de destino de la migración. Generalmente, la topología de comunicación mapea directamente a la arquitectura computacional utilizada, sea esta un hipercubo, toroide, anillo o malla totalmente interligada, etcétera.

Una variante de la presente propuesta respecto de otras anteriores, es la utilización de optimizadores locales diferentes según sean las características del procesador que lo ejecuta. **Para resultados experimentales que utilizaron un algoritmo numérico de optimización local, disponible en una biblioteca, se tiene los siguientes algoritmos**:

m_1: método del gradiente.

m_2: algoritmo del ascenso suficiente.

m_3: algoritmo modificado de Hookes y Jeeves.

m_4: algoritmo de métrica variable.

De esta forma, un operador de optimización local trabaja de la siguiente forma:

1. Escoge un individuo en forma aleatoria o aplicando el operador de selección, haciendo que el individuo más adaptado tenga mayor probabilidad de ser elegido.
2. Aplica el algoritmo numérico asignado a ese procesador, al individuo escogido, optimizándolo hasta un posible máximo (generalmente local).
3. Se reemplaza al individuo escogido por su versión optimizada.

La versión secuencial del pseudocódigo implementado utilizando el operador de optimización local es presentada a continuación. Puede notarse que el mismo solo se aplica después de la iteración numMax para evitar la optimización local de individuos que aún no tienen información respecto a posibles óptimos globales [15]. A continuación se muestra el siguiente pseudocódigo 1 [4]:

Pseudocódigo 1. Algoritmos genéticos combinados

```
Inicializa_la_Población;
Estadística_de_la_Población;
t ← 0;
Criterio_de_Parada=FALSE;
DO WHILE NOT (Criterio_de_Parada)
t ← t + 1;
Reproducción (Selección, Cruzamiento y Mutación);
IF (t ≥ numMax) THEN Optimizador_Local;
Estadística_de_la_Población;
IF (t>numMax) THEN Criterio_de_parada=TRUE;
END DO
```

La utilización del optimizador local puede ser perjudicial cuando se aplica secuencialmente, pues puede disminuir rápidamente la variedad genética, acelerando la convergencia del algoritmo a valores subóptimos. Sin embargo, cuando es aplicado a subpoblaciones en un contexto paralelo, la existencia de migrantes minimiza el peligro de pérdida genética, sin perder la capacidad de acelerar convenientemente todo el proceso. En consecuencia, los optimizadores locales muestran todo su potencial en un contexto paralelo, que puede ser implementado utilizando un proceso maestro que se encarga de

administrar todo el sistema se muestra en el pseudocódigo 2 que incluye el lanzamiento de diversos procesos esclavos en cada uno de los procesadores disponibles en el sistema computacional, estos procesos que realizan los cálculos propiamente dichos, conforme se muestra en el siguiente pseudocódigo 3 [4]:

Pseudocódigo 2. Estructura del proceso maestro [2]

```
Leer_Datos;
Levantar_Procesos_Esclavos;
Enviar_Parámetros_a_cada_Esclavo;
Fin ← FALSE;
DO WHILE NOT (fin)
Hacer_Procesos_Esclavos;
Recibir_Mensaje_Terminación_de_Esclavos;
IF (Todas las máquinas terminaron) THEN fin ←TRUE;
END DO
Enviar_Mensaje_de_Fin_a_Esclavos;
Eliminar_Procesos_Esclavos;
```

Pseudocódigo 3. Estructura de cada proceso esclavo [4]

```
Recibir_Parámetros;
Iniciar_Población;
Estadística_de_la_Población;
Selección_de_Individuos;
t ← 0;
DO WHILE (TRUE)
t ← t + 1;
Reproducción (Selección, Cruzamiento y Mutación);
Escoger_Migrantes;
Enviar_Migrantes (a otros procesos esclavos);
Recibir_Migrantes (de otros procesos esclavos);
Seleccionar_Individuos (Manteniendo tamaño de la Población);
Estadística_ de_la_Población;
IF (t ≥ numMax) THEN Optimizador_Local;
IF (Criterio de fin) THEN
 {
```

```
   Mensajes_al_Master;
   Salir;
  }
END DO
```

Se observa en una implementación asíncrona que el número de migrantes recibidos pueden variar de una generación a otra, por la cual es necesario aplicar el operador de selección para mantener constante el tamaño de la subpoblación en cada procesador.

Conforme lo expuesto antes, la implementación del operador de optimización local puede variar de un procesador a otro, haciendo cálculos más precisos en las computadoras de mayor desempeño y hasta eliminándolo completamente de las computadoras más lentas, sirviendo este mecanismo como una forma de mejorar el balance de carga sin tener que utilizar subpoblaciones de tamaños muy dispares.

De esta forma, es posible aprovechar toda la capacidad computacional de las redes de computadoras existentes, aún en presencia de computadoras con substanciales diferencias de desempeño.

Cabe mencionar que en sistemas de comunicaciones con suficientes recursos de máquinas, es posible crear un proceso especializado para optimizar suficientemente a los buenos candidatos, sin implementar los algoritmos genéticos. Así, cuando un procesador encuentra un buen individuo, por ejemplo, al uniformizar su subpoblación en una generación dada, este puede ser enviado al referido proceso para su optimización posterior y eventual difusión a los demás procesadores del sistema.

Finalmente, se enfatiza el hecho de que cada procesador trabaja sin coordinación con los demás procesadores, posiblemente con tamaños diferentes de la subpoblación e implementaciones y criterios diferentes para los operadores genéticos, lo que contribuyen en diversidad genética y, por consiguiente, para la obtención de mejores soluciones que las obtenidas con implementaciones totalmente uniformes. Sin embargo, estas implementaciones asíncronas pierden el concepto de generación pues cada procesador puede realizar un número diferente de iteraciones que a su vez no están sincronizadas entre sí.

2.5.3 Resultados experimentales

Las funciones F2 y F5 que se usan en los algoritmos genéticos para determinar los valores de la Tabla 2.11. En caso de x_2 representa puntos de espacio de búsqueda del problema.

Los resultados experimentales que se presentan a continuación se basan en la optimización de las siguientes funciones de [5]:

F2: $f(x_i) = 100 * (x_1^2 - x_2)^2 + (1 - x_1)^2, \quad \forall \quad -2.048 \leq x_i \leq 2.048$

F5: $f(x_i) = 0.002 + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=10}^{2} (x_i - a_{ij})^6}, \quad \forall \quad -65.536 \leq x_i \leq 65.536$

Para obtener estos datos experimentales se utilizaron las siguientes estaciones de trabajo interconectadas a una red tipo Ethernet (10Base T) con un centenar de computadoras personales:

1. Workstation DEC 3000 modelo 300, con procesador alpha de 150 MHz, 32 Mb en RAM y sistema operativo OSF/1 v. 2.0.
2. Workstation SUN SPARC Station 5, con procesador Sparc de 66 MHz, 32 MB en RAM y sistema operativo Solaris 5.3.

Las diversas implementaciones realizadas y descritas en la tabla 2.10 fueron codificadas en **máquina virtual paralelo** (MVP) en la versión extendida de ANSI C.

Estas implementaciones fueron luego combinadas con distintos métodos numéricos de optimización, descritos en la sección anterior, los que fueron utilizados como operadores de algoritmos genéticos.

Tabla 2.10. Algoritmos genéticos implementados.
Fuente [4].

IMPLEMENTACIÓN	COMUNICACIÓN	ALGORITMO	SÍMBOLO
Secuencial	-	-	SS
Secuencial	-	Combinado	SCm_i
Paralela	Síncrona	-	PS
Paralela	Síncrona	Combinado	PSm_im_k
Paralela	Asíncrona	-	PA
Paralela	Asíncrona	Combinado	PAm_im_k

Donde m_i representa el optimizador local utilizado en la workstation SUN y m_k utilizado en la DEC en caso de las implementaciones combinadas, conforme a la descripción de índices de la sección anterior.

En las mediciones de tiempo y desempeño, se consideró que la ejecución de cada una estaba terminada si se cumplía uno de los siguientes criterios de parada:

1. Varianza de la población menor que uno y dado.

2. Número máximo de iteraciones.

3. Tiempo máximo de ejecución.

El desempeño de un algoritmo se dio corriendo el mismo algoritmo N veces (típicamente, N = 50) y calculando el valor medio de los tiempos empleados y los valores obtenidos para las funciones objetivos que se estaban maximizando.

En la tabla 2.11, se muestra algunos de los resultados experimentales obtenidos con una población de 100 individuos de los cuales 40% son asignados a la SUN y el resto a la DEC 3000. Como puede observarse, las implementaciones paralelas asíncronas y combinadas (con optimizadores locales) son claramente superiores a las implementaciones paralelas más tradicionales, tanto en la calidad de la solución encontrada (desempeño) como en el tiempo de procesamiento de datos, por ejemplo, PAm1m3 con la función F2, PAm1m1 y PAm1m3 con la función F5 [6].

Tabla 2.11. Resultados experimentales.
Fuente [4].

Función	Implementación	Desempeño	Tiempo (seg.)
F2	SS	3838.72119	3.664911
	SCm1	3853.59815	2.980207
	PA	3887.17944	1.4161
	PAm1m1	3871.78833	3.370761
	PAm1m3	3888.93872	3.986055
	PS	3820.40942	3.145111
	PSm1m1	3777.49829	3.634333
	PSm1m4	3672.52002	3.509075
	PSm1m3	3786.3147	3.756697
F5	SS	3.790546	3.94999
	SCm1	3.817955	1.800819
	PA	3.815276	3.276627
	PAm1m1	3.817958	1.448785
	PAm1m4	3.817958	2.792711
	PAm1m3	3.817958	2.326697
	PS	3.791299	3.834485
	PSm1m1	3.817954	3.480729
	PSm1m4	3.817899	3.988115

En la tabla 2.12, se muestra el efecto de desbalancear las cargas, variando la cantidad de individuos a ser asignados a cada procesador. Como es de esperar, los tiempos de procesamiento dependen del balanceamiento de carga en general, la utilización de algoritmos combinados diferentes en cada procesador resulta casi siempre benéfica, sobre todo cuando las cargas están bien balanceadas. Por ejemplo, se puede observar

que los mejores desempeños del algoritmo se obtuvieron al combinar los algoritmos genéticos paralelos con el método del gradiente (m_1) en un procesador y el de métrica variable en el otro (m_4).

Tabla 2.12. Resultados para diferentes particiones.
Fuente [4].

Función de Jong	Desempeño óptimo	Desempeño experimental	Tiempo (segundos)	Subpoblación SUN	Método SUN	Método DEC
	3.817958	3.56356	3.517408	20	m1	m1
F5	3.817958	3.56381	2.068437	30	m1	m1
	3.817958	3.817958	1.448785	40	m1	m1
	3.817958	3.817958	2.236543	50	m1	m1
	3.817958	3.817958	1.268846	20	m4	m4
	3.817958	3.817958	3.086533	30	m4	m4
	3.817958	3.817958	2.55616	40	m4	m4
	3.817958	3.817958	2.981364	50	m4	m4
	3.817958	3.817958	1.574217	50	m1	m4
	3.817958	3.817958	2.054251	50	m4	m1
	3.817958	3.817958	1.040199	20	m1	m4
	3.817958	3.817958	1.085705	20	m4	m1
	3.817958	3.817958	2.96558	40	m1	m4
	3.817958	3.817958	3.391128	40	m4	m1

a) Análisis

Las implementaciones paralelas mejoran tiempos de computación de algoritmos genéticos en una relación que puede llegar a ser súperlineal, llegando inclusive a mejores soluciones que los algoritmos secuenciales al trabajar con subpoblaciones independientes [5]. Como una forma de aprovechar estas características en un sistema computacional heterogéneo y mejorar los tiempos de ejecución, se postuló combinar estos algoritmos genéticos paralelos con diversos métodos numéricos de optimización en un ambiente computacional asíncrono, lográndose mejores resultados sin una pérdida importante de la información genética y la aleatoriedad requerida.

Los resultados experimentales de la sección anterior demuestran:

1. Las implementaciones asíncronas son mejores y que las implementaciones tradicionales secuenciales y síncronas.
2. La utilidad de combinar otros métodos de optimización con algoritmos genéticos, y se puede implementar totalmente secuencial.

3. La ventaja de utilizar diferentes métodos en cada uno de los procesadores de un sistema computacional heterogéneo, aprovechando esta heterogeneidad para privilegiar la diversidad de la población y la aleatoriedad.

La posibilidad de mejorar los tiempos de ejecución y desempeño general del algoritmo combinado efectos de paralelizar los algoritmos genéticos, implementarlo en forma asíncrona y combinarlo con otros algoritmos, en una implementación totalmente heterogénea.

En conclusión, la reconocida potencialidad de los algoritmos genéticos paralelos puede ser hoy en día aprovechada no-solo en los centros que cuentan con importantes computadoras paralelas, sino por cualquier institución o empresa con acceso a una red de computadoras, por más heterogéneas que puedan ser sus máquinas, pues implementaciones como la descrita permite el uso eficiente de los recursos computacionales disponibles actual y en el futuro.

CAPÍTULO 3

METODOLOGÍA DE IMPLEMENTACIÓN DEL SIMULADOR PROTOTIPO DE SISTEMAS DE COMUNICACIÓN UTILIZANDO ALGORITMOS GENÉTICOS PARA OPTIMIZAR PROCESOS

3.1 Introducción

Se ha tomado como base el capítulo II, para implementar un simulador prototipo de sistemas de comunicación utilizando algoritmos genéticos para optimizar procesos.

Se aplica al procesador que puede ejecutar múltiples procesos. Cada nodo tiene una cola para agrupar los procesos en espera de ser atendidos.

Cuando el despachador asigna el proceso a un nodo, este es almacenado en la cola de procesos en espera de ser atendidos.

Modelo de distribución de procesos en múltiples nodos. Todos los nodos son idénticos y los procesos tienen una estructura de requerimientos dado por una cadena binaria de requerimiento por recursos de nodos.

Existe un despachador que se encarga de asignar los procesos a los nodos. El despachador evalúa si el nodo cuenta con una cola disponible para almacenar el proceso hasta que sea atendido.

El modelo de optimización se aplica al servidor, que puede ejecutar procesos de tareas. Cada nodo tiene una cola para agrupar los procesos en espera de ser atendidos. Cuando el despachador asigna el proceso a un nodo, se almacena en la cola de procesos del nodo en espera de ser atendidos.

El proceso tiene un cromosoma que está definido por una cadena binaria que en estado 1 implica que solicita recurso específico.

3.2 Supuestos y restricciones

a) Supuestos

- Los nodos son generados por el usuario.
- Los procesos son generados por el usuario.
- La cola de cada nodo es establecido por el usuario.
- La probabilidad de mutación es establecido por el usuario.
- La probabilidad de cruzamiento es establecido por el usuario.
- El número de iteraciones son establecidos por el usuario.

- El número de alelos son establecidos por el usuario.
- Existe un nodo despachador, el cual asigna los procesos de manera secuencial a cada nodo, conforme tiene cola disponible para almacenar los procesos en espera de ser atendidos.
- Los nodos que inician con disponibilidad de recursos, mantienen sus características a través del proceso de simulación. No se degradan los recursos de cada nodo.
- Un proceso es un conjunto de solicitudes de recursos. En otras palabras, un proceso se transforma en requerimientos que necesitan para ser atendidos.
- La generación de un proceso se basa en un valor aleatorio, que está conformado por un número entero cuyo rango máximo es la combinación máxima de la longitud del cromosoma.
- Los nodos son: recurso disponible si tienen cola disponible.
- El despachador determina en función del algoritmo genético, cuál de los nodos va atender un proceso.

b) Restricciones

- Existe un tiempo limitado por el simulador.
- Los valores generados por el nuevo requerimiento son aleatorios.
- Existe un valor limitado para las iteraciones en los algoritmos genéticos.

3.3 Características y requerimientos del sistema

a) Características son:

Lenguaje: C++.

Compilador: Borland Developer Studio 2006.

Librerias: VCL Borland Developer 2006.

Sistema operativo: Windows XP, WS 2000, Windows Server 2003, Windows 2008, Windows 7.

Nativo: WIN32.

b) Requerimientos para simulador prototipo, son:

Memoria: 256 Kb (o superior) RAM.

Disco: 2 Mb.

Seguridad: acceder como administrador o equivalente.

3.4 Análisis y diseño del sistema simulador

En esta sección, se hace un breve resumen de análisis y diseño del sistema simulador como:

3.4.1 Entidades

En esta sección, la presente Tesina no se usa el **mapeo relacional a objetos** (Object Relational Mapping) debido que los procesos de tareas demoran más tiempo y en la Tesina se usa las estructuras independientes que son más rápidos los procesos.

Se ha elegido las entidades que se muestra en la figura 3.1 y **ver en la página 51 sobre entidades**, para desarrollo del simulador siguiente:

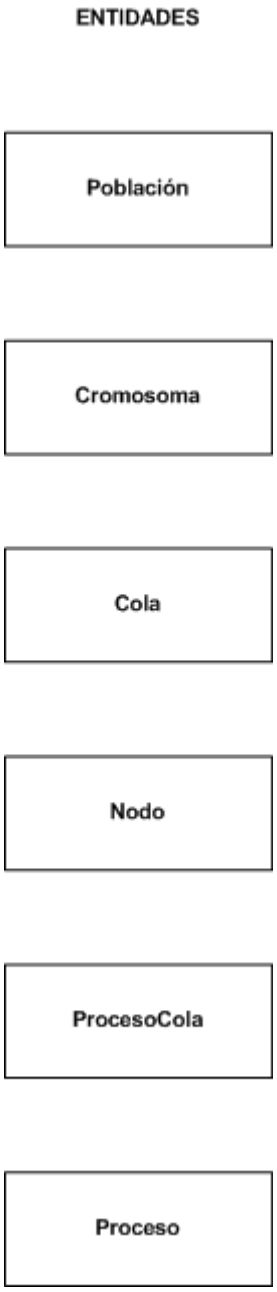


Figura 3.1. Entidades.
Fuente análisis y diseño.

3.4.2 Caso de uso

Los procesos a atender son asignados por usuarios. El sistema acumula una lista de procesos. En el simulador la carga de procesos son externos al sistema, el usuario determina la lista de procesos que el sistema deberá distribuir entre los nodos disponibles.

Los nodos son parte de la configuración del sistema, los cuales deben ser especificados por el usuario. Parte de la información para el funcionamiento del proceso de simulación involucra poder especificar la cantidad de procesos y los nodos que servirán para atender dichos procesos. El sistema deberá tomar la lista de procesos y nodos como parámetros de funcionamiento. En el proceso de ejecución, el sistema tomara la lista de procesos y con los algoritmos genéticos determinar que nodo atenderá dicho proceso. En nuestro sistema estamos haciendo suposiciones de la cantidad de recursos y tiempo requerido por el proceso; además, los nodos son idénticos. Definitivamente es necesario hacer suposiciones sobre estos puntos. En situaciones reales, los procesos necesitan otras variables que son necesarias para su tratamiento.

En los casos de uso, se determinan los siguientes procesos:

a. Asignar nodos. El usuario registra los nodos que conforman la lista de recursos para atender los procesos.
b. Registrar procesos. El usuario registra los procesos que serán atendidos por los nodos.
c. Explorar procesos. El sistema explora los procesos pendientes de atención.
d. Asignar procesos a cola de nodos. El sistema determina que nodo cuenta con cola disponible para aceptar el proceso.
e. Seleccionar óptimos. En base a los algoritmos genéticos se determina que el nodo que atenderá un proceso. Si bien un proceso está en cola, el algoritmo genético determina que nodo se activara para atender un proceso que se encuentra en la cola. Una vez que se ha atendido, la cola del nodo incrementa su disponibilidad.
f. Asignar proceso a nodo. El sistema asigna el proceso al nodo seleccionado previamente para que sea atendido, se muestra en la figura 3.2 casos de uso siguiente:

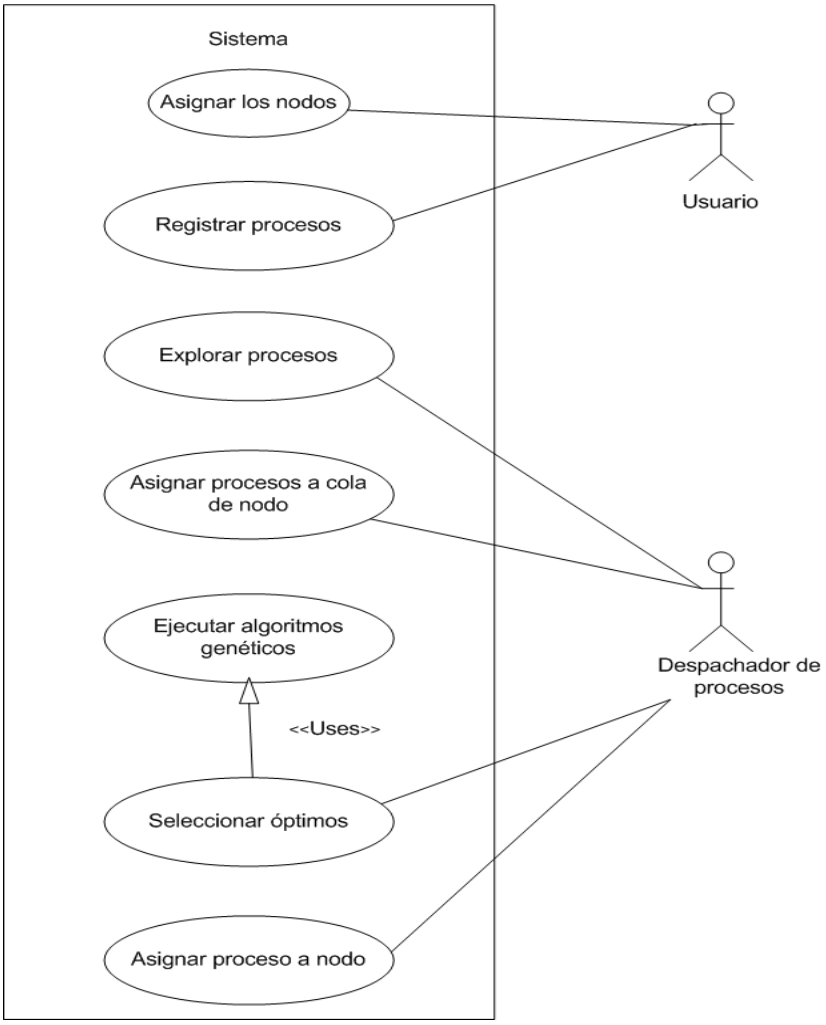


Figura 3.2. Casos de uso**.**
Fuente análisis y diseño.

Casos de uso:

a. Asignar nodos

Precondición. El usuario está conectado al sistema.

Flujo. El usuario registra los nodos que conforman la lista de recursos para atender los procesos. El usuario establece la cola de cada nodo. La cola de los nodos acumulan de manera temporal los procesos asignados al nodo.

Flujo alternativo. El usuario guarda la configuración del sistema en el disco.

Poscondición. Los nodos están configurados.

b. Registrar procesos

Precondición. El usuario está conectado al sistema.

Flujo. El usuario registra los procesos que serán atendidos por los nodos.

Flujo alternativo. El usuario guarda la configuración del sistema en el disco.

Poscondición. Los procesos están registrados.

c. Explorar procesos

Precondición. Los nodos están cargados, los procesos están definidos, la probabilidad de mutación, probabilidad de cruzamiento, el número de iteraciones y el número de alelos están definidos.

Flujo. El sistema explora los procesos pendientes de atención.

Flujo alternativo. No existen procesos pendientes, el sistema termina la simulación.

Poscondición. Determina un proceso entrante de la lista de procesos.

d. Asignar procesos a cola de nodos

Precondición. Existe proceso entrante, existe cola disponible en nodos.

Flujo. El sistema determina que nodo cuenta con cola disponible para aceptar el proceso.

Flujo alternativo. No existen nodos con cola disponible.

Poscondición. El proceso está asignado en la cola del nodo. Se reduce el total de procesos pendientes en el simulador.

e. Seleccionar óptimos

Precondición. La cola tiene procesos pendientes por asignar.

Flujo. En base a los algoritmos genéticos se determina que nodo atenderá un proceso que se encuentra en cola.

Poscondición. Se selecciona el nodo óptimo para que ejecute el proceso pendiente en cola.

f. Asignar proceso a nodo

Precondición. Se ha obtenido el proceso óptimo que será atendido por el nodo.

Flujo. El sistema asigna el proceso localizado en cola para que sea atendido por el nodo.

Poscondición. Se libera en un proceso la cola de procesos pendientes en el nodo (ver d.).

3.4.3 Diagrama de actividad del sistema

El proceso de simular asignación de procesos a nodos, está basado en la aplicación de algoritmos genéticos. Existe un proceso que se encarga de despachar los requerimientos a los nodos finales, a su vez, estos se encargarán de agruparlos en la cola y seleccionar el nodo óptimo que atenderá el proceso en cola.

Para el modelo, el sistema está en un proceso de asignación de requerimientos, estos son un conjunto de procesos en espera de ser asignados a los nodos. Para simular el cromosoma del requerimiento, el sistema genera un número entero en el rango de la longitud del cromosoma y como población toma la cantidad de procesos por atender en la cola de cada nodo. El proceso de selección en base a los cromosomas, aplica el

algoritmo genético para hacer la decisión óptima de selección. El objetivo siempre es buscar el valor mayor, recuerde que una cadena binaria se convierte a un entero. El valor entero es el valor significativo para la toma de decisiones. En la figura 3.3 se muestra el diagrama de actividad del sistema.

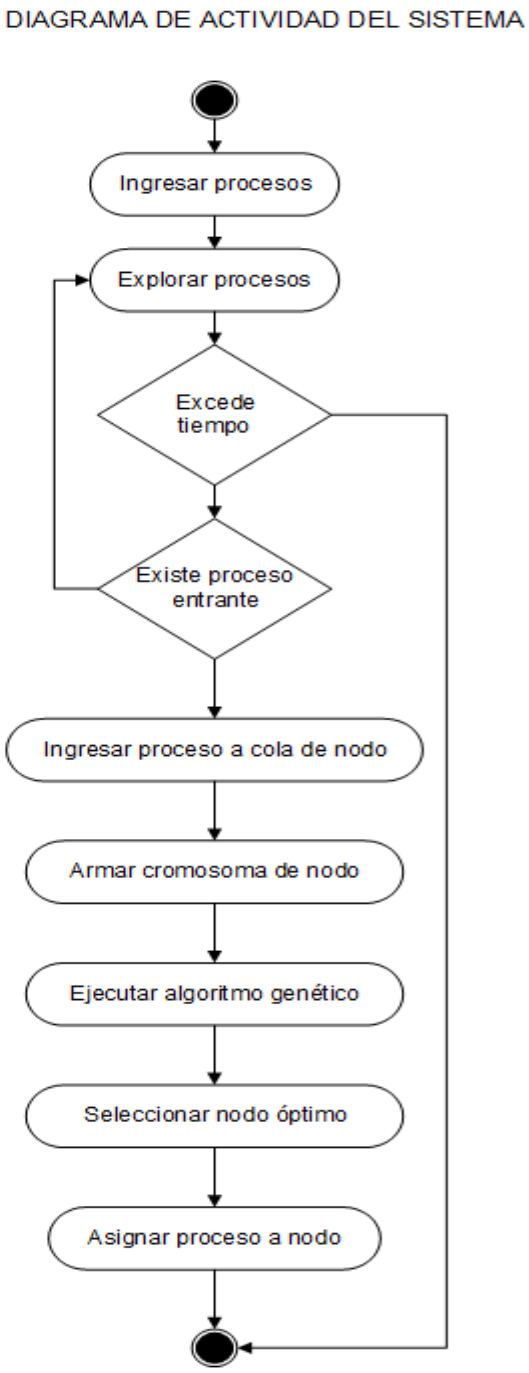


Figura 3.3. Diagrama de actividad del sistema.
Fuente análisis y diseño.

Observación. No se ha considerado el **diagrama del modelo de base de datos** del presente trabajo de investigación, y no se ha planteado inicialmente la base de datos.

3.4.4 Diagrama de clases

En esta sección, se muestra en la figura 3.4 el diagrama de clases:

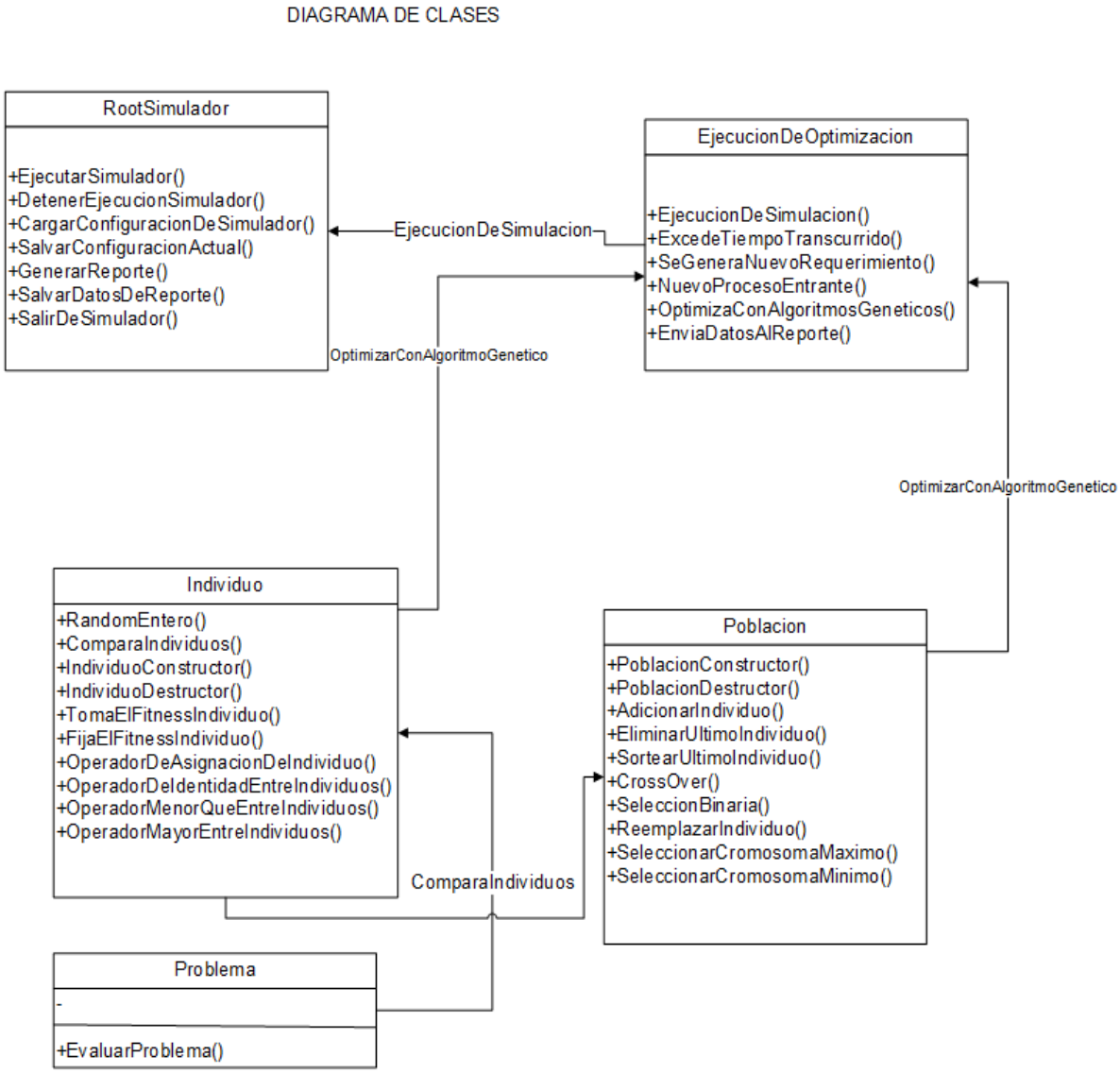


Figura 3.4. Diagrama de clases.
Fuente análisis y diseño.

i) RootSimulador

a) EjecutarSimulacion. Asigna los parámetros del simulador y crea el hilo principal del sistema que está conformado por EjecucionDeSimulacion() de la clase EjecucionDeOptimizacion.

b) DetenerEjecucionSimulador. Detiene el simulador, desasigna el hilo que ha sido asignado en la función anterior.

c) CargaConfiguracionDeSimulador. Función que carga archivo previamente generado por salvar la configuración del simulador.

d) SalvarConfiguracionActual. Salva la configuración del simulador. Parámetros que son configurados por el usuario antes de su ejecución. Estos pueden volver a ser leídos nuevamente por la función CargarConfiguracionActual().

e) GenerarReporte. Genera el reporte de una simulación o corrida previa.

f) SalvarDatosDelReporte. Guarda los datos de la corrida previa como un archivo de reporte.

ii) EjecucionDeOptimización

a) EjecucionDeSimulacion. Proceso principal de ejecución del simulador. Verifica el tiempo programado de la simulación. Verifica si hay procesos entrantes que deben ser asignados a los nodos. Optimizar con los algoritmos genéticos.

b) ExcedeTiempoTranscurrido. Verifica si el tiempo asignado para corrida del simulador coincide con el tiempo ejecutado. Si es verdad, avisa al proceso EjecucionDeSimulacion que ha llegado el momento de terminar la corrida.

c) SeGeneraNuevoRequerimiento. En función del tiempo asignado para la corrida, se genera un valor aleatorio para simular un nuevo requerimiento entrante.

d) NuevoProcesoEntrante. El nuevo requerimiento es el valor de tiempo que indica que debo seleccionar un proceso de la lista de procesos para asignarlo a la cola de un nodo.

e) OptimizaConAlgoritmosGenéticos. Selecciona el nodo óptimo y libera la cola en un proceso.

f) EnviaDatosAlReporte. Envía los datos generados por cada operación de optimización al reporte.

iii) Individuo

a) RandomEntero. Genera un valor entero aleatorio entre un rango. Es el valor aleatorio entre 0 y el valor entero máximo generado por un cromosoma.

b) CompараIndividuos. Compara 2 individuos. Toma el mayor valor entre el **fitness** (mejor aptitud) de 2 individuos.

c) IndividuoConstructor. Tiene 2 funciones: la primera convierte los cromosomas a valores enteros de los nodos en cola y permuta los cromosomas entre ellos. La segunda funcionalidad es crear un nuevo individuo como constructor copia.

d) IndividuoDestructor. Destruye un individuo.

e) TomaElFitnessIndividuo. Retorna el Fitness del individuo seleccionado.

f) FijaElFitnessIndividuo. Retorna el Fitness seleccionado.

g) SwapMutacion. Efectúa la mutación de un individuo, mediante el cambio aleatorio de 2 alelos de un cromosoma.

h) OperadorDeAsignacion. Sobre un objeto individuo construido asigna un objeto a otro. Asignacion entre objetos.

i) OperadorDeIdentidad. Compara 2 individuos.

j) OperadorMayorQue. Compara 2 individuos.

k) OperadorMenorQue. Compara 2 individuos.

iv) Poblacion

a) PoblacionConstructor. Instancia una nueva población.

b) PoblacionDestructor. Destruye la instancia de una población.

c) AdicionarIndividuo. Asigna un individuo a la población.

d) EliminarUltimoIndividuo. Elimina el último individuo.

e) Sortear. Sortea la lista de individuos de la población.

f) CrossOver. Efectúa el cruzamiento de 2 individuos.

g) SeleccionBinaria. Selecciona el mayor Fitness entre 2 individuos.

h) ReemplazarIndividuo. Reemplaza el peor individuo.

i) SeleccionarCromosomaMaximo. Selecciona el cromosoma máximo comparando la cadena binaria de cada individuo.

j) SeleccionarCromosomaMinimo. Selecciona el cromosoma mínimo comparando la cadena binaria de cada individuo.

v) Problema

EvaluarProblema. Asigna un valor aleatorio a un individuo como Fitness.

3.4.5 Diagrama de Entidad-Relación lógico.

El diagrama de entidades se muestra en la figura 3.5.

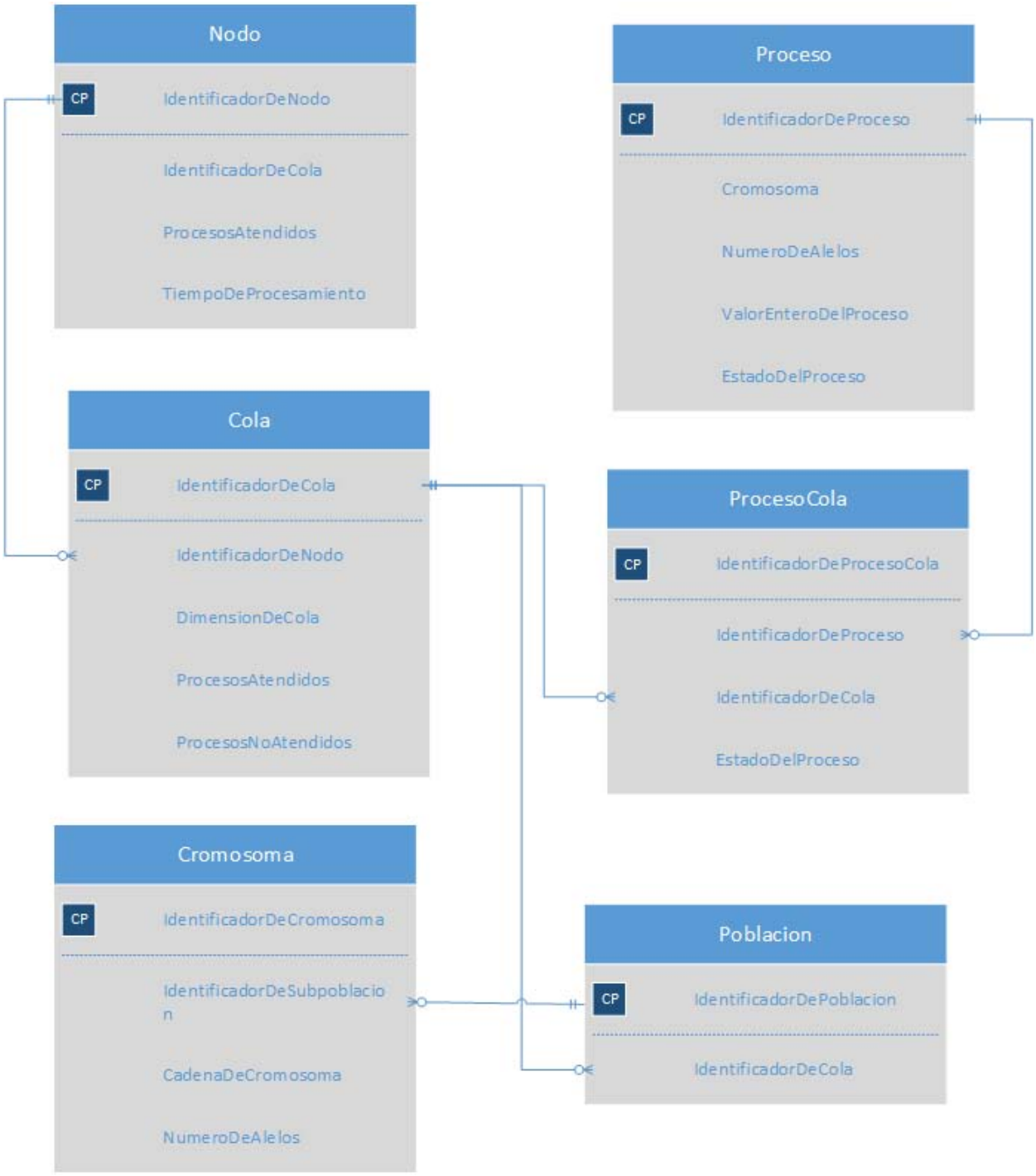


Figura 3.5. Diagrama de Entidad-Relación lógico.
Fuente análisis y diseño

a. Nodo. Identificador del nodo, identificador de la cola, procesos atendidos, tiempo de procesamiento.

b. Proceso. Identificador del proceso, cromosoma, número de alelos, valor entero del proceso, estado del proceso (proceso por asignar/proceso asignado).

c. Cola. Identificador de la cola, identificador del nodo, dimensión de la cola, procesos atendidos, procesos no atendidos.

d. ProcesoCola. Identificador del procesoCola, identificador del proceso, identificador de cola, estado del proceso (atendido/no atendido).

e. Población (subpoblación). Identificador de población, Identificador de la cola.

f. Cromosoma. Identificador del cromosoma, identificador de población, cadena del cromosoma, número de alelos.

3.4.6 Pseudocódigo

i) Simulador. El simulador es un hilo del aplicativo. De manera independiente se ejecuta hasta que el usuario aborte el proceso de simulación.

```
CrearHiloPrincipal (ExploracionDeSimulador);

ExploracionDeSimulador()
{
 if(ExisteNuevoRequerimiento())
 {
  ObtenerLosCromosomasDelosProcesosEnCola()
   BuscarElProcesoOptimo ()
   ProcesarElRequerimiento()
 }
 if(AbandonarExploracionDeRequerimiento)
  Salir()
}
```

ii) Explora la cola de los nodos. Si un nodo tiene procesos en cola, aplica los algoritmos genéticos a dicho conjunto de procesos y selecciona el óptimo.

```
ExisteProcesosEnCola()
{
 AsignarLosCromosomasALosProcesos()
}
```

iii) Construir los cromosomas de los procesos en cola. Para poder obtener el cromosoma de cada nodo que conforma la población, se selecciona en el rango de la longitud del cromosoma un valor entero. El valor finalmente consiste de una cadena binaria que es equivalente al valor entero.

```
ObtenerLosCromosomasPorCadaProcesos()
{
 AsignarLosValoresAlArrayDeIndividuos()
```

}

iv) Buscar nodo óptimo. El proceso de búsqueda del nodo elegido consiste en aplicar los algoritmos genéticos que toman como parámetro el array de los cromosomas de la población. Cada nodo conforma un individuo de la población. El objetivo es buscar el valor máximo del cromosoma.

```
BuscarNodoOptimo()
 {
   AsignarArrayDeCromosomasAAlgoritmoGenetico()
   EjecuciónDeAlgoritmoGenético()
   RetornoDeNodoOptimo()
 }
```

v) Asignar requerimiento a procesar a nodo.

```
AsignarRequerimientoANodo()
 {
   InformarDeProcesoAtendido()
   DisminuirLaCola()
   InformarDeColaLibre()
 }
```

3.5 Caracterización del sistema simulador

El proceso de simular asignación de procesos a nodos, está basado en la aplicación de algoritmos genéticos. Existe un proceso que se encarga de despachar los requerimientos a los nodos finales, los cuales se encargarán de agruparlos en la cola y seleccionar el óptimo a procesar.

Para el modelo, el sistema está en un proceso de asignación de requerimientos, estos son un conjunto de procesos en espera de ser asignados a los nodos. Para simular el cromosoma del requerimiento, el sistema genera un número entero en el rango de la longitud del cromosoma y como población toma la cantidad de procesos por atender en la cola de cada nodo.

El proceso de selección sobre la base de los cromosomas, aplica los algoritmos genéticos para hacer la decisión óptima de selección. El objetivo siempre es buscar el valor mayor, en una cadena binaria que ha convertido a un entero. El valor entero es el valor significativo para la toma de decisiones.

Los diagramas del simulador se muestran en la figura 3.6 arquitectura de procesos del simulador y figura 3.7 diagrama de ejecución del simulador, siguientes:

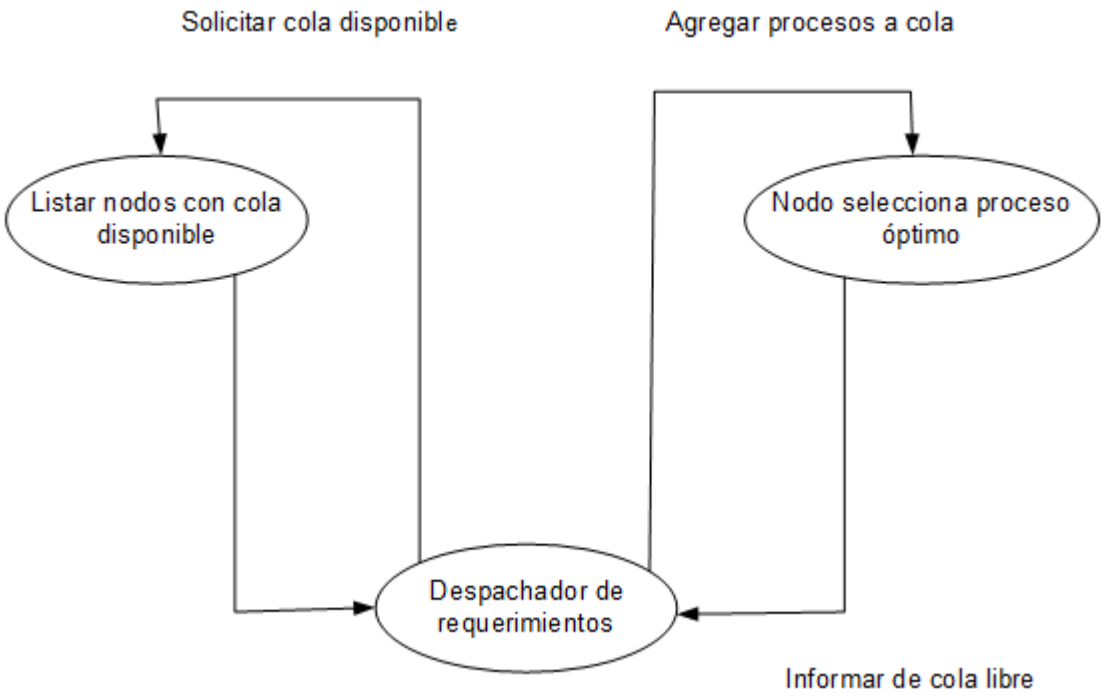
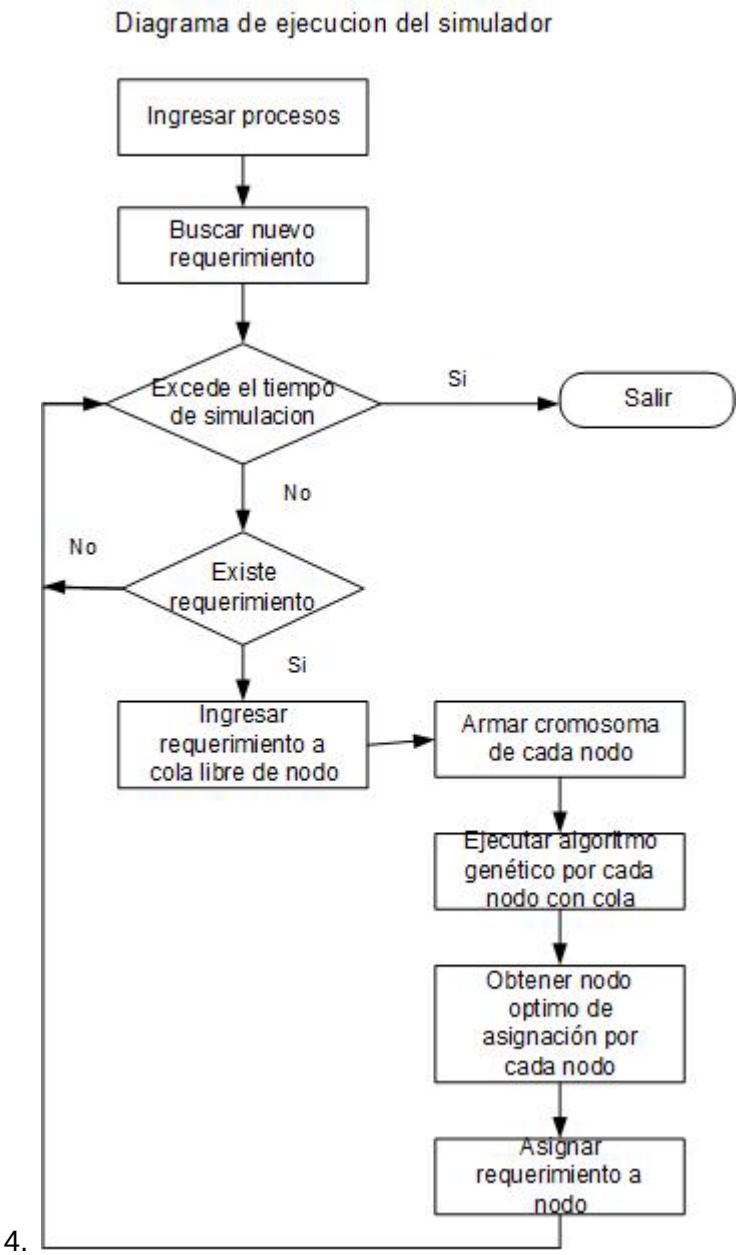


Figura 3.6. Arquitectura de procesos del simulador.
Fuente análisis y diseño.

Se hace un breve resumen de procesos de la arquitectura de procesos del simulador:

1. **Nodo selecciona proceso óptimo**. Clases que usan: RootSimulador, Individuo, Poblacion y EjecucionDeOptimizacion.
2. **Listar nodos con la disponible**. Clases que usan: EjecucionDeOptimizacion, Individuo, y Problema.
3. **Despachador de requerimientos**. Clases que usan: RootSimulador, Individuo, Poblacion, Problema y EjecucionDeOptimizacion.



4.

Figura 3.7. Diagrama de ejecución del simulador.
Fuente análisis y diseño.

CAPÍTULO 4

ANÁLISIS E INTERPRETACIÓN DE RESULTADOS

4.1 Presentación

El sistema consta de un conjunto de pasos para llegar a ejecutarlo, está organizado de manera secuencial. Primero, debe asignar la información para la opción Configuración inicial del sistema, y ejecute la opción de Cargar configuración que está incluido en la configuración inicial del sistema. En la opción de Ejecución puede iniciar el proceso de simulación y ver el reporte de salida en la opción Reportes.

Antes de configurar el sistema para su ejecución, se describe datos de configuración.sim siguiente:

[PARAMETROSINICIO]
TiempoSimulacion=0
NroProcesos=0
ProbCruzamiento=0.90
ProbMutacion=0.10
NroIteraciones=6
NroAlelos=6
[PARAMETROSFIN]
[HOSTSINI]
H1=3
H2=2
H3=3
H4=3
H5=2
H6=4
H7=3
H8=4
[HOSTFIN]

Se muestra en la figura 4.1 la ventana principal del simulador y presione botón Aceptar, aparecerá en la figura 4.2:

Figura 4.1. Ventana principal del simulador.
Fuente simulador.

4.2 Configuración inicial del sistema

Este es el paso inicial. Indique el tiempo de simulación en número de segundos. Indique el número de procesos por atender.

Inicialización genética. Indique la probabilidad de cruzamiento, mutación y el número de iteraciones que se ejecutarán al obtener el cromosoma óptimo, utilizando cromosomas de la población. El botón Guardar configuración le permite guardar todos los parámetros que usted ha configurado. Se muestra en la figura 4.2 la configuración inicial del simulador:

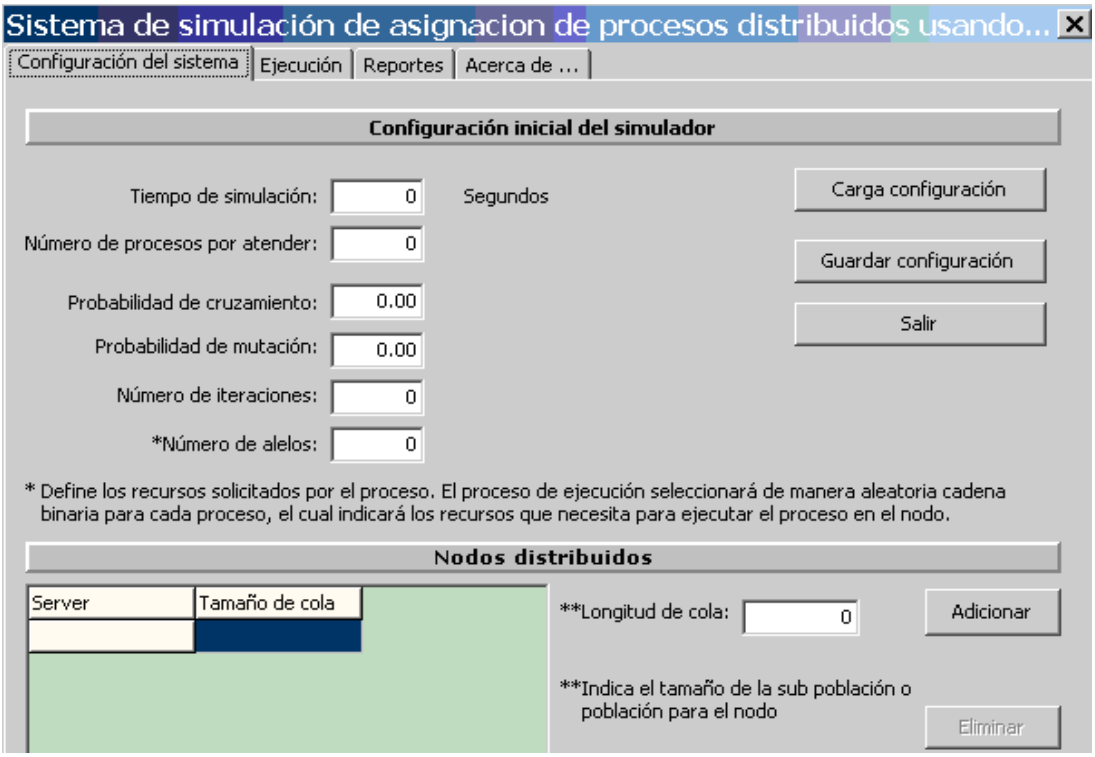


Figura 4.2. Configuración inicial del simulador.
Fuente simulador.

4.3 Carga archivo de configuración

El sistema tiene la opción de guardar todos los parámetros del sistema, en un archivo de configuración. Si usted tiene el archivo previamente guardado, puede cargarlo con la opción Carga configuración, ver figura 4.3 configuracion.sim:

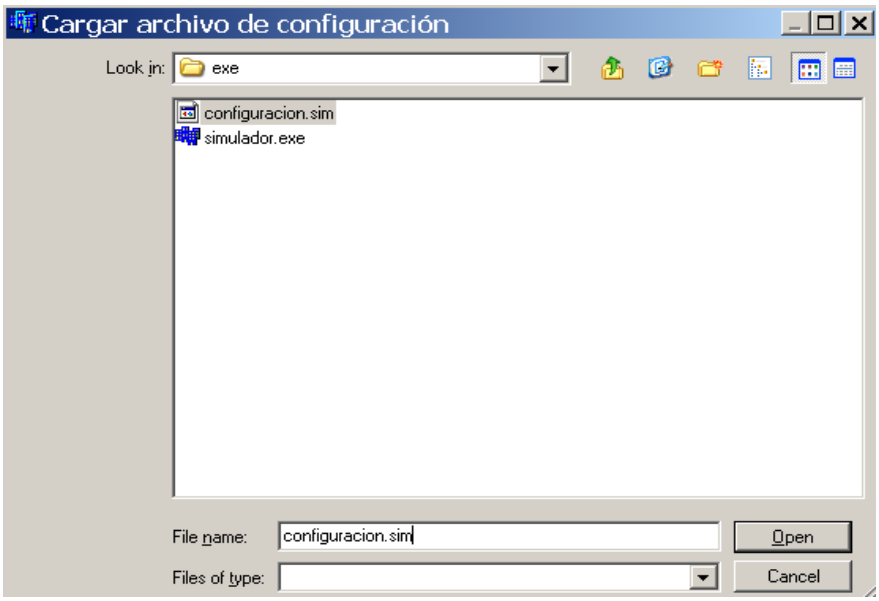
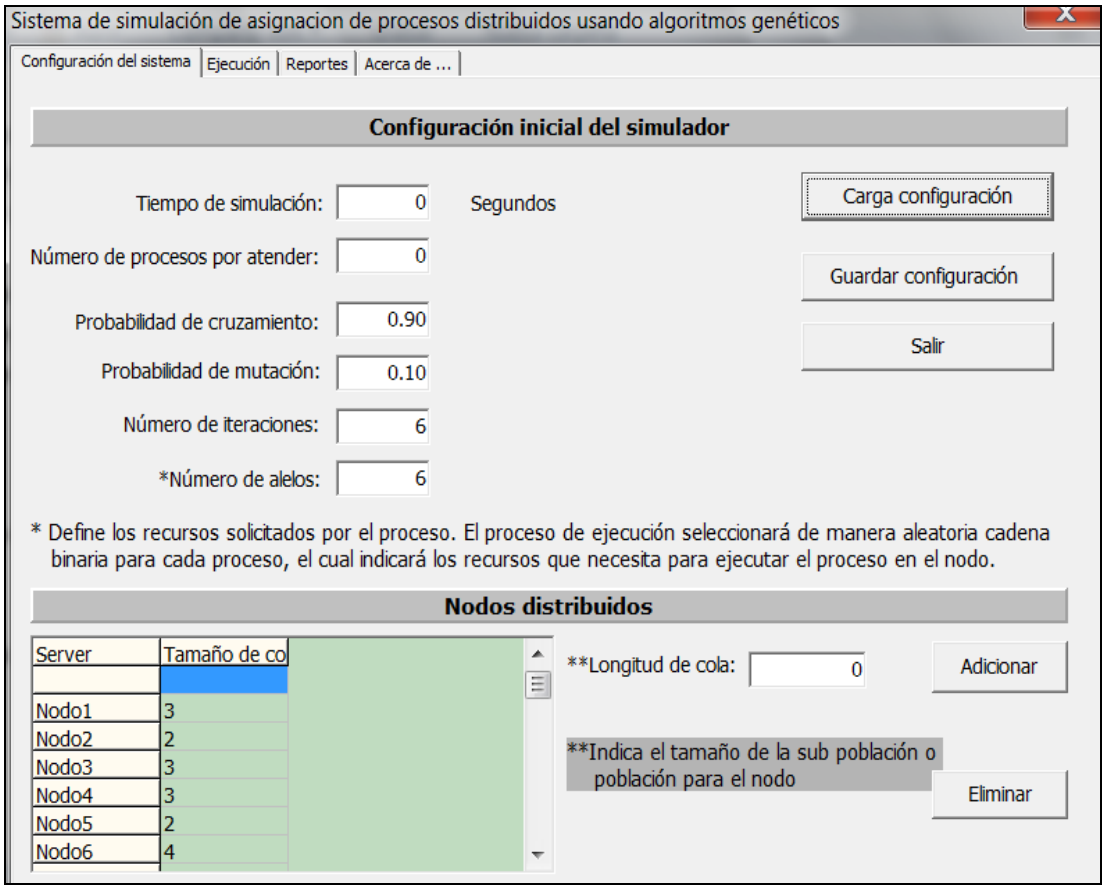


Figura 4.3. configuracion.sim.
Fuente simulador.

4.4 Configuracion.sim

Ahora seleccione configuracion.sim y pulse botón Open para abrir configuración inicial del simulador y nodos distribuidos del simulador. Se muestra en la figura 4.4, al inferior de la caja de diálogo para cargar el archivo de configuración con la extensión sim.

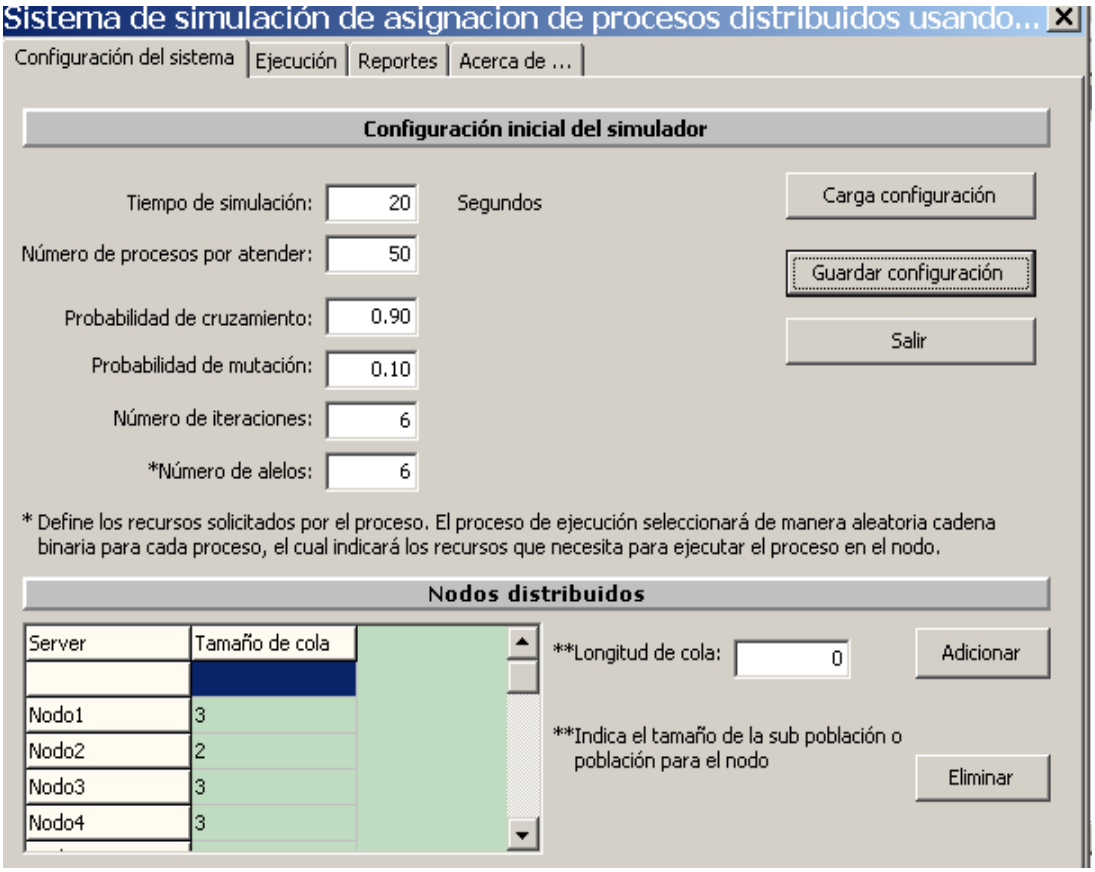
Como en el caso anterior, se muestra una caja de diálogo para que pueda guardar el archivo de configuración.



Figuras 4.4. Configuración inicial del simulador nodos distribuidos.
Fuente simulador.

En el archivo configuracion.sim están cargados los datos de: probabilidad de cruzamiento 0.90, probabilidad de mutación 0.10, número de iteraciones 6 y número de alelos 6.

Ahora al simulador se asignan 20 segundos de tiempo de ejecución y 50 procesos; y se muestra en la figura 4.5 y estas asignaciones varían de acuerdo a las cantidades especificadas siguientes:



Figuras 4.5. Configuración inicial del simulador nodos distribuidos y datos cargados. Fuente simulador.

4.5 Ejecución

En la ventana de ejecución muestra 3 bloques. El primer bloque es 'información del simulador'. Este bloque muestra la evolución de los tiempos de simulación de los procesos configurados. Varía según evoluciona la simulación.

El bloque 2 muestra 'proceso asignado', que es el proceso que el despachador está asignando a los nodos. Se muestra un identificador del requerimiento, el nodo responsable y tiempo que se requiere para atenderlo.

La grilla muestra el comportamiento dinámico de los nodos. Indica cómo se comporta las colas y los procesos atendidos por los nodos. Pulse la pestaña Ejecución, entonces aparecerá la ventana de ejecución inicial del simulador y se muestra en la figura 4.6 siguiente:

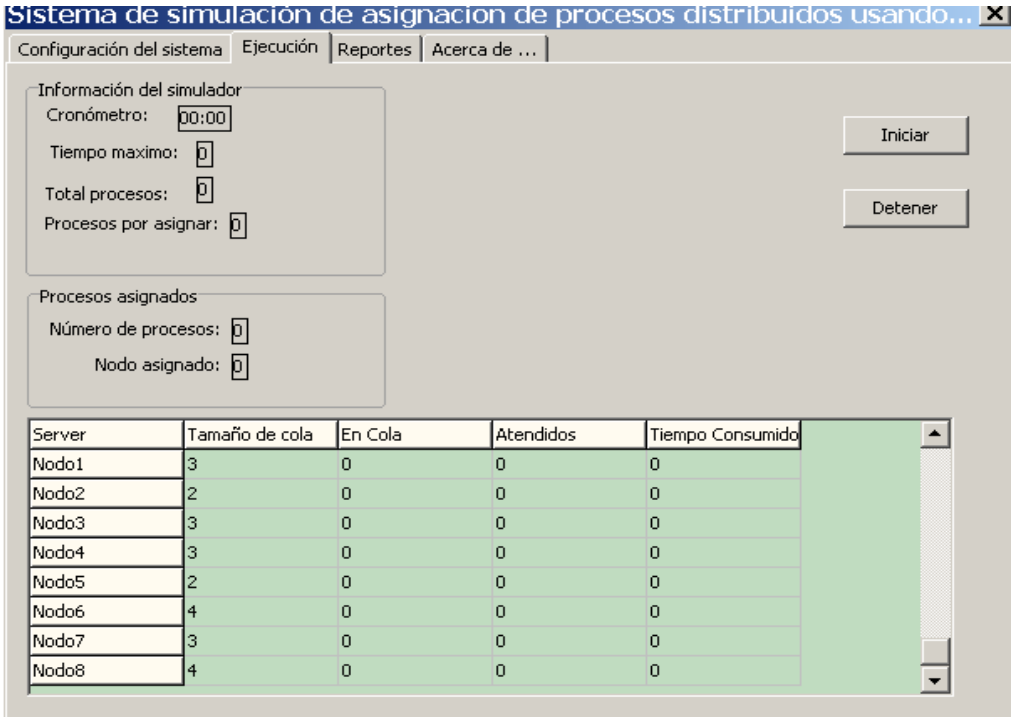


Figura 4.6. Ventana de ejecución inicial del simulador.
Fuente simulador.

Ahora pulse botón **Iniciar** del simulador, entonces empieza a procesar los datos en simulador y se muestra en figura 4.7:

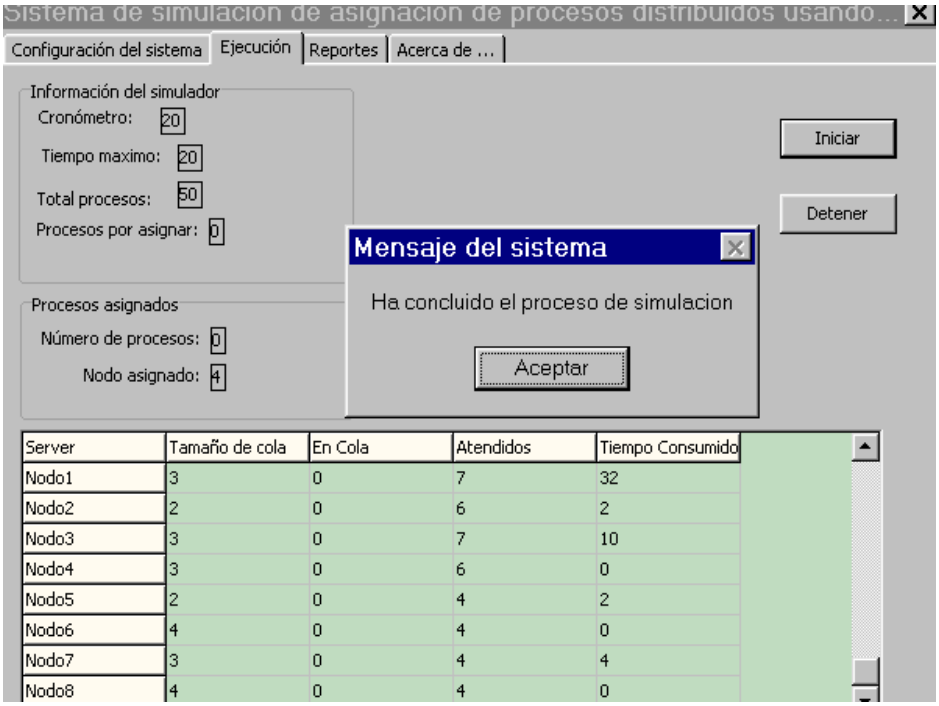


Figura 4.7. Simulador ejecuta los datos y termina.
Fuente simulador.

Los nodo1 al nodo8, el tamaño de la cola están establecidos de (3, 2, 3, 3, 2, 4, 3 y 4) del servidor. **En cola;** una vez terminado los procesos quedan en ceros las colas.

En el resultado del proceso del simulador de algoritmos genéticos, se puede observar que el nodo1 y nodo3 han atendido 7 tareas (individuos) cada uno, son cantidades máximos procesados, mientras que el nodo5, nodo6, nodo7 y nodo8 han atendido máximos procesados cuatro tareas cada uno, es la cantidad mínima procesado.

En caso del tiempo consumido puede observarse que el nodo1 ha utilizado 32 milisegundos; que es tiempo máximo de proceso, y el resto son menores.

En conclusión, como se ha observado el procesamiento de nodos con algoritmos genéticos son rápidos, han sido utilizados en procesos paralelos, que anteriormente han sido experimentados por investigadores de algoritmos genéticos.

4.6 Reportes

El sistema muestra en línea la evolución de la simulación. Está se envía a un reporte de salida. Usted puede guardar el reporte en un archivo de texto. Seleccione el botón Guardar para salvar reporte.

Pulse la pestaña Reportes para presentar resultado del simulador, que se muestra en la figura 4.8 reportes, siguiente:

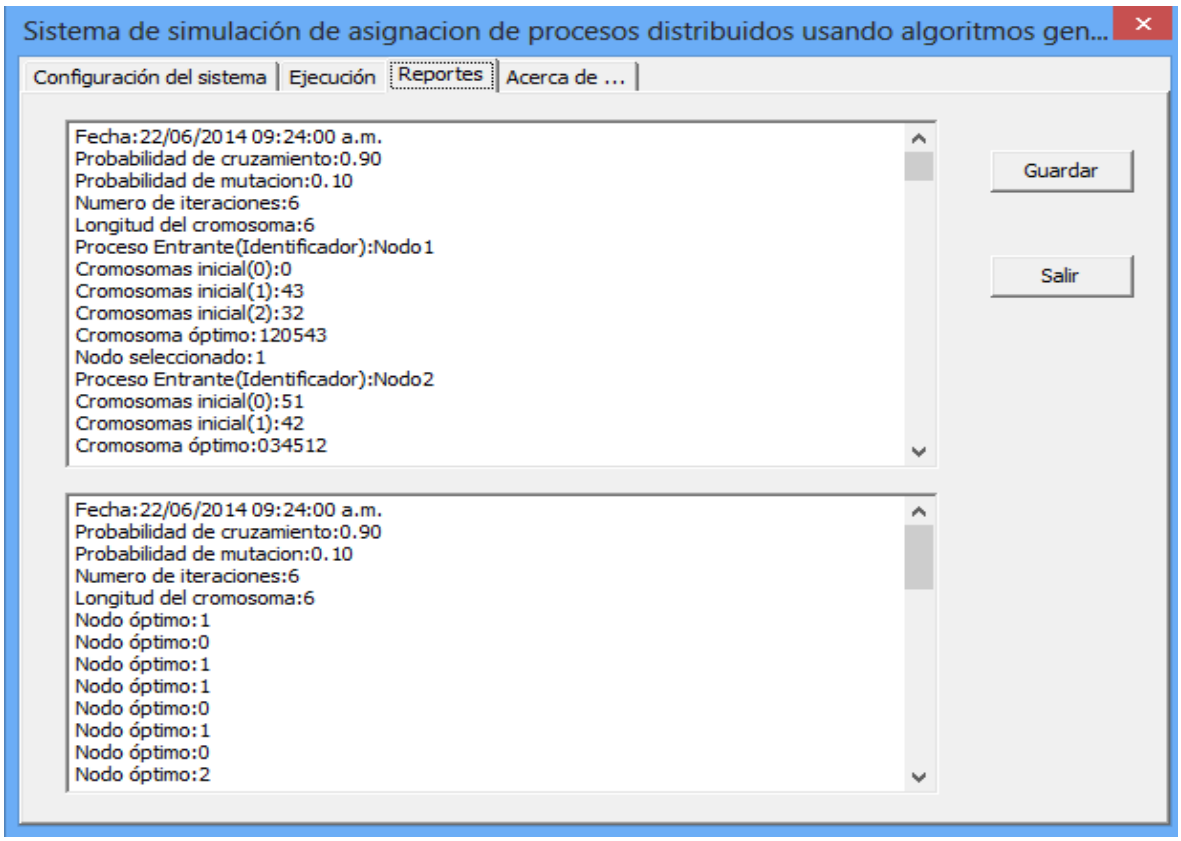


Figura 4.8. Reportes.
Fuente simulador

4.6.1 Listado de procesos de tareas

Fecha:22/06/2014 09:24:00 a.m.
Probabilidad de cruzamiento:0.90
Probabilidad de mutacion:0.10
Numero de iteraciones:6
Longitud del cromosoma:6
Proceso Entrante(Identificador):Nodo1
Cromosomas inicial(0):0
Cromosomas inicial(1):43
Cromosomas inicial(2):32
Cromosoma óptimo:120543
Nodo seleccionado:1
Proceso Entrante(Identificador):Nodo2
Cromosomas inicial(0):51
Cromosomas inicial(1):42
Cromosoma óptimo:034512
Nodo seleccionado:0
Proceso Entrante(Identificador):Nodo3
Cromosomas inicial(0):7
Cromosomas inicial(1):57
Cromosomas inicial(2):11
Cromosoma óptimo:531402
Nodo seleccionado:1
Proceso Entrante(Identificador):Nodo4
Cromosomas inicial(0):43
Cromosomas inicial(1):43
Cromosomas inicial(2):29
Cromosoma óptimo:105243
Nodo seleccionado:1
Proceso Entrante(Identificador):Nodo5
Cromosomas inicial(0):8
Cromosomas inicial(1):45
Cromosoma óptimo:021354
Nodo seleccionado:0
Proceso Entrante(Identificador):Nodo6
Cromosomas inicial(0):60
Cromosomas inicial(1):59
Cromosomas inicial(2):8
Cromosomas inicial(3):36
Cromosoma óptimo:031425
Nodo seleccionado:1
Proceso Entrante(Identificador):Nodo7
Cromosomas inicial(0):51
Cromosomas inicial(1):5
Cromosomas inicial(2):55
Cromosoma óptimo:102345
Nodo seleccionado:0
Proceso Entrante(Identificador):Nodo8
Cromosomas inicial(0):52
Cromosomas inicial(1):53
Cromosomas inicial(2):61
Cromosomas inicial(3):32
Cromosoma óptimo:154032

Nodo seleccionado:2
Proceso Entrante(Identificador):Nodo1
Cromosomas inicial(0):63
Cromosomas inicial(1):17
Cromosoma óptimo:431250
Nodo seleccionado:0
Proceso Entrante(Identificador):Nodo2
Cromosomas inicial(0):17
Cromosoma óptimo:105234
Nodo seleccionado:0
Proceso Entrante(Identificador):Nodo3
Cromosomas inicial(0):16
Cromosomas inicial(1):19
Cromosoma óptimo:542031
Nodo seleccionado:0
Proceso Entrante(Identificador):Nodo4
Cromosomas inicial(0):21
Cromosomas inicial(1):51
Cromosoma óptimo:143025
Nodo seleccionado:1
Proceso Entrante(Identificador):Nodo5
Cromosomas inicial(0):19
Cromosoma óptimo:243105
Nodo seleccionado:0
Proceso Entrante(Identificador):Nodo6
Cromosomas inicial(0):20
Cromosomas inicial(1):35
Cromosomas inicial(2):53
Cromosoma óptimo:541320
Nodo seleccionado:2
Proceso Entrante(Identificador):Nodo7
Cromosomas inicial(0):27
Cromosomas inicial(1):36
Cromosoma óptimo:124503
Nodo seleccionado:0
Proceso Entrante(Identificador):Nodo8
Cromosomas inicial(0):48
Cromosomas inicial(1):1
Cromosomas inicial(2):36
Cromosoma óptimo:012345
Nodo seleccionado:1
Proceso Entrante(Identificador):Nodo1
Cromosomas inicial(0):33
Cromosomas inicial(1):8
Cromosoma óptimo:013542
Nodo seleccionado:0
Proceso Entrante(Identificador):Nodo2
Cromosomas inicial(0):54
Cromosomas inicial(1):51
Cromosoma óptimo:230145
Nodo seleccionado:0
Proceso Entrante(Identificador):Nodo3
Cromosomas inicial(0):29
Cromosomas inicial(1):33

Cromosoma óptimo:354012
Nodo seleccionado:0
Proceso Entrante(Identificador):Nodo4
Cromosomas inicial(0):10
Cromosomas inicial(1):29
Cromosoma óptimo:142035
Nodo seleccionado:1
Proceso Entrante(Identificador):Nodo5
Cromosomas inicial(0):18
Cromosomas inicial(1):61
Cromosoma óptimo:314052
Nodo seleccionado:1
Proceso Entrante(Identificador):Nodo6
Cromosomas inicial(0):61
Cromosomas inicial(1):33
Cromosoma óptimo:051324
Nodo seleccionado:0
Proceso Entrante(Identificador):Nodo7
Cromosomas inicial(0):53
Cromosomas inicial(1):24
Cromosoma óptimo:510423
Nodo seleccionado:0
Proceso Entrante(Identificador):Nodo8
Cromosomas inicial(0):44
Cromosomas inicial(1):31
Cromosoma óptimo:412305
Nodo seleccionado:0
Proceso Entrante(Identificador):Nodo1
Cromosomas inicial(0):34
Cromosoma óptimo:435012
Nodo seleccionado:0
Proceso Entrante(Identificador):Nodo2
Cromosomas inicial(0):12
Cromosoma óptimo:350412
Nodo seleccionado:0
Proceso Entrante(Identificador):Nodo3
Cromosomas inicial(0):7
Cromosoma óptimo:245103
Nodo seleccionado:0
Proceso Entrante(Identificador):Nodo4
Cromosomas inicial(0):34
Cromosoma óptimo:235410
Nodo seleccionado:0
Proceso Entrante(Identificador):Nodo5
Cromosomas inicial(0):27
Cromosoma óptimo:410253
Nodo seleccionado:0
Proceso Entrante(Identificador):Nodo6
Cromosomas inicial(0):49
Cromosoma óptimo:052431
Nodo seleccionado:0
Proceso Entrante(Identificador):Nodo7
Cromosomas inicial(0):63
Cromosoma óptimo:531402

Nodo seleccionado:0
Proceso Entrante(Identificador):Nodo8
Cromosomas inicial(0):55
Cromosoma óptimo:143520
Nodo seleccionado:0
Proceso Entrante(Identificador):Nodo1
Cromosomas inicial(0):16
Cromosomas inicial(1):30
Cromosomas inicial(2):1
Cromosoma óptimo:250431
Nodo seleccionado:0
Proceso Entrante(Identificador):Nodo2
Cromosomas inicial(0):17
Cromosomas inicial(1):0
Cromosoma óptimo:032451
Nodo seleccionado:0
Proceso Entrante(Identificador):Nodo3
Cromosomas inicial(0):46
Cromosomas inicial(1):20
Cromosomas inicial(2):24
Cromosoma óptimo:412035
Nodo seleccionado:1
Proceso Entrante(Identificador):Nodo4
Cromosomas inicial(0):15
Cromosomas inicial(1):29
Cromosoma óptimo:215034
Nodo seleccionado:1
Proceso Entrante(Identificador):Nodo1
Cromosomas inicial(0):32
Cromosomas inicial(1):28
Cromosoma óptimo:523140
Nodo seleccionado:0
Proceso Entrante(Identificador):Nodo2
Cromosomas inicial(0):26
Cromosoma óptimo:530124
Nodo seleccionado:0
Proceso Entrante(Identificador):Nodo3
Cromosomas inicial(0):20
Cromosomas inicial(1):48
Cromosoma óptimo:021543
Nodo seleccionado:0
Proceso Entrante(Identificador):Nodo4
Cromosomas inicial(0):18
Cromosoma óptimo:514320
Nodo seleccionado:0
Proceso Entrante(Identificador):Nodo1
Cromosomas inicial(0):16
Cromosoma óptimo:432105
Nodo seleccionado:0
Proceso Entrante(Identificador):Nodo3
Cromosomas inicial(0):32
Cromosoma óptimo:104235
Nodo seleccionado:0

4.6.2 Listado de procesos óptimos

Fecha:22/06/2014 09:24:00 a.m.
Probabilidad de cruzamiento:0.90
Probabilidad de mutacion:0.10
Numero de iteraciones:6
Longitud del cromosoma:6
Nodo óptimo:1
Nodo óptimo:0
Nodo óptimo:1
Nodo óptimo:1
Nodo óptimo:0
Nodo óptimo:1
Nodo óptimo:0
Nodo óptimo:2
Nodo óptimo:0
Nodo óptimo:0
Nodo óptimo:0
Nodo óptimo:1
Nodo óptimo:0
Nodo óptimo:2
Nodo óptimo:0
Nodo óptimo:1
Nodo óptimo:0
Nodo óptimo:0
Nodo óptimo:0
Nodo óptimo:1
Nodo óptimo:1
Nodo óptimo:0
Nodo óptimo:0
Nodo óptimo:0
Nodo óptimo:0
Nodo óptimo:0
Nodo óptimo:0
Nodo óptimo:0
Nodo óptimo:0
Nodo óptimo:0
Nodo óptimo:0
Nodo óptimo:0
Nodo óptimo:0
Nodo óptimo:0
Nodo óptimo:0
Nodo óptimo:1
Nodo óptimo:1
Nodo óptimo:0
Nodo óptimo:0
Nodo óptimo:0
Nodo óptimo:0
Nodo óptimo:0
Nodo óptimo:0

4.7 Acerca del sistema

Ahora presione la opción Acerca del sistema y aparecerá la ventana de ello, trata sobre la versión y el autor del sistema aplicativo, y se muestra en la figura 4.9 acerca del sistema.

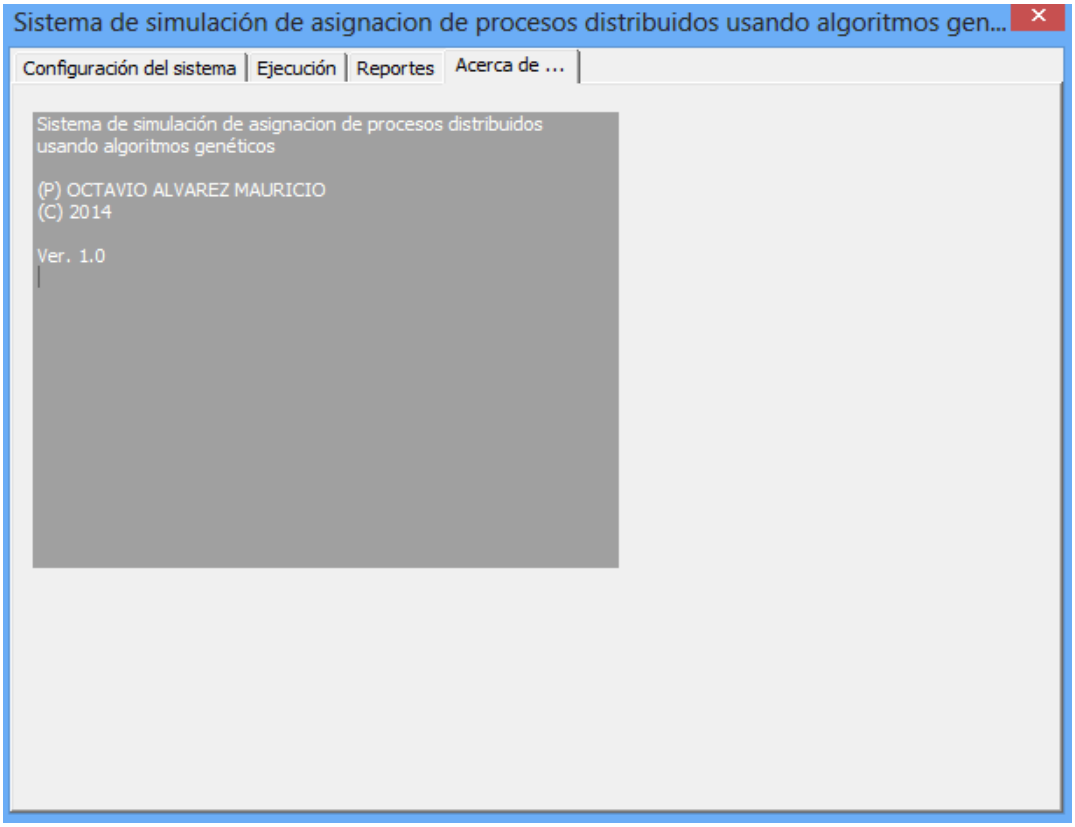


Figura 4.9. Acerca del sistema.
Fuente simulador.

4.8 Ejemplo de comparación de procesos realizados con algoritmos genéticos

a) Del capítulo II de "algoritmos genéticos asíncronos combinados para una red heterogénea de computadoras", se muestra en la tabla 4.1 los procesos realizados con algoritmos genéticos, el efecto de desbalancear las cargas, variando la cantidad de individuos a ser asignados a cada procesador. Como es de esperar, los tiempos de procesamiento dependen del balanceamiento de carga en general, la utilización de algoritmos combinados diferentes en cada procesador resulta casi siempre benéfica, sobre todo cuando las cargas están bien balanceadas. Por ejemplo, se puede observar que los mejores desempeños del algoritmo se obtuvieron al combinar los algoritmos genéticos paralelos con el método del gradiente (m1) en un procesador y el de métrica variable en el otro (m4).

Tabla 2.12. Resultados para diferentes participaciones.
Fuente [4].

		PROCESOS REALIZADOS DE 1995				
Función	Desempeño	Desempeño	Tiempo	Subpoblación	Método	Método
de Jong	óptimo	experimental	(segundos)	SUN	SUN	DEC
F1	3.817958	3.56356	3.517408	20	m1	m1
F2	3.817958	3.56381	2.068437	30	m1	m1
F3	3.817958	3.817958	1.448785	40	m1	m1
F4	3.817958	3.817958	2.236543	50	m1	m1
F5	3.817958	3.817958	1.268846	20	m4	m4
F6	3.817958	3.817958	3.086533	30	m4	m4
F7	3.817958	3.817958	2.55616	40	m4	m4
F8	3.817958	3.817958	2.981364	50	m4	m4
F9	3.817958	3.817958	1.574217	50	m1	m4
F10	3.817958	3.817958	2.054251	50	m4	m1
F11	3.817958	3.817958	1.040199	20	m1	m4
F12	3.817958	3.817958	1.085705	20	m4	m1
F13	3.817958	3.817958	2.96558	40	m1	m4
F14	3.817958	3.817958	3.391128	40	m4	m1

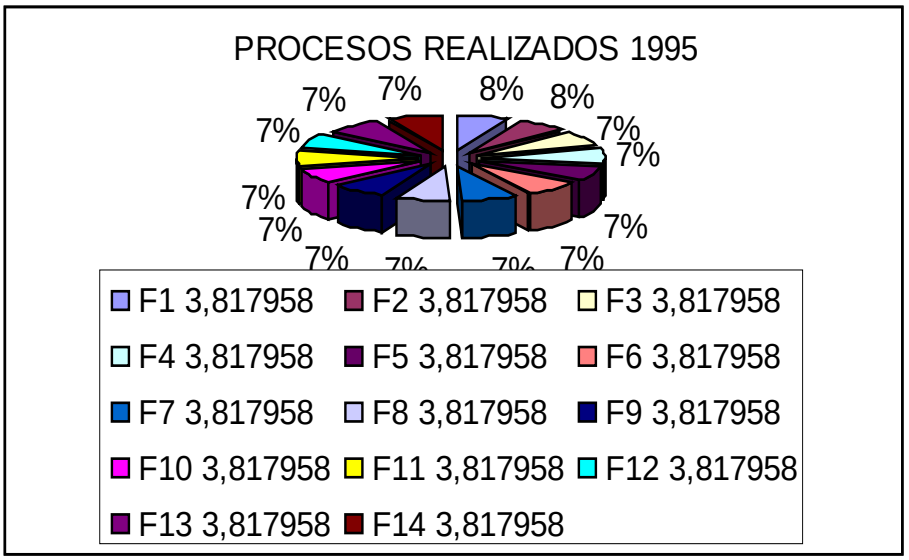


Figura 4.10. Resultados de procesos realizados 1995.
Fuente simulador.

De la tabla 2.12 se observa los resultados para diferentes participaciones de la implementación paralela que mejoran tiempos de computación de algoritmos genéticos en una relación que puede llegar a ser súperlineal, llegando inclusive a mejores soluciones que los algoritmos secuenciales al trabajar con subpoblaciones independientes [3]. De los resultados de procesos realizados de la figura 4.10, que se

observa la representación 7% y 8% los tiempos de procesos que tienen poca variación de cada proceso de subpoblación, además, se determinan:

- La superioridad de las implementaciones asíncronas de implementaciones más tradicionales secuenciales y síncronas.
- La utilidad de combinar con otros métodos de optimización con algoritmos genéticos (secuenciales).
- La ventaja de utilizar diferentes métodos en cada uno de los procesadores de un sistema computacional heterogéneo de población y aleatoriedad.
- La posibilidad de mejorar los tiempos de ejecución y desempeño general del algoritmo combinado efectos de paralelos, que a los algoritmos genéticos.

b) Del capítulo IV de "análisis e interpretación de resultados", se muestra en la tabla 4.1 los resultados de procesos realizados con algoritmos genéticos, el efecto de distribuir equitativamente las cargas, variando la cantidad de individuos a ser asignados a cada proceso. Como es de esperar, los tiempos (milisegundos) de procesamiento dependen de la asignación equitativa de carga en general, la utilización de algoritmos genéticos en el procesador resulta casi siempre benéfica. Por ejemplo, se puede observar que los mejores desempeños del algoritmo se obtuvieron al combinar los algoritmos genéticos paralelos en los procesos de tareas del procesador.

Tabla 4.1. Resultados de procesos realizados.
Fuente simulador.

RESULTADOS DE PROCESOS REALIZADOS 2014				
Server	Tamaño de cola	En cola	Atendidos	Tiempo consumido
Nodo1	3	0	7	32
Nodo2	2	0	6	2
Nodo3	3	0	7	10
Nodo4	3	0	6	0
Nodo5	2	0	4	2
Nodo6	4	0	4	0
Nodo7	3	0	4	4
Nodo8	4	0	4	0

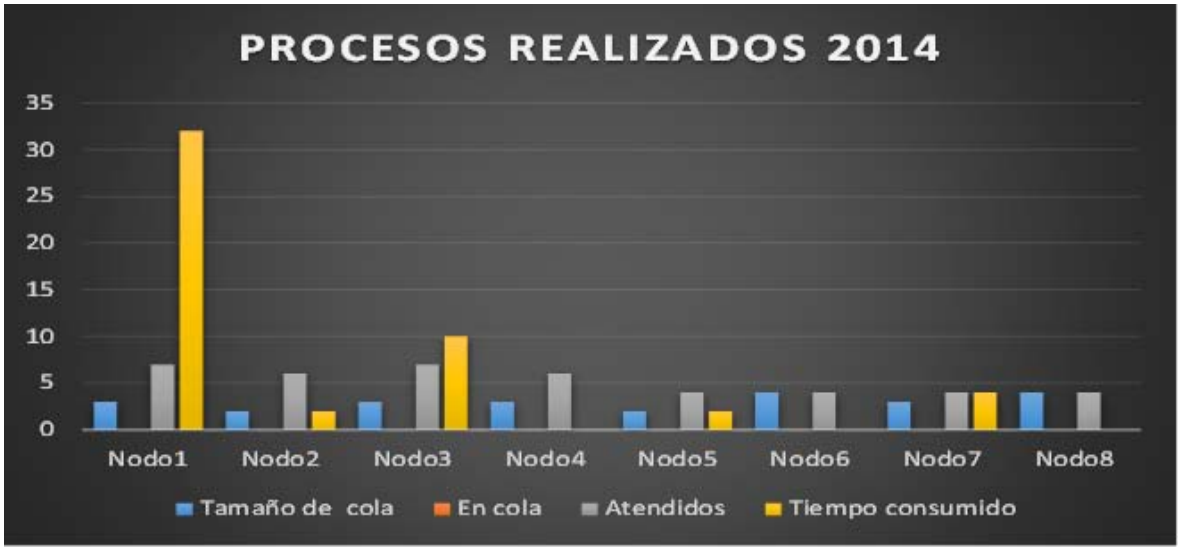


Figura 4.11. Resultados de procesos realizados.
Fuente simulador.

De la tabla 4.1 los procesos realizados se observan las implementaciones de procesamiento de tareas (cromosomas) con algoritmos genéticos, que mejoran tiempos de computación de algoritmos tradicionales, llegando inclusive a mejores soluciones que los algoritmos secuenciales al trabajar con subpoblaciones independientes. Los resultados de la figura 4.11 se observa los resultados de procesos realizados, que representan 8%, 13%, y 16% los tiempos (milisegundos) de procesos tienen variación de cada proceso de subpoblación; el tiempo mínimo utilizado es 8% y tiempo máxima utilizado es 16%, además, determinan:

- Mejoran implementaciones asíncronas de más tradicionales implementaciones secuenciales y síncronas.
- Permite ejecutar varias colas (cromosomas) en el simulador.
- Ventaja de utilizar carga equitativa en cada uno de los procesadores de un sistema computacional heterogéneo de población y aleatoriedad.
- Mejoran tiempos de ejecución y desempeño general con algoritmos genéticos.

En conclusión, algoritmos genéticos asíncronos combinados para una red heterogénea de computadoras; que ha reconocido su potencialidad de algoritmos genéticos paralelos que pueden ser aprovechado no-solo en los centros que cuentan con importantes computadoras, sino por cualquier institución o empresa con acceso a una red de computadoras, por más heterogéneas que puedan ser sus máquinas y/o software, pues implementaciones como la descrita ha sido permitido el uso eficiente de los recursos computacionales disponibles actual y en el futuro.

Por lo expuesto, se ha implementado el “simulador prototipo de aplicación utilizando algoritmos genéticos para optimizar procesos en comunicaciones”; para tesina de

“Sistemas de Comunicación Utilizando Algoritmos Genéticos Para Optimizar Procesos”, que permitirá reducir los cálculos de procesos (tareas) en servidor, menos costo de recursos computacionales y el resultado devolverá a los usuarios / clientes en menos tiempo de procesos de tareas, está realiza a través de líneas comunicaciones del sistema de local y/o remoto. Dicho simulador se puede utilizar en los centros de cómputo de las empresas, instituciones, y centros de investigaciones.

El modelo propuesto del simulador de aplicación algoritmos genéticos para optimizar procesos de tareas en comunicaciones, simulador evaluará los procesos de tareas de: número población para proceso, número de individuos, probabilidad de cruce, probabilidad de mutación, colas atendidas y no atendidas, tiempo consumido, resultados de procesos óptimos. El tiempo de procesamiento de datos es menor que el tiempo de procesamiento de datos que el convencional en el procesador.

CONCLUSIONES

EL desarrollo de la investigación de la tesina de “Sistemas de Comunicación Utilizando Algoritmos Genéticos Para Optimizar Procesos”, que ha permitido llegar a las siguientes conclusiones:

1. La evolución de Internet e Internet 2 han llegado a una alta proliferación y rendimiento en consultas de informaciones del servidor web por usuarios / clientes del mundo.
2. Hay deficiencia en el tiempo de procesos de tareas por procesador del sistema; que están trabajando con algoritmos tradicionales. Por ejemplo, retardo en la cola de espera, retardo en tiempo de proceso de tareas, retardo en tiempo de respuesta, que ocasionan un efecto de cuello de botella, falta memoria, etcétera.
3. Por el avance de la tecnología de informática y comunicación, es necesario estudiar los procesos del sistema de comunicación en una red, de tal manera que se aplique otra alternativa como los algoritmos genéticos para optimizar los procesos en el servidor y para mejorar servicios de atención a los clientes / usuarios.
4. Se desarrolla con algoritmos genéticos un simulador prototipo para solucionar la mejora de procesos de tareas en el procesador del servidor.
5. La implementación de un simulador prototipo para sistemas de comunicación utilizando algoritmos genéticos para optimizar procesos, antes de proceso de datos se cargarán datos iniciales para procesos de tareas como tiempo de proceso, número de proceso, probabilidad de cruce, probabilidad de mutación, iteraciones y alelos; después de proceso de datos se tiene como cola, atendidos, tiempo consumido, resultados de procesos óptimos.
6. El tiempo de procesamiento de datos es menor que el tiempo de procesamiento de datos convencional en el procesador, se consigue mejor uso de los recursos del sistema y mejora la calidad de servicio usuarios / clientes.

GLOSARIO

Agencia de proyectos avanzados de investigación para la defensa (**DARPA**: Defense Advanced Research Projects Agency). Se trata de una agencia de investigación del Departamento de Defensa de Estados Unidos.

Alelos. Recursos del sistema.

Cromosoma. Conjunto de alelos o individuos. En genética, cromosoma es una cadena de **Ácido DesoxirriboNucleico** (ADN) enrollada dentro del núcleo celular.

Gigabit en presencia de punto (**gigaPoP**: Gigabit in point presence). Es la distribución de direcciones del servidor web principal a diferentes servidores web en la Internet 2.

Grupos de trabajo (**WG**: Working groups). Agrupación de trabajadores de 2 o más personas especializados en la Internet 2.

Individuo. Es un proceso del sistema.

Instituto de Ingenieros Eléctricos y Electrónicos / Instituto Nacional Americano de Estándares (**IEEE/ANSI**: Institute of Electrical and Electronic Engineers/American National Standards Institute).

Miles de billones de bits por segundo (**Gbps**: Gigabits per second). Se refiere a miles de billones de bits por segundo.

Oficina de proyectos avanzados de investigación (**ARPA**: Advanced Research Projects Agency). Nombre con que se conocía a la agencia del gobierno estadounidense que financió la creación de la ARPANET y posteriormente denominado Internet; ahora es la DARPA.

Protocolo de control de transmisión / protocolo de Internet (**TCP/IP**: Transmission Control Protocol / Internet Protocol). Una serie de protocolos de redes de telecomunicaciones utilizada por Internet, intranets y extranets; que se ha convertido en un estándar de facto de la arquitectura de redes para las empresas o instituciones. Grupo de estándares para transmisión de datos y corrección de errores que permite la transferencia de información desde una computadora enlazada a Internet hacia otra del mismo tipo; que parte los mensajes en pequeños paquetes y que de este modo asegura su correcta recepción.

Protocolo de Internet versión 6 (**IPv6:** Internet protocol version 6). Protocolo específico que ha sido propuesto por grupo de ingeniería de la Internet (IETF) como sucesor del IPv4. El protocolo de Internet versión 6 (IPv6) utiliza direcciones de 128 bits.

Red de área amplia / extensa (**WAN**: Wide area network). Es una red de gran alcance con un sistema de comunicaciones que interconectan redes geográficamente remotas,

utilizando servicios por empresas de servicio público como comunicaciones vía telefónica o en ocasiones instalados por la misma organización. Los hosts se encuentran conectados a subredes de comunicaciones, cuya función es conducir mensajes de un host a otro, a distancia del sistema telefónico que conducen voz, vídeo, datos; los hosts conducen datos utilizando la vía de red telefónica y satélites; por ejemplo, en diferentes ciudades.

Red de área local (**LAN**: Local Area Network). Es la red de propiedad privada o institucional que funcionan dentro de una oficina, edificio o terreno hasta unos kilómetros, conectadas bajo un mismo protocolo y tipo de conexión física, sin modulación de la señal y en distancias cortas hasta 10 Km.

Red de área metropolitana (**MAN**: Metropolitan area network). Es básicamente una versión mayor que las redes de área local, con una tecnología bastante similar. Una red de área metropolitana puede manejar voz, vídeo, imágenes y datos e incluso podría estar relacionada con la red de televisión local por cable. Este estándar define un protocolo de gran velocidad, en donde las computadoras conectadas comparten un bus doble de fibra óptica utilizando el método de acceso llamado bus de cola distribuido. Una red que se extiende al rango de 75 kilómetros, opera a velocidades de uno a 200 Mbits/s y proporciona un grupo de servicios integrados para datos en tiempo real, voz, vídeo y transmisión de imágenes.

Sistema de información (**SI**: Information system). Es un conjunto de normas y procedimientos muy bien establecidos, referidos a las operaciones de información de una organización de la empresa o institución.

BIBLIOGRAFÍA

[1] ARAUJO Lourdes y CERVIGÓN Carlos.
Algoritmos Evolutivos: un enfoque práctico. México: Alfaomega Rama Grupo Editor SA; 2009.

[2] ARROYO APAZA Víctor Manuel.
Modelo de un Algoritmo Genético con Selección Discriminatoria de Individuos bajo un Esquema de Ponderación de Probabilidades de Mutación. Brasil: Universidad católica San Pablo; 2013.

[3] BARÁN B, KASZKUREWICZ E, BHAYA A.
"Parallel Asinchronous Team Algorithms: Convergence and Performance Analysis. IEEE Transactions on Parallel and Distributed Systems". 7(7), 1983.

[4] BARÁN Benjamin, CHAPARRO Enrique
"Algoritmos Genéticos Asíncronos Combinados Para una Red Hetereogénea de Computadoras". Paraguay: Centro Nacional de Computación de la Universidad Nacional de Asunción, 1999.

[5] CANTÚ-PAZ Erick.
"A Summary of Research on Parallel Genetic Algorithms". Technical Report Nº 95007, July 1995.

[6] DAVIS L.
"Manual de Algoritmos Genéticos". Nueva York: Van Nostrand Reinhold, 1991.

[7] DE JONG A.
"An Analysis of the Behavior of a Class of Genetic Adaptive Systems". Estados Unidos: [Doctoral Thesis], University of Michigan, 1975.

[8] FERNÁNDEZ ALDANA Luís Antonio.
"Transmisión y Comunicación de datos". México: BUAP, 2005.

[9] GARCIA TOMÁS Jesús.
"Redes Para Proceso Distribuido". México, D. F.: Editorial Rama, 1997.

[10] GOLDBERG D. E.
"Algoritmos Genéticos en la Búsqueda Para Optimización y Aprendizaje de la Máquina". Estados Unidos-Massachusetts: Addison-Wesley, 1989.

[11] HALSALL Fred.
"Comunicación de Datos. Redes de Computadoras y Sistemas Abiertos". México: Prentice Hall, 2001.

[12] HOLLAND John. H.
"Compendio del Algoritmos Genéticos". Estados Unidos: Universidad de Michigan, 7(22), 1975.

[13] HUIDOBRO MOYA José Manuel, MILLÁN TEJEDOR Ramón J, ROLDÁN MARTÍNEZ
"Tecnologías de Telecomunicaciones". México: Alfaomega Grupo Editor SA, 2006.

[14] IEEE Communications Magazine.
"Generalized Multiprotocol Label Switching: An Overview of Routing and Managment Enhancements". IEEE Communications Magazine, Jan 2001.

[15] MÜHLENBEIN H., SCHOMISCH, M., BORN J.
"The Parallel Genetic Algorithm as Function Optimizer". Proceeding of the Fourth International Conference on Genetic Algorithm, 271-278, 1991.

[16] MUÑOZ PÉREZ Miguel Ángel.
"Funcionamiento de un Algoritmos genéticos". Abril, 2005.

[17] SCHAFFER J.
"Procedimientos de III Conferencia Internacional de los Algoritmos Genéticos". San Mateo: Publicadores de Morgan Kaufmann, CA, 1989.

[18] SERVATI Al, BREMNER Lynn, LASI Anthony.
"La Biblia de Intranet". México, D. F.: Editorial McGraw –Hill, 1998.

[19] SWANY Martín.
"Red Internet 2". Estados Unidos: Universidad de Delaware, 2007.
http://www.internet2.edu/networkresearch/

[20] WHITLEY D.
"El algoritmo de genitor y presión de la selección: por qué la línea de asignación basado de ensayos reproductores son mejores". En los procedimientos de la III Conferencia Internacional de los Algoritmos Genéticos. San Mateo: P. 116-121, 1989.

ANEXOS

ANEXO 1: CÓDIGO FUENTE DEL SIMULADOR

```
1   /* ga.cpp
2   * Ejecucion principal del algoritmo genetico.
3   *
4   */
5
6   #include <vcl.h>
7   #include <math.h>
8   #pragma hdrstop
9   #include "mainsimulador.h"
10
11  #include <time.h>
12  #include "Poblacion.h"
13
14  extern Tfrmmain *frmmain;
15
16  int *nodosGeneticos;
17
18  //-- Aca verifico si paso el tiempo solicitado en la simulacion
19  bool ExcedeTiempoTranscurrido(void)
20  {
21
22   time(&(frmmain->FinalTimeSimulador));
23   frmmain->ContadorDeSegundos=difftime(frmmain->FinalTimeSimulador,frmmain-
     >StartTimeSimulador);
24   return(frmmain->ContadorDeSegundos > frmmain->TiempoSolicitado);
25  }
26
27  //-- En base a la cantidad especificada de los procesos por atender.
28  //-- Se debe redistribuir entre el total de tiempo de la simulacion.
29  //-- El proceso de activacion del evento es cuando el valor aleatorio
30  //-- cae dentro del contador de segundos que actualmente evoluciona
31  bool SeGeneraNuevoRequerimiento(void)
32   {
33     Randomize();
34     return(frmmain->TiempoSolicitado>=Random(frmmain->TiempoSolicitado*4));
35   }
36
37  //-- Selecciono los procesos entrantes, se asignan a la cola. Si la cola esta
38  //-- esta llena , se avanza al siguiente nodo. Los procesos atendidos aumentan
39  //-- y los procesos no atendidos disminuyen
40  bool NuevoProcesoEntrante(void)
41   {
42       int ColaXAsignar=0;
43       if(frmmain->MaxProcesosXAtender>frmmain->ProcesosAtendidos)
44        {
45         if(SeGeneraNuevoRequerimiento())
46          {
47                  for(int i = 2;i<frmmain->DGridResultados->RowCount;i++)
48                   {
```

```
49                  ColaXAsignar=  frmmain->DGridResultados->Cells[1][i].ToInt() - frmmain->DGridResultados->Cells[2][i].ToInt();
50                  if(ColaXAsignar>0)
51                        {
52   //-- Si los procesos que aun estan por asignar:MaxProcesosXAtender-ProcesosAtendidos,
53   //-- es mayor que la cola del nodo, entra aca y significa que aun hay procesos

54  //-- por atender
55                        if(frmmain->MaxProcesosXAtender- frmmain->ProcesosAtendidos - ColaXAsignar>=0)
56                         {
57                          frmmain->ProcesosAtendidos+=ColaXAsignar;
58                          frmmain->DGridResultados->Cells[2][i]=AnsiString(ColaXAsignar);
59                         }
60                        else
61                         {
62                          frmmain->DGridResultados->Cells[2][i]=AnsiString(frmmain->MaxProcesosXAtender- frmmain->ProcesosAtendidos);
63                          frmmain->ProcesosAtendidos+=(frmmain->MaxProcesosXAtender-frmmain->ProcesosAtendidos);
64                          break;
65                         }
66                        // Ya no existen mas procesos pendientes
67                        if(frmmain->MaxProcesosXAtender- frmmain->ProcesosAtendidos<=0)
68                         break;
69                       }
70                }
71               return(true);
72         }
73       }
74      return(false);
75   }
76
77  //-- Proceso de aplicacion del algoritmo Genetico: 111111
78  int OptimizarConAlgoritmoGenetico(int TamanoSubPoblacion,char *strOptimoNodo)
79   {
80    //  Inicializa la variable o semilla
81    Poblacion * poblacion = new Poblacion(TamanoSubPoblacion);
82
83
84    frmmain->StartTimeAlgoritmo=time(NULL);
85
86    // 1. Inicializacion. Se asigna la poblacion al procesos de optimizacion
87    for (int i = 0; i < poblacion->maximumSize_ ; i++)
88     {
89       Individual * individual = new Individual(frmmain->LONGITUD_CROMOSOMA,nodosGeneticos[i],i) ;
90       Randomize();
91       individual->fitness_ = Random() ;
92       individual->nodo_=i;
93       poblacion->addIndividual(individual) ;
94     }
95
96    // 2. Loop principal del algoritmo genetico
97    for (int i = 0; i < frmmain->NUMERO_DE_ITERACIONES; i++)
98     {
99       Individual  parent1 ;
100      Individual  parent2 ;
```

```
101      Individual  *child   ;
102
103      // 2.1. Selection
104      parent1 = poblacion->binaryTournamentSelection() ;
105      parent2 = poblacion->binaryTournamentSelection() ;
106
107      // 2.2. Cruzamiento
108      child = poblacion->crossover(frmmain->PROBABILIDAD_CRUZAMIENTO, parent1,
     parent2) ;
109
110      // 3.3. Mutacion
111      child->swapMutation(frmmain->PROBABILIDAD_MUTACION) ;
112
113      // 3.4 Evaluacion
114      Randomize();
115      child->fitness_ = Random() ;
116
117      // 3.5. Reemplazo
118      poblacion->replacement(child) ;
119    }
120   poblacion->sort() ;
121   Sleep(100);  // Se asume la cantidad:tiempo de procesamiento en milisegundos
122   frmmain->FinalTimeAlgoritmo=time(NULL);
123   //-- Algoritmo genetico:
124   int nodo=0;
125   strcpy(strOptimoNodo,poblacion->CromosomaMax(&nodo));
126   frmmain->TiempoConsumido=difftime(frmmain->FinalTimeAlgoritmo, frmmain-
     >StartTimeAlgoritmo);
127   delete poblacion;
128   return(nodo);
129 }
130
131
132 //-- Envio la data a la pantalla de reporte
133
134 void EnvioDatosAlReporte(AnsiString Texto)
135
136  {
137
138   AnsiString TextoLineas=frmmain->txtReporteSimulador->Lines->Text;
139
140   TextoLineas=TextoLineas+Texto;
141
142   frmmain->txtReporteSimulador->Lines->Text=(TextoLineas+"\r\n");
143
144  }
145
146 //-- Envio la data al text box
147 void EnvioDatosAlReporteOptimos(AnsiString Texto)
148
149  {
150
151   AnsiString TextoLineas=frmmain->txtReporteOptimos->Lines->Text;
152
153   TextoLineas=TextoLineas+Texto;
154
155   frmmain->txtReporteOptimos->Lines->Text=(TextoLineas+"\r\n");
156
157  }
```

```
158
159 //-- Hilo del proceso de simulacion
160 DWORD EjecucionDeSimulacion(LPDWORD lparam)
161  {
162   int Cromosoma=(int) pow(2,(double)frmmain->LONGITUD_CROMOSOMA);//-- La cadena se
    inicia : 111111  =63
163   while(!ExcedeTiempoTranscurrido() && frmmain->ProcesoEjecutandose)
164   {
165    //-- Simulador:
166    frmmain->msgCronometro->Caption=AnsiString(frmmain->ContadorDeSegundos);
167    frmmain->msgTiempoMaximo->Caption=AnsiString(frmmain->TiempoSolicitado);
168    frmmain->msgProcesosTotales->Caption=AnsiString(frmmain->MaxProcesosXAtender);
169    frmmain->msgProcesosPorAsignar->Caption=AnsiString(frmmain->MaxProcesosXAtender-
    frmmain->ProcesosAtendidos);
170
171    NuevoProcesoEntrante();
172    //-- Asigno los valores binarios para el calculo genetico
173    for(int iParametro = 2;iParametro<frmmain->DGridResultados->RowCount;iParametro++)
174      {
175        if(frmmain->DGridResultados->Cells[2][iParametro].ToInt()>0)
176         {
177                  EnvioDatosAlReporte("Proceso Entrante(Identificador):"+frmmain-
    >DGridResultados->Cells[0][iParametro]);
178                  nodosGeneticos=new int[frmmain->DGridResultados->Cells[2][iParametro].ToInt()
    + 1];
179                  for(int i = 0;i<frmmain->DGridResultados->Cells[2][iParametro].ToInt();i++)
180                   {
181                    Randomize();
182                    nodosGeneticos[i]=Random(Cromosoma);
183                    EnvioDatosAlReporte("Cromosomas
    inicial("+AnsiString(i)+"):"+AnsiString(nodosGeneticos[i]));
184                   }
185                  //-- Aca se ejecuta el proceso de simulacion
186                  char strOptimo[50];
187                  int nodo= OptimizarConAlgoritmoGenetico(frmmain->DGridResultados-
    >Cells[2][iParametro].ToInt(),strOptimo);
188                  double TiempoConsumidoNodo=frmmain->DGridResultados-
    >Cells[4][iParametro].ToDouble();
189
190                  delete [] nodosGeneticos;
191
192                  //-- Asigno el nodo al array de procesos ejecutandose
193                  EnvioDatosAlReporte("Cromosoma óptimo:"+AnsiString(strOptimo));
194                  EnvioDatosAlReporte("Nodo seleccionado:"+AnsiString(nodo));
195                  EnvioDatosAlReporteOptimos("Nodo óptimo:"+AnsiString(nodo));
196
197                  int ActualEnCola, ActualAtendidos;
198                  // Aca debo restar los procesos que voy seleccionando
199                  ActualEnCola=frmmain->DGridResultados->Cells[2][iParametro].ToInt() - 1;
200                  frmmain->DGridResultados->Cells[2][iParametro]= AnsiString(ActualEnCola);
201                  ActualAtendidos=frmmain->DGridResultados->Cells[3][iParametro].ToInt() + 1;
202                  frmmain->DGridResultados->Cells[3][iParametro]= AnsiString(ActualAtendidos);
203                  //-- Acumulo el tiempo consumido para el nodo correspondiente
204                  TiempoConsumidoNodo+=frmmain->TiempoConsumido;
205                  //frmmain->DGridResultados->Cells[4][iParametro]=
    AnsiString(TiempoConsumidoNodo);
206
207                  frmmain->DGridResultados->Cells[4][iParametro]= AnsiString(frmmain-
    >DGridResultados->Cells[4][iParametro].ToDouble()+TiempoConsumidoNodo);
```

```
208
209                 //-- El nodo que recibe el proceso.
210                 frmmain->msgNodoAsignado->Caption=AnsiString(iParametro);
211         }
212         Sleep(50);
213       }
214   }
215   frmmain->ProcesoEjecutandose=false;
216   Application->MessageBox(L"Ha concluido el proceso de simulacion",L"Mensaje del
      sistema",MB_OK);
217   return(0);
218  }



// Individuo.cpp
// Representa un individuo
//
#include "Individuo.h"
#include "mainsimulador.h"

//-- Retorna un valor aleatorio entre mix y max
int randInteger(int min, int max) {
  randomize();
  double random =Random() ;
  random = random * (max - min) + min ;
  return (int) random ;
}

//-- Compara 2 individuos. Retorna TRUE si el fitness del primer individuo
//-- es superior o igual que el  segundo. FALSE en caso contrario
bool compareIndividuals(Individual  individual1, Individual  individual2) {
  return individual1.fitness_ >= individual2.fitness_ ;
}

//-- Constructor de permutacion de enteros
Individual::Individual(int length,int nodoGenetico,int nodo)
{
  chromosomeLength_ = length ;
  chromosome_ = new int[chromosomeLength_ + 1] ;
  int * randNum = new int[chromosomeLength_ + 1];

  for(int i = 0; i < chromosomeLength_; i++) {
          int num  ;
          num           = randInteger(0,nodoGenetico) ;
          randNum[i]    = num ;
          chromosome_[i] = i   ;
  }

  for(int i = 0; i < chromosomeLength_-1; i++)
          for(int j = i+1; j < chromosomeLength_; j++)
            if(randNum[i] > randNum[j]) {
                    int temp ;
                    temp          = randNum[i]    ;
                    randNum[i]    = randNum[j]    ;
                    randNum[j]    = temp          ;
                    temp          = chromosome_[i] ;
                    chromosome_[i] = chromosome_[j] ;
                    chromosome_[j] = temp          ;
```

```
            } // if
  delete [] randNum;
  fitness_ = 0 ;
  nodo_=nodo;
}

//-- Constructor copia, genera un nuevo individuo de otro
Individual::Individual(Individual * individual) {
  chromosomeLength_ = individual->chromosomeLength_;
  chromosome_ = new int[chromosomeLength_ + 1] ;

  for (int i = 0; i < chromosomeLength_; i++)
          chromosome_[i] = individual->chromosome_[i] ;
  fitness_ = individual->fitness_ ;
  nodo_ = individual->nodo_ ;
}

//-- Constructor vacio
Individual::Individual() {

  chromosome_ = new int[20] ;
  fitness_ =0;
  nodo_=0;
}

//-- Destructor
Individual::~Individual()
{
  delete []chromosome_ ;
}

//-- Retorna el Fitness de un individuo
double Individual::getFitness()
{
  return fitness_ ;
}

//-- Establece el fitness de un individuo
void Individual::setFitness(double fitness)
{
  fitness_ = fitness ;
}

//-- intercambia 2 posiciones seleccionada de manera aleatoria
void Individual::swapMutation(double probability)
{
  double random ;
  Randomize();
  random = Random() ;
  if (random < probability)
  {
          int pos1 ;
          int pos2 ;

          pos1 = randInteger(0, chromosomeLength_-1);
          pos2 = randInteger(0, chromosomeLength_-1);
          //while(pos1 == pos2)
            if (pos1 == (chromosomeLength_ - 1))
                  pos2 = randInteger(0, chromosomeLength_- 2);
```

```
            else
                    pos2 = randInteger(pos1, chromosomeLength_- 1);

            // Intercambia
            int temp = this->chromosome_[pos1];
            this->chromosome_[pos1] = this->chromosome_[pos2];
            this->chromosome_[pos2] = temp;
  }
}

//-- Constructor copia de un individuo
Individual & Individual::operator=(const Individual& individual) {
  chromosomeLength_ = individual.chromosomeLength_ ;
  for (int i = 0; i < chromosomeLength_; i++)
            chromosome_[i] = individual.chromosome_[i] ;
  fitness_ = individual.fitness_ ;
  nodo_ = individual.nodo_ ;
  return *this ;
}

//-- Compara 2 individuos
bool Individual::operator==(const Individual& individual) {
  for (int i = 0; i < chromosomeLength_; i++)
            if (chromosome_[i] != individual.chromosome_[i])
              return false ;
  return true ;
}

//-- Operador menor que .Compara si el Fitness del individuo es menor que el Fitness
bool Individual::operator<(const Individual& individual) {
  if (fitness_ < individual.fitness_)
            return true ;
  else
            return false ;
}




// $$---- Entrada principal del sistema de Simulacion ----
// mainsimulador.cpp
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "Splash.h"
#include "mainsimulador.h"
#include "ga.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma link "RzEdit"
#pragma link "RzPanel"
#pragma link "RzShellDialogs"
#pragma link "RzDBEdit"
#pragma resource "*.dfm"
Tfrmmain *frmmain;

//---------------------------------------------------------------------------
```

```
__fastcall Tfrmmain::Tfrmmain(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------

void __fastcall Tfrmmain::BtnSalirClick(TObject *Sender)
{
 FrmSplash->Close();
 frmmain->Close();
}

//---------------------------------------------------------------------------

void __fastcall Tfrmmain::FormActivate(TObject *Sender)
{
  int cnt=0;
  AnsiString Titulos[]={"Server","Tamaño de cola","En Cola","Atendidos","Tiempo Consumido"};
  for(cnt=0;cnt<DGridProcesadores->ColCount;cnt++)
   {
        DGridProcesadores->Cells[cnt][0]=Titulos[cnt];
   }
  for(int cnt=0;cnt<DGridResultados->ColCount;cnt++)
   {
        DGridResultados->Cells[cnt][0]=Titulos[cnt];
   }
 }
//---------------------------------------------------------------------------

void __fastcall Tfrmmain::BtnAdicionarClick(TObject *Sender)
{
 if(atoi(txtCfgNroProcesosCola->Text.c_str())>0)
  {
   DGridProcesadores->Cells[0][DGridProcesadores-
>RowCount]="Nodo"+AnsiString(DGridProcesadores->RowCount-1);
   DGridProcesadores->Cells[1][DGridProcesadores->RowCount]=txtCfgNroProcesosCola->Text;
   DGridProcesadores->RowCount++;

   DGridResultados->Cells[0][DGridResultados->RowCount]="Nodo"+AnsiString(DGridResultados-
>RowCount-1);
   DGridResultados->Cells[1][DGridResultados->RowCount]=txtCfgNroProcesosCola->Text;
   DGridResultados->Cells[2][DGridResultados->RowCount]="0";
   DGridResultados->Cells[3][DGridResultados->RowCount]="0";
   DGridResultados->Cells[4][DGridResultados->RowCount]="0";
   DGridResultados->RowCount++;

   BtnEliminar->Enabled=true;
  }
 else
  Application->MessageBox(L"Debe ingresar valores mayores a cero", L"Error", MB_OK);
}
//---------------------------------------------------------------------------

void __fastcall Tfrmmain::BtnEliminarClick(TObject *Sender)
{
 if(DGridProcesadores->RowCount>2)
 {
  if(DGridProcesadores->Row>1)
   {
```

```
            for(int i = DGridProcesadores->Row;i<DGridProcesadores->RowCount;i++)
             {
              DGridProcesadores->Rows[i]->Assign(DGridProcesadores->Rows[i+1]);
              DGridResultados->Rows[i]->Assign(DGridResultados->Rows[i+1]);
             }
            DGridProcesadores->RowCount--;
            DGridResultados->RowCount--;
   }
 }
 else
  BtnEliminar->Enabled=false;
}
//---------------------------------------------------------------------------

//Asigno la linea de la grilla
void AsignoLaEstructuraAGrilla(CHAR *StrUserText)
{
 int cnt=1;
 CHAR *token, seps[] =",";
 token = strtok( StrUserText, seps );
 while( token != NULL )
 {
  frmmain->DGridProcesadores->Cells[cnt][frmmain->DGridProcesadores->RowCount]=token;
  frmmain->DGridResultados->Cells[cnt][frmmain->DGridResultados->RowCount]=token;
  frmmain->DGridResultados->Cells[cnt+1][frmmain->DGridResultados->RowCount]="0";
  frmmain->DGridResultados->Cells[cnt+2][frmmain->DGridResultados->RowCount]="0";
  frmmain->DGridResultados->Cells[cnt+3][frmmain->DGridResultados->RowCount]="0";
  cnt++;
  token = strtok( NULL, seps );
 }
}

//-- Cargo la configuracion
bool LeoParametrosCfg(CHAR *StrUserText)
{
 bool crcInicio=false, crcFin=false, crcInicioArray=false;
 AnsiString NodoNombre="", CadenaArray[100];
 int Nodos=0;
 CHAR *token, *PtrArranque;
 CHAR seps[]   = "\r\n";
 token = strtok( StrUserText, seps );
 while( token != NULL )
 {
  if(strstr(token,"[PARAMETROSINICIO]")!=NULL)
   crcInicio=true;

  if(crcInicio)
   {
            if(strstr(token,"TiempoSimulacion")!=NULL)
             {
              PtrArranque=strchr(token,'=');
              PtrArranque++;
              frmmain->txtTiempoSimulacion->Text=PtrArranque;
             }
            if(strstr(token,"NroProcesos")!=NULL)
             {
              PtrArranque=strchr(token,'=');
              PtrArranque++;
              frmmain->txtProcesosPorAtender->Text=PtrArranque;
```

```
            }
           if(strstr(token,"ProbCruzamiento")!=NULL)
            {
             PtrArranque=strchr(token,'=');
             PtrArranque++;
             frmmain->txtIniProbCruzamiento->Text=PtrArranque;
            }
           if(strstr(token,"ProbMutacion")!=NULL)
            {
             PtrArranque=strchr(token,'=');
             PtrArranque++;
             frmmain->txtIniProbMutacion->Text=PtrArranque;
            }
           if(strstr(token,"NroIteraciones")!=NULL)
            {
             PtrArranque=strchr(token,'=');
             PtrArranque++;
             frmmain->txtIniNroIteraciones->Text=PtrArranque;
            }
           if(strstr(token,"NroAlelos")!=NULL)
            {
             PtrArranque=strchr(token,'=');
             PtrArranque++;
             frmmain->txtIniNroAlelos->Text=PtrArranque;
            }
           if(strstr(token,"[PARAMETROSFIN]")!=NULL)
             crcFin=true;
  }
 if(crcInicio && crcFin)
  {
          if(strstr(token,"[HOSTSINI]")!=NULL)
           crcInicioArray=true;

           if(crcInicioArray)
            {
             NodoNombre="H"+AnsiString(Nodos+1);
             if(strstr(token,NodoNombre.c_str())!=NULL)
                    {
                     PtrArranque=strchr(token,'=');
                     PtrArranque++;
                     CadenaArray[Nodos] =PtrArranque;
                     Nodos++;
                    }
             if(strstr(token,"[HOSTSFIN]")!=NULL)
                     break;
            }
  }
 token = strtok( NULL, seps );
}

 //Comienzo a asignar las lineas del array para los nodos
 frmmain->DGridProcesadores->RowCount=2;
 frmmain->DGridResultados->RowCount=2;
 for(int cnt=0;cnt<Nodos;cnt++)
  {
          frmmain->DGridProcesadores->Cells[0][frmmain->DGridProcesadores-
>RowCount]="Nodo"+AnsiString(cnt+1);
          frmmain->DGridResultados->Cells[0][frmmain->DGridProcesadores-
>RowCount]="Nodo"+AnsiString(cnt+1);
```

```
            AsignoLaEstructuraAGrilla(CadenaArray[cnt].c_str());
            frmmain->DGridProcesadores->RowCount++;
            frmmain->DGridResultados->RowCount++;
            frmmain->BtnEliminar->Enabled=true;
  }
 return(true);
}

void __fastcall Tfrmmain::BtnCargarClick(TObject *Sender)
{
 DWORD cbFile=0, cbRead=0;
 CHAR *StrUserText;
 bool crc=false;
 OpenFileSim->Execute();
 HANDLE hHandle = CreateFile(OpenFileSim-
>FileName.c_str(),GENERIC_READ,FILE_SHARE_READ,NULL,OPEN_EXISTING,0,NULL );
 if ( hHandle != INVALID_HANDLE_VALUE )
 {
  cbFile = GetFileSize( hHandle, NULL );
  if ( cbFile != (DWORD) -1 )
  {
            StrUserText = (CHAR *)LocalAlloc( LPTR, cbFile + 1 );
            if (StrUserText)
            {
             ReadFile( hHandle,StrUserText,cbFile,&cbRead,NULL );
             StrUserText[cbRead]='\0';
             crc=LeoParametrosCfg(StrUserText);
             LocalFree(StrUserText);
             if(!crc)
              Application->MessageBox(L"No se puede leer el archivo de configuracion", L"Error",
MB_OK);
            }
  }
  CloseHandle(hHandle);
 }
}
//---------------------------------------------------------------------------

void __fastcall Tfrmmain::BtnGuardarClick(TObject *Sender)
{

  SaveFileSim->Filter="*.sim";
  SaveFileSim->Execute();
  HANDLE hFile = CreateFile(SaveFileSim->FileName.c_str(),
                                        GENERIC_WRITE, FILE_SHARE_READ,
                                         NULL, CREATE_ALWAYS,
                                         FILE_ATTRIBUTE_NORMAL, NULL);
 if(hFile == INVALID_HANDLE_VALUE)
   Application->MessageBox(L"No puede crear archivo de configuracion",L"Error",MB_OK);
 else
 {
  DWORD dwAmt;
  AnsiString Datos="[PARAMETROSINICIO]\r\n";
  Datos+=("TiempoSimulacion="+txtTiempoSimulacion->Text+"\r\n");
  Datos+=("NroProcesos="+txtProcesosPorAtender->Text+"\r\n");
  Datos+=("ProbCruzamiento="+txtIniProbCruzamiento->Text+"\r\n");
  Datos+=("ProbMutacion="+txtIniProbMutacion->Text+"\r\n");
  Datos+=("NroIteraciones="+txtIniNroIteraciones->Text+"\r\n");
  Datos+=("NroAlelos="+txtIniNroAlelos->Text+"\r\n");
```

```
   Datos+=("[PARAMETROSFIN]\r\n");
   Datos+=("[HOSTSINI]\r\n");
   if(DGridProcesadores->RowCount>1)
           {
            for(int i = 2;i<DGridProcesadores->RowCount;i++)
             {
             Datos+=("H"+AnsiString(i-1)+"="+DGridProcesadores->Cells[1][i]+","+\
                                DGridProcesadores->Cells[2][i]+"\r\n");
             }
           }
   Datos+=("[HOSTFIN]\r\n");
   if(!WriteFile(hFile,Datos.c_str(),Datos.Length(), &dwAmt, NULL))
           {
            CloseHandle(hFile);
            Application->MessageBox(L"No se puede escribir archivo de
configuracion",L"Error",MB_OK);
           }
   else
            CloseHandle(hFile);
  }
}

//---------------------------------------------------------------------------

//-- Envio la data al text box
void EnvioDatosAlReporte(AnsiString Texto)

 {

  AnsiString TextoLineas=frmmain->txtReporteSimulador->Lines->Text;

  TextoLineas=TextoLineas+Texto;

  frmmain->txtReporteSimulador->Lines->Text=(TextoLineas+"\r\n");

 }

//-- Envio la data al text box
void EnvioDatosAlReporteOptimos(AnsiString Texto)

 {

  AnsiString TextoLineas=frmmain->txtReporteOptimos->Lines->Text;

  TextoLineas=TextoLineas+Texto;

  frmmain->txtReporteOptimos->Lines->Text=(TextoLineas+"\r\n");

 }

//-- Entrada para ejecutar el proceso
void __fastcall Tfrmmain::BtnIniciarProcesoClick(TObject *Sender)
{
 if( DGridProcesadores->RowCount-2>0)
  {
   if(!frmmain->ProcesoEjecutandose)
           {
            frmmain->StartTimeSimulador=0;
            frmmain->FinalTimeSimulador=0;
```

```
        frmmain->StartTimeAlgoritmo=0;
        frmmain->FinalTimeAlgoritmo=0;
        frmmain->ContadorDeSegundos=0;
        frmmain->TiempoSolicitado=0;
        frmmain->MaxProcesosXAtender=0;
        frmmain->ProcesosAtendidos=0;
        frmmain->txtReporteSimulador->Text="";

        //-- Inicializo los valores globales del algoritmo genetico
        frmmain->PROBABILIDAD_CRUZAMIENTO =frmmain->txtIniProbCruzamiento-
>Text.ToDouble(); // 0.9
        frmmain->PROBABILIDAD_MUTACION  =frmmain->txtIniProbMutacion-
>Text.ToDouble(); //0.1
        frmmain->NUMERO_DE_ITERACIONES  =frmmain->txtIniNroIteraciones->Text.ToInt();
        frmmain->LONGITUD_CROMOSOMA    =frmmain->txtIniNroAlelos->Text.ToInt();
        frmmain->TAMANO_POBLACION      =frmmain->txtProcesosPorAtender->Text.ToInt();

        //--              ******                  -----

        EnvioDatosAlReporte("Fecha:"+Now());

        EnvioDatosAlReporte("Probabilidad de cruzamiento:"+frmmain->txtIniProbCruzamiento-
>Text);

        EnvioDatosAlReporte("Probabilidad de mutacion:"+frmmain->txtIniProbMutacion->Text);

        EnvioDatosAlReporte("Numero de iteraciones:"+frmmain->txtIniNroIteraciones->Text);

        EnvioDatosAlReporte("Longitud del cromosoma:"+frmmain->txtIniNroAlelos->Text);


        EnvioDatosAlReporteOptimos("Fecha:"+Now());

        EnvioDatosAlReporteOptimos("Probabilidad de cruzamiento:"+frmmain-
>txtIniProbCruzamiento->Text);

        EnvioDatosAlReporteOptimos("Probabilidad de mutacion:"+frmmain->txtIniProbMutacion-
>Text);

        EnvioDatosAlReporteOptimos("Numero de iteraciones:"+frmmain->txtIniNroIteraciones-
>Text);

        EnvioDatosAlReporteOptimos("Longitud del cromosoma:"+frmmain->txtIniNroAlelos-
>Text);


        frmmain->ProcesoEjecutandose=true;
        frmmain->TiempoSolicitado=txtTiempoSimulacion->Text.ToDouble();
        frmmain->MaxProcesosXAtender=txtProcesosPorAtender->Text.ToInt();

        DWORD dwThreadId, dwThrdParam = 1;
        time(&(frmmain->StartTimeSimulador));
        frmmain-
>hThread=CreateThread(NULL,0,(LPTHREAD_START_ROUTINE)EjecucionDeSimulacion,
                                                        &dwThrdParam,0,&dwThreadId);
      }
  else
      Application->MessageBox(L"El proceso de simulacion esta corriendo",L"Error",MB_OK);
 }
```

```
 else
          Application->MessageBox(L"No tiene nodos participantes",L"Error",MB_OK);
}
//---------------------------------------------------------------------------
//-- Detiene el proceso de simulacion
void __fastcall Tfrmmain::BtnDetenerProcesoClick(TObject *Sender)
{
          frmmain->ProcesoEjecutandose=false;
}
//---------------------------------------------------------------------------


void __fastcall Tfrmmain::BtnSalirOpcionalClick(TObject *Sender)
{
 frmmain->Close();
}
//---------------------------------------------------------------------------

//-- Guarda el reporte en el disco
void __fastcall Tfrmmain::BtnGuardarReporteClick(TObject *Sender)
{
  SaveFileSim->Filter="*.txt";
  SaveFileSim->Execute();
  HANDLE hFile = CreateFile(SaveFileSim->FileName.c_str(),
                                        GENERIC_WRITE, FILE_SHARE_READ,
                                         NULL, CREATE_ALWAYS,
                                         FILE_ATTRIBUTE_NORMAL, NULL);
 if(hFile == INVALID_HANDLE_VALUE)
   Application->MessageBox(L"No puede crear archivo",L"Error",MB_OK);
 else
  {
   DWORD dwAmt;
   if(!WriteFile(hFile,txtReporteSimulador->Lines->Text.c_str(),txtReporteSimulador->Lines-
>Text.Length(), &dwAmt, NULL))
          {
           CloseHandle(hFile);
           Application->MessageBox(L"No se puede escribir archivo",L"Error",MB_OK);
          }
   else
           CloseHandle(hFile);
  }
}
//---------------------------------------------------------------------------


void __fastcall Tfrmmain::FormClose(TObject *Sender, TCloseAction &Action)
{
 FrmSplash->Close();
 frmmain->Close();
}
//---------------------------------------------------------------------------
//
// Poblacion.cpp
// Representa una poblacion
//
#include "Poblacion.h"
#include "mainsimulador.h"

//-- Constructor del objeto poblacion
```

```
Poblacion::Poblacion(int maximumSize) {
  maximumSize_ = maximumSize ;
  size_        = 0           ;
  Arrindividual_= new Individual[maximumSize_+1] ;
}

//-- Destruye el objeto poblacion
Poblacion::~Poblacion()
{
 delete [] Arrindividual_;
}

//-- Adiciona un individuo al conjunto
void Poblacion::addIndividual(Individual * individual)
{
  if (size_ == maximumSize_)
  {
          exit(-1) ;
  }
  else {
          Arrindividual_[size_]=*individual;
          size_ ++ ;
  }
}

//--- Elimina el ultimo individuo
void Poblacion::deleteLastIndividual() {
  if (size_ > 0) {
          size_ -- ;
  }
  else {
          exit(-1) ;
  }
}


//-- Sortea la poblacion
void Poblacion::sort()
{
  //std::stable_sort(individual_.begin(),
  //                                 individual_.end(),
  //                                 compareIndividuals);
}

//-- Cruzamiento
Individual *Poblacion::crossover(double       probability,
                                                        Individual  ind1,
                                                        Individual  ind2) {

  Individual * newIndividual     ;
  double       random            ;
  Randomize();
  random = Random() ;
  if (random < probability) {
          newIndividual = new Individual(&ind1) ;
  }
  else
          newIndividual = new Individual(&ind2) ;
  return newIndividual ;
}
```

```
//-- Seleccion binaria, competencia
Individual  Poblacion::binaryTournamentSelection() {
  int          random      ;
  Individual  individual1 ;
  Individual  individual2 ;

  random = randInteger(0, size_ - 1) ;
  individual1 = Arrindividual_[random] ;
  random = randInteger(0, size_- 1) ;
  individual2 = Arrindividual_[random] ;

  if (individual1.fitness_ > individual2.fitness_)
          return individual1 ;
  else
          return individual2 ;
}

//-- Sortea y reemplaza el peor individuo
void Poblacion::replacement(Individual  *individual) {
  // Paso 1: sortear la Poblacion
  this->sort() ;

  // Paso 2: Reemplazar el peor individuo
  if (compareIndividuals(individual,
                                                &Arrindividual_[size_ - 1]))
          Arrindividual_[size_ - 1] = *individual ;

  delete individual ;
}

//-- Retorna el cromosoma maximo
char *Poblacion::CromosomaMax(int *nodo) {
 char *rtnStr= new char[Arrindividual_[size_ -1].chromosomeLength_  + 1];
 for (int j = 0; j < Arrindividual_[size_ -1].chromosomeLength_ ; j++) {
            rtnStr[j]='0'+Arrindividual_[size_ -1].chromosome_[j];
          }
 *nodo=Arrindividual_[size_ -1].nodo_;
 return(rtnStr);
}

//-- Retorna el cromosoma maximo
char *Poblacion::CromosomaMin(int *nodo) {
 char *rtnStr= new char[Arrindividual_[0].chromosomeLength_ + 1];
 for (int j = 0; j < Arrindividual_[0].chromosomeLength_ ; j++) {
            rtnStr[j]='0'+Arrindividual_[0].chromosome_[j];
          }
 *nodo=Arrindividual_[size_ -1].nodo_;
 return(rtnStr);
}



/* Problema.cpp
 * Representa un problema a resolver
 */

#include <Problema.h>
```

```
Problema::Problema(void) {
}

void Problema::evaluate(Individual * individual) {
  individual->fitness_ = rand() ;
} // Problema::evalua



//$$---- Form CPP ----
// Salir.cpp
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "Salir.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm3 *Form3;
//---------------------------------------------------------------------------
__fastcall TForm3::TForm3(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------



//$$---- EXE CPP ----
// simulador.cpp
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop
//---------------------------------------------------------------------------
USEFORM("mainsimulador.cpp", frmmain);
USEFORM("Salir.cpp", Form3);
USEFORM("Splash.cpp", FrmSplash);
//---------------------------------------------------------------------------
WINAPI WinMain(HINSTANCE, HINSTANCE, LPSTR, int)
{
        try
        {
                Application->Initialize();
                Application->CreateForm(__classid(TFrmSplash), &FrmSplash);
                Application->CreateForm(__classid(Tfrmmain), &frmmain);
                Application->CreateForm(__classid(TForm3), &Form3);
                Application->Run();
        }
        catch (Exception &exception)
        {
                Application->ShowException(&exception);
        }
        catch (...)
        {
                try
                {
```

```
                throw Exception("");
            }
            catch (Exception &exception)
            {
                Application->ShowException(&exception);
            }
        }
        return 0;
}
//---------------------------------------------------------------------------



//$$---- Form CPP ----
// Splash.cpp
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "Splash.h"
#include "mainsimulador.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TFrmSplash *FrmSplash;
//---------------------------------------------------------------------------
__fastcall TFrmSplash::TFrmSplash(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TFrmSplash::Button1Click(TObject *Sender)
{
  FrmSplash->Hide();
 frmmain->Show();
}
//---------------------------------------------------------------------------
```

ANEXO 2: ALGUNOS AGORITMOS TRADICIONALES PARA EL PROCESADOR DE UN SERVIDOR DESDE PUNTO VISTA CON LOS ALGORITMOS GENÉTICOS

2.1 ALGORITMOS DE PLANIFICACIÓN

En esta sección, se hace un breve resumen de algunos algoritmos tradicionales para el procesador de un servidor como:

- **Primero llego y primero atiende** (**FCFS**: First Come, First Served).
- **Primer trabajo más corto** (**SJF**: Shortest Job First) o **siguiente proceso más corto** (**SPN**: Shortest Process Next).
- **Primero el tiempo restante más corto** (**SRTF**: Shortest RemainingTime First).
- Planificación por prioridad.
- **Planificación por turno circular** (**RR:** Round Robin).
- Realimentación.
- **Siguiente tasa de respuesta más alto** (**HRRN**: Highest Response Ratio Next).

a) Primero llego y primero atiende (**FCFS**: First Come, First Served).

Planificación de servicio por orden de llegada.

Es el algoritmo más sencillo, el primer proceso que solicita a la **unidad central de procesamiento** (CPU) es el primero en recibirla. Fácil de implementar con una política FIFO para la cola de preparado.

Tiempo de espera promedio bastante largo.

Planificación de servicio por orden de llegada. Calcular el tiempo de espera, tiempo de retorno y tiempo medio de espera si aplicamos el algoritmo FCFS suponiendo que los procesos siguientes llegan en el mismo instante y en el orden: P1, P2, P3.

b) Primer trabajo más corto (**SJF**: Shortest Job First) o **siguiente proceso más corto** (**SPN**: Shortest Process Next).

Entra en la **unidad central de procesamiento** (CPU) el proceso con la ráfaga de CPU más corta.

Minimiza el tiempo de espera medio.

Riesgo de inanición de los procesos de larga duración.

No es implementable. Se pueden estimar las duraciones de los procesos, según su historia reciente.

Versión expulsiva SRTF: el proceso en CPU es desalojado si llega a la cola un proceso con duración más corta.

c) Planificación por prioridad

Cada proceso tiene una prioridad, entrará primero en la **unidad central de procesamiento** (CPU) el que tenga mayor prioridad.

Política de prioridades expulsiva o no.

•La prioridad se puede definir:

- ✓ De forma interna, la define el sistema operativo.
- ✓ De forma externa, la definen los usuarios.
- ✓ SJF es un caso de planificación por prioridad.

Los procesos de prioridad más baja tienen riesgo de inanición.

Solución: envejecimiento. Ir aumentando de forma progresiva la prioridad de los procesos en espera.

d) Planificación por turno circular (RR: Round Robin)

- • Adecuado para implementar tiempo compartido.
- • Comportamiento como FCFS, con la diferencia de que cada proceso dispone de un tiempo máximo.
  - ✓ Si cuando expira el tiempo de proceso continúa en la CPU, el planificador lo desaloja y lo ingresa al final de la cola de preparado.
- • La cola de preparado se gestiona como **primero de entrar y primero de salir** (FIFO).
- • Según sea el cuanto de tiempo, Q:
  - ✓ Si Q es muy grande, los procesos terminan sus ráfagas de CPU antes de que termine el tiempo: se comporta como un FCFS.
  - ✓ Si Q=>0, se tiende a un sistema en el que cada proceso dispone de un procesador a 1/N de la velocidad del procesador real (procesador compartido).
  - ✓ Si Q es muy pequeño se suceden constantemente los cambios de contexto y el rendimiento disminuye mucho.

2.2 SISTEMAS OPERATIVOS PARA MULTIPROCESO (SOPMultproceso)

La interconexión entre los procesadores determinará las funciones de control del sistema operativo. El control será centralizado en el caso de organizaciones jerárquicas o verticales, y distribuido en organizaciones horizontales, en las que todos los procesadores son iguales desde el punto de vista lógico.

En un sistema con múltiples procesos concurrentes puede ocurrir que dos procesos traten de acceder a la vez a recursos que no pueden ser utilizados simultáneamente por más de un procesador (unidades de disco, I/O, etc.), o bien traten de acceder a recursos virtuales, como tablas dé datos o búfer de comunicación entre procesos. En estos casos, el encargado de proporcionar mecanismos para garantizar un uso exclusivo del recurso es el sistema operativo, y los procesos que deseen acceder al recurso deben competir por él. A esta exclusividad de acceso se le denomina exclusión mutua entre procesos.

a) Clasificación de los sistemas operativos para multiproceso

Básicamente, se han utilizado tres tipos de organizaciones en el diseño de sistemas operativos para sistemas multiprocesadores:

1) Configuración maestro-esclavo.
2) Supervisor independiente para cada procesador.
3) Supervisor flotante.

2.3 COMPARACIÓN ENTRE LAS METODOLOGÍAS DE ALGORITMOS TRADICIONALES PARA PROCESOS Y LOS ALGORITMOS GENÉTICOS

En esta sección, se hace un breve resumen de comparación entre las metodologías de algoritmos tradicionales para procesos y los algoritmos genéticos, para describir una tabla 2 de comparación entre algoritmos tradicionales para procesos y los algoritmos genéticos, se ha tomado como base del conocimiento de la presente Tesina. Se muestra en la tabla 2.

Tabla 2. Comparación entre algoritmos tradicionales para procesos y los algoritmos genéticos

Agoritmos Requerimientos o procesos	Primero llego primero atiende	Primero el trabajo más corto	Primero el tiempo restante más corto	Planificación por prioridad	Turno circular	Realimentación	Sistema sistema para multproceso	Algoritmo genét. simulador
Nodos son generados por el usuario	genera el sistema	genera el sistema	genera el sistema	genera el sistem	genera el sistema	genera el sistema	genera el sistema	genera el usuario
Algoritmo es robusto	no	no	no	no	no	no	a veces	sí
Soluciona problema complejo	no	no	no	no	no	no	a veces	sí
Procesos son generad. por el usuario	genera el sistema	genera el sistema	genera el sistema	genera el sistem	genera el sistema	genera el sistema	genera el sistema	genera el usuario
Número de iteraciones	determina el sist.	determina el sist.	determina el sist.	determina el sist.	determina el sist.	determina el sist.	determina el sist.	deter. el usuario
Velocidad alta de procesos	no	no	no	no	no	no	a veces	sí
Optimiza los procesos	poco	poco	poco	poco	poco	poco	poco	sí