A-Teams para la minización del caudal total turbinado de una represa hidroeléctrica.

Este trabajo está dedicado a mis padres y hermanas. Espero poder retribuirles el apoyo, el cariño y la comprensión que me brindaron durante todos estos años de grandes sacrificios.

ÍNDICE

CAPÍTULO 1: INTRODUCCIÓN

1.1 CONSIDERACIONES INICIALES

El Paraguay es un país exportador de energía eléctrica gracias a sus dos represas hidroeléctricas de gran envergadura: Itaipú y Yacyretá. Estas centrales de generación poseen múltiples unidades generadoras, con 18 turbinas en Itaipú y 20 turbinas en Yacyretá [1-2]. Debido a la importancia de optimizar la utilización de los recursos hídricos (con el fin de satisfacer la creciente demanda de energía eléctrica), se hace necesaria la implementación de un sistema de control que optimice la generación de energía (producir la potencia requerida minimizando la cantidad de agua utilizada).

Gracias a los rápidos avances en los campos de la informática y las comunicaciones, actualmente todas las centrales de generación se encuentran automatizadas en gran parte, incluyendo la generación de energía, control y diagnóstico precoz de fallas [3-4]. La automatización está dada por la instalación de sistemas computacionales, sensores y sus correspondientes dispositivos de comunicación [3-4] y la implementación de algoritmos o programas especializados que procesan la información proveniente de los sensores, ya sea para realizar el monitoreo y diagnóstico de las turbinas en funcionamiento como para el proceso de generación de energía propiamente dicho. De este modo, con dichos sistemas se puede controlar el caudal a ser turbinado en las unidades generadoras para producir energía, y satisfacer así la demanda de potencia. Igualmente, con la automatización de una central de generación, es posible establecer un control altamente confiable de los parámetros que hacen al funcionamiento físico de cada turbina (temperatura, presión, vibración, desgaste, etc.) para prevenir así posibles fallas disminuyendo el número y frecuencia de paradas de las turbinas, mejorando el desempeño de las unidades más importantes de la usina hidroeléctrica, y aumentando su vida útil [3].

Sin embargo, en las instalaciones hidroeléctricas de gran envergadura, la generación eficiente de energía es aún hoy un tema de continua investigación, en lo que se refiere a la optimización en la generación de energía [4]. Dicha optimización consiste en encontrar el punto óptimo de operación de la usina teniendo como datos la altura del agua, las curvas de eficiencia de cada turbina (obtenida a partir del "*index-test*"), fajas prohibidas de generación, número de turbinas en servicio, entre otros parámetros. El óptimo de generación (punto óptimo de operación) para el presente trabajo es aquel en el cual con un determinado número N_T de turbinas en funcionamiento se genera la potencia requerida (P_D) para una determinada zona de consumo, usando en las unidades generadoras la menor cantidad posible de agua (Q_{Total}) del embalse.

En la literatura se conocen numerosos métodos numéricos de optimización tradicionalmente aplicados para la obtención del punto óptimo de generación en represas hidroeléctricas de gran porte [4]. La desventaja principal que poseen estos métodos numéricos es el elevado tiempo de computación que invierten para la resolución del problema volviéndose no viables para la implementación en centrales de generación que utilizan numerosas unidades generadoras.

De modo a lidiar con las crecientes dimensiones de estos problemas, en las últimas décadas surgió una alternativa que consiste en descomponer un problema global en varios problemas más pequeños (subproblemas). A continuación, se aplica sobre cada subproblema un método numérico adecuado para resolverlo. La metodología así descripta, en la que se combinan diversos métodos numéricos para resolver un problema grande y complejo, se denomina *Team Algorithm* (TA) [5].

Recientemente ha surgido una técnica que mejora aún más el desempeño de los *Team Algorithm*. Dicha técnica se sirve de los grandes logros y avances tecnológicos en el campo de las comunicaciones entre computadoras y entre los diversos procesadores de un mismo sistema computacional. La técnica consiste en resolver cada subproblema del problema global en un procesador diferente de una red de computadoras. Si se aplican sobre cada subproblema un método numérico adecuado pero diferente al resto de los métodos escogidos, se tiene la versión *paralela* de los *Team Algorithms* [5-6].

Con respecto a la implementación de este método, algunos investigadores aplicaron técnicas de Inteligencia Artificial sobre algunos de los subproblemas, utilizando métodos numéricos tradicionales en los restantes [7]. Otros, en cambio, implementaron técnicas de Inteligencia Artificial combinadas con técnicas matemáticas tradicionales sobre cada subproblema [8]. Todas estas implementaciones paralelas, denominadas genéricamente *A-Teams* (Asynchronous Teams) debido a que la comunicación de datos entre procesadores se realiza de forma asíncrona, mostraron ser más eficientes (con respecto a la velocidad de procesamiento y calidad de los resultados logrados) que las implementaciones tradicionales [5-11].

En el presente trabajo se plantea la posibilidad de utilizar los *A-Teams* combinando técnicas de Inteligencia Artificial con métodos numéricos de optimización conocidos para la minimización del caudal total turbinado de una represa hidroeléctrica de gran envergadura (como Itaipú y Yacyretá) para optimizar de esta manera la generación de energía eléctrica (menor costo de producción y mayor rapidez en el cálculo del caudal mínimo).

1.2 BREVE DESCRIPCIÓN E HISTORIA DE LOS *TEAM ALGORITHMS*

Los grandes problemas de ingeniería y específicamente, los relacionados con grandes sistemas eléctricos, son generalmente representados por complejas ecuaciones matemáticas. Como ejemplo, se pueden citar los problemas del flujo de potencia, el punto de colapso, etc. Un sistema que también es comúnmente estudiado en ingeniería consiste en una central de Generación de energía eléctrica constituida por múltiples turbinas monitoreadas y controladas por sensores [4].

Una de las razones por la que se modela matemáticamente este tipo de sistema es la necesidad de obtener el punto de operación óptimo (con respecto al máximo o al mínimo de algunas magnitudes asociadas al sistema) utilizando métodos numéricos de cálculo, los cuales son implementados (generalmente en forma secuencial) en computadoras personales, para aprovechar la capacidad que tienen las mismas de realizar millones de cálculos por segundo. Por su lado, la principal característica que poseen las representaciones matemáticas de sistemas físicos es la de constituir funciones matemáticas muy complejas formadas por un gran número de variables independientes (parámetros que definen el valor numérico de la función), y un gran número de picos (valores máximos) y valles (valores mínimos) a lo largo del dominio de la misma. También, la función podría presentar características peculiares como, por ejemplo, ser discontinua y/o poseer numerosas restricciones.

Se tienen a la fecha numerosos métodos de optimización de probada eficacia [12]. Sin embargo, las funciones que modelan los sistemas reales mencionados son muy difíciles de optimizar utilizando los métodos numéricos tradicionales, porque:

- los métodos tradicionales generalmente encuentran un óptimo local, y no el óptimo global buscado, debido a la característica *multimodal* de la función (existencia de numerosos picos y valles);

- dichos métodos invierten mucho tiempo de computación, debido a la elevada cantidad de variables independientes (parámetros que definen a la función) que pueden llegar a tener las funciones;
- ocasionalmente podría suceder que, para determinados problemas de optimización [5], ninguno de los métodos tradicionales disponibles sea aplicable para buscar el punto óptimo.

Por estas razones, se comenzó a pensar en la posibilidad de combinar diferentes métodos numéricos con el fin de obtener un algoritmo que sea capaz de optimizar grandes problemas superando estas desventajas.

En este contexto, se resalta el trabajo de Dusonchet et al. [13], quienes en 1971 desarrollaron uno de los primeros trabajos en el que se combinan diferentes algoritmos en la resolución de problemas de ingeniería, tales como el problema del flujo de potencia eléctrica, combinando métodos numéricos conocidos, tales como el método de Jacobi y el método de Newton, presentando resultados positivos. En este trabajo ya se presentaba la idea de dividir un sistema de ecuaciones algebraico grande, constituido por un gran número de variables, en varios sistemas de ecuaciones más pequeños, de modo que sobre cada uno de ellos se aplique un método numérico adecuado a las características propias del subconjunto de ecuaciones a resolver. De este modo, con el trabajo de Dusonchet et al. comienza la motivación para el desarrollo de los *Team Algorithms*, especialmente en lo que se refiere a dividir un problema en subproblemas menores y aplicar a cada subproblema un método numérico adecuado para resolverlo.

A partir del trabajo de Marschak y Radner [14] se comienza a mencionar la idea de combinar diferentes algoritmos para la resolución de un mismo problema global. Nace así la posibilidad de usar estos algoritmos en sistemas distribuidos, que son sistemas computacionales formados por un conjunto de procesadores con capacidad de intercambiar mensajes o datos.

Por esta razón, a partir del mencionado trabajo, y durante toda la década de los 80 [5], comienzan los *Team Algorithm* a ser estructurados como una combinación de diferentes algoritmos (o métodos) aplicados a cada procesador de un sistema distribuido; como por ejemplo, el trabajo desarrollado en 1982 por Talukdar, Pyo y Giras [15] y su extensión en Talukdar, Pyo y Mehrotra en 1983 [16]. En dichos trabajos se desarrolló un *Team Algorithm* formado por una combinación de diferentes métodos, contando además con un *Administrador*, que consiste en un proceso que controla y supervisa la comunicación entre los diversos métodos que están siendo ejecutados en los diferentes procesadores de un sistema distribuido. El algoritmo anterior fue aplicado a la resolución de sistemas de ecuaciones algebraicas, formados por una gran cantidad de variables, demostrándose así que dicho algoritmo es capaz de resolver problemas que ninguno de los métodos combinados consigue resolver cuando son aplicados en forma independiente.

En el trabajo realizado por Talukdar et al. [7] en 1991, se desarrolló un *A-Team* en el que se combinó el Algoritmo Genético, que es una técnica de Inteligencia Artificial, con un método numérico conocido, tal como el método de Newton, aplicado a la resolución de sistemas de ecuaciones algebraicas no lineales. Los resultados obtenidos en este trabajo fueron muy promisorios confirmando las ventajas de utilizar los *A-Teams* en complejos sistemas de ecuaciones matemáticas.

A su vez, en el trabajo desarrollado por Talukdar, Ramesh y Nixon [17] se estableció una analogía entre los *A-Teams* y una sociedad de insectos, basado en el trabajo de Fox [18], que postula la posibilidad de obtener un efecto sinergético, es decir, mostrar que la combinación de algoritmos es más que una simple suma de propiedades de cada uno de los algoritmos combinados. En estos trabajos también se describe la forma de aplicar los *A-Teams* en sistemas de administración de energía (Energy Managements Systems), control de redes eléctricas y otras aplicaciones relacionadas con sistemas de potencia.

Una discusión más completa puede ser encontrada en Ramesh et al. (1991) [19], en la cual los *A-Teams* son utilizados en aplicaciones tan diversas como: soluciones de sistemas de ecuaciones algebraicas no lineales, combinación de Algoritmos Genéticos (GA) con métodos de cálculo intensivo, proyecto de edificios, control de redes eléctricas y otras aplicaciones de sistemas de potencia.

Las últimas publicaciones mencionadas se constituyeron en la base inspiradora del presente trabajo, teniendo en mente que una represa hidroeléctrica con múltiples unidades generadoras y con restricciones operativas está representada por un modelo matemático complejo [4]. Por consiguiente, se plantea en el presente trabajo la utilización de los *A-Teams* para aplicar en este tipo de problemas de optimización.

1.3 TÉCNICAS DE INTELIGENCIA ARTIFICIAL

La naturaleza presenta una gran variedad de problemas de optimización, tales como: la supervivencia y evolución de las especies, la búsqueda del camino crítico por parte de una sociedad de insectos desde su nido hasta la fuente de su alimentación y otros tantos ejemplos en los que la misma naturaleza se encargó de proveer metodologías para solucionar dichos problemas. Los investigadores consideraron que dichas soluciones encontradas por la naturaleza son dignas de ser imitadas para ser aplicadas a problemas que los métodos disponibles en la actualidad aún no pueden resolver satisfactoriamente (como la optimización de las funciones matemáticas que modelan ciertos sistemas físicos). Por esa razón, los investigadores diseñaron métodos computacionales (algoritmos) que imitan los procesos naturales, presentando características que los vuelven muy atractivos en la resolución de problemas de gran porte [20]. Entre tales características pueden citarse:

- *Transportabilidad*: los algoritmos que imitan procesos naturales pueden ser implementados en cualquier arquitectura de computadoras.

- *Robustez*: el algoritmo es capaz de hallar muy buenos resultados para una amplia gama de problemas.
- *Implementación relativamente fácil*.

Estos métodos computacionales son comúnmente denominados algoritmos o técnicas de Inteligencia Artificial [21]. Entre las técnicas de Inteligencia Artificial resaltan las **Redes Neuronales**, que son algoritmos inspirados en la enorme capacidad de procesamiento del cerebro humano. Esta metodología realiza una simulación computacional del comportamiento de las neuronas del cerebro [21-22]. La principal aplicación que han tenido las **Redes Neuronales** ha sido en la predicción de resultados a partir de un conjunto de datos, aunque también han sido aplicadas a la optimización de funciones. También se las suele aplicar en el procesamiento de imágenes y en la visión y reconocimiento de patrones gráficos.

Otra técnica de Inteligencia Artificial, la cual es aplicada principalmente en la optimización de problemas del tipo combinatorial, es el método del **Temple Simulado** (*Simulated Annealing Method*) [12 Cap.10]. Este método imita el proceso termodinámico del temple, el cual está relacionado con la manera en que los líquidos se congelan y cristalizan, buscando llegar al estado de mínima energía, la cual puede asociarse a la función objetivo a ser optimizada.

Una rama muy importante de las técnicas de Inteligencia Artificial es la llamada **Computación Evolucionaria**, la que, partiendo de la premisa que la Teoría de la Evolución de Darwin es un proceso adaptativo de optimización, sugiere un modelo en el que poblaciones de estructuras computacionales evolucionan mediante la aplicación de operadores análogos a los utilizados en la naturaleza con el fin de mejorar el desempeño de la población. El desempeño de la población se asocia a la función objetivo que se quiere optimizar, y se busca mejorar dicho desempeño en la medida en que el valor de la función se aproxima al mejor valor posible (óptimo global) [21].

Actualmente, la **Computación Evolucionaria** engloba un número considerable y cada vez creciente de métodos [23], entre los cuales los más importantes son los **Algoritmos Genéticos** (AG) [21, 24-26], la **Programación Evolucionaria** [27], **Estrategias Evolucionarias** [28], **Programación Genética** [29], y los **Sistemas Clasificadores** [24 Cap. 6, 30]. Entre todas estas técnicas citadas, los **Algoritmos Genéticos** son los más difundidos y estudiados debido a su flexibilidad, relativa simplicidad de implementación, y una gran eficacia en realizar búsqueda global en ambientes adversos.

Como se describirá en los capítulos siguientes, la ventaja principal que poseen los **Algoritmos Genéticos** consiste en el hecho de realizar la búsqueda del punto óptimo a partir de una "población" de puntos, y no a partir de un solo punto, existiendo por lo tanto, un *paralelismo* implícito en la metodología de búsqueda de esta técnica. Esta característica implícita de búsqueda paralela puede ser particularmente aprovechada para dividir la población de puntos iniciales y colocarlas en diferentes procesadores de un sistema distribuido y aplicar el **Algoritmo Genético** en cada procesador, buscando así mejorar la eficiencia en la optimización global con respecto a la calidad de la solución encontrada y el tiempo de computación utilizado [8, 10].

1.4 IMPORTANCIA ECONÓMICA

La eficiente y económica producción de energía por parte de sistemas de generación de grandes dimensiones ha sido siempre de gran importancia en los países en desarrollo. Al respecto, la situación del Paraguay en el campo del abastecimiento de energía eléctrica ha sido, y es, muy especial, hasta tal punto que es verdaderamente difícil que pueda encontrarse un caso similar en todo el mundo.

Aunque el Paraguay posee variados recursos naturales, tales como el gas natural, caudalosos ríos, etc., antes de la firma del Tratado con el Brasil [31], no disponía de la adecuada tecnología y del capital financiero necesario para el aprovechamiento eficiente de dichos recursos de modo a transformarlos en energía aprovechable, mediante la construcción de centrales de generación con gran potencia instalada. Además, aún aprovechando las cuencas de los ríos internos que ofrecen mejores condiciones de caudal (como el Acaray, el Monday, el Ñacunday, etc.) solo hubiera podido generarse entre 2000 MWh a 4000 MWh al año, cantidad que, ante el crecimiento industrial y poblacional del país, pronto sería totalmente absorbida, forzando al país a buscar otras fuentes de energía [31].

Ante este panorama, existió siempre la posibilidad de recurrir al aprovechamiento energético de uno de los ríos más caudalosos del cono sur: el río Paraná. Esto llevaría aparejada la construcción de una usina hidroeléctrica de grandes dimensiones. Con la firma del Tratado de Itaipú, los dos países signatarios, Paraguay y Brasil, se comprometieron a llevar adelante la construcción de una central hidroeléctrica de gran envergadura con vista al aprovechamiento hídrico del río Paraná. Actualmente la represa hidroeléctrica de Itaipú posee una potencia instalada de 12600 MW y una capacidad de producción anual de más de 70000 GWh [1, 31]. Un emprendimiento análogo fue llevado a cabo entre argentinos y paraguayos para la construcción de la usina hidroeléctrica de Yacyretá, la cual posee una potencia instalada de 4800 MW y una capacidad de producción anual de 20000 GWh [2]. De esta forma, Paraguay tiene gracias a estas dos grandes instalaciones una considerable reserva energética asegurando de esta manera la cobertura de su consumo eléctrico presente y futuro a un largo plazo.

A su vez, el costo de la energía producida por una central de generación hidroeléctrica, no depende solamente de los costos de los servicios de mantenimiento sino también de la cantidad de agua turbinada. Cuanto menor sea la cantidad de agua que se utilice para generar la misma potencia requerida, los costos de producción disminuirían al igual que las tarifas de consumo [4]. Además, otro aspecto que se ha venido estudiando en las últimas décadas está relacionado con el aumento de la disponibilidad energética de las centrales de generación. La disponibilidad de energía está relacionada con la cantidad de agua que permanece en el reservorio (embalse) para producir energía eléctrica, en el caso de una usina hidroeléctrica; o bien, con la cantidad combustible (fuel, diesel, etc.) que se mantenga en el tanque, en el caso de una central térmica [4].

De todo esto, se deduce la consecuencia de la optimización en la utilización del recurso energético (por ejemplo: el caudal turbinado de una central hidroeléctrica) de tal forma que el costo de producción se reduzca y aumente la disponibilidad energética. Al respecto, ya se han venido estudiando técnicas matemáticas computacionales a ser implementadas en los sistemas automáticos que controlan los arranques y paradas de las unidades generadoras, así como también en la producción de potencia eléctrica de un conjunto de generadores con el fin de satisfacer la demanda, utilizando la menor cantidad posible de fluido combustible (agua, vapor, fuel, etc.) [4 Cap. 3, 17, 32].

1.5 OBJETIVOS Y ORGANIZACIÓN DEL PRESENTE TRABAJO

El objetivo principal del presente trabajo consiste en la minimización del caudal turbinado de una represa hidroeléctrica de gran envergadura (como Itaipú y Yacyretá), en la generación de la potencia demandada. Para llegar a tal objetivo, se propone:

a) Implementar métodos numéricos computacionales tradicionalmente utilizados para la optimización de la generación de energía en represas hidroeléctricas de gran envergadura.

b) Combinar técnicas de Inteligencia Artificial (como el Algoritmo Genético) con el método numérico desarrollado anteriormente, buscando mejorar el tiempo de cálculo. El algoritmo que resulta de tal combinación se denomina comúnmente Algoritmo Genético Combinado [10].

c) Implementar el Algoritmo Genético Combinado en un ambiente paralelo asíncrono. El algoritmo así implementado es conocido en la literatura como *A–Teams* (Asynchronous Teams) [7].

d) Aplicar el *A–Team* desarrollado en la minimización del caudal turbinado de una represa hidroeléctrica de gran envergadura. Para lograrlo se utilizarán datos reales de represas hidroeléctricas existentes en el país.

e) Utilizando los datos obtenidos de las turbinas correspondientes a una determinada represa hidroeléctrica, comparar el desempeño del *A–Team* desarrollado con el método numérico tradicionalmente utilizado en este tipo de problemas, mediante resultados experimentales, con el fin de comprobar los aportes de la presente propuesta.

f) Realizar una estimación económica de la posible implementación del *A-Team* desarrollado para el planeamiento de la generación de energía de una represa hidroeléctrica, en base a los resultados experimentales.

De esta forma, se tendrá una herramienta computacional que puede ser aplicada sobre una red de computadoras como las existentes en nuestro país sin necesidad de recurrir a las costosas supercomputadoras disponibles en países más desarrollados, combinando técnicas de Inteligencia Artificial con técnicas matemáticas tradicionales para lograr reducir el tiempo de cálculo mejorando al mismo tiempo los resultados obtenidos.

El trabajo fue organizado en ocho capítulos. La automatización de las usinas hidroeléctricas y las ventajas obtenidas gracias a ella son descriptas en el Capítulo 2.

En el Capítulo 3 se describirán algunos métodos numéricos utilizados comúnmente para la optimización en la generación de energía y luego se explicará detalladamente el método numérico utilizado en el presente trabajo.

El Algoritmo Genético será descrito en el Capítulo 4; donde se hablará sobre las características que los vuelven aptos para la optimización global de funciones muy complejas, en las que los métodos numéricos tradicionales ya no tienen un buen desempeño. También se expondrán los operadores probabilísticos utilizados en el Algoritmo Genético. Se mostrarán además los fundamentos matemáticos de esta técnica, describiendo por último a los Algoritmos Genéticos Paralelos.

La combinación de diferentes algoritmos en un ambiente distribuido asíncrono, que se conoce en la literatura como *A-Team*, será descripta en el Capítulo 5, mientras que la metodología a seguir para la implementación de esta combinación para el cálculo del caudal mínimo a ser turbinado será tratada en el Capítulo 6.

En el Capítulo 7, se analizará un ejemplo práctico con vista a levantar resultados experimentales, que consistirán en una usina hidroeléctrica con su correspondiente número de turbinas. Dichos resultados serán comparados con los resultados obtenidos por un método numérico tradicionalmente utilizado para la optimización de la generación de energía, utilizando para ello curvas y tablas. Las conclusiones que se obtengan de los resultados experimentales serán expuestas en el Capítulo 8 del presente trabajo.

CAPÍTULO 2: AUTOMATIZACIÓN DE USINAS HIDROELÉCTRICAS

2.1 GENERALIDADES

La incorporación de tecnología de punta en las grandes industrias y el aumento de la demanda de potencia eléctrica debido al creciente número de usuarios, ha llevado a la sociedad a ser cada vez más dependiente del consumo energía eléctrica. Este hecho también ha influido sobre los sistemas eléctricos (encargados de la distribución de la potencia eléctrica a los distintos puntos de consumo) haciendo que éstos sean cada vez más grandes y complejos [33].

A su vez, el aumento en las dimensiones y complejidad de los sistemas de potencia eléctrica requiere de la implementación de un sistema de control y supervisión de fallas en el sistema eléctrico, de tal forma a garantizar que la distribución de energía a los puntos de consumo (por parte de la Central de Administración de Energía Eléctrica de cada país) sea estable. Otras de las necesidades requeridas por parte de las Empresas Distribuidoras de Energía Eléctrica es la de ser capaces de realizar un continuo chequeo del estado en que se encuentra el sistema eléctrico que administran con el fin de prever posibles ampliaciones del sistema eléctrico de determinadas zonas, así como también la modernización de los equipamientos eléctricos, controlar la demanda de potencia, y supervisar y diagnosticar posibles fallas [33].

El vertiginoso desarrollo de la informática y las redes de computadoras así como los notables avances en la tecnología de las comunicaciones, ha hecho posible la automatización de los procesos de control y supervisión de fallas, así como la distribución de energía eléctrica de los sistemas eléctricos atendidos por las centrales de administración de energía de varios países, teniendo de esta manera un continuo chequeo de los distintos puntos de consumo y del estado en que se encuentra el sistema eléctrico [33].

La misma tecnología en comunicaciones y las redes de computadoras utilizadas para el control y supervisión de los sistemas eléctricos, es utilizada también en las centrales hidroeléctricas de generación con el fin de realizar un monitoreo *on-line* de puntos importantes de las mismas, como las turbinas en la casa de máquinas, los alternadores, los transformadores de elevación y demás equipamientos para la transmisión de energía existentes en la subestación de transmisión [3, 34]. Las terminales de la red de computadoras están conectadas generalmente a una computadora central (servidor) la que a su vez está conectada mediante un determinado sistema de conexión con múltiples sensores colocados en cada unidad generadora y demás equipamientos de generación y transmisión de energía. De esta forma, se realiza la constante supervisión del sistema de generación y transmisión de la central a la vez que algoritmos computacionales implementados en las computadoras de la red procesan los datos provenientes de los sensores y con los resultados obtenidos, pueden realizarse mandos a distancia y optimizar, de esta forma, la producción de energía eléctrica.

2.2 SISTEMAS AUTOMÁTICOS DE CONTROL

Actualmente, las represas hidroeléctricas que son implementadas en diversos países están provistas de múltiples unidades generadoras (o bien, aquellas que están en funcionamiento desde hace mucho tiempo son sometidas a trabajos de modernización dotándolas de equipamientos más modernos y de mayor capacidad) en respuesta a la creciente utilización de la energía eléctrica y al aumento en dimensiones y complejidad de los sistemas eléctricos encargados de transmitir y distribuir la energía eléctrica a los consumidores (usuarios) y a las grandes industrias.

Las represas hidroeléctricas de gran envergadura constituyen la principal fuente de suministro de energía eléctrica en muchos países, y los picos de carga de determinadas zonas de consumo son cubiertos por las pequeñas centrales térmicas correspondientes. Además, las unidades generadoras así como los demás equipamientos relacionados (compuertas, alternadores, transformadores de elevación, etc.) son muy costosos, y por consiguiente es necesario realizar un constante chequeo sobre las condiciones de operación de las mismas con el fin de prevenir posibles desperfectos. Por esta razón, para poder realizar un efectivo control y mantenimiento de este sistema complejo se hace necesario la automatización, mediante sistemas de control y supervisión, utilizando una red de computadoras y sensores, todos enlazados mediante sistemas de comunicación.

Los principales sistemas de control y supervisión de una central hidroeléctrica de gran porte son los siguientes [34]:

- Control en la Generación: mediante programas computacionales se procesan las informaciones provenientes de las turbinas. Los resultados obtenidos del procesamiento (correspondientes al caudal a ser utilizado, a la potencia a ser generada, a las unidades que serán utilizadas, etc.) son utilizados para la generación y suministro de la potencia requerida.
- Control de las compuertas de la represa: se utiliza un sistema computacional que controla los equipamientos correspondientes a las compuertas de la represa.
- Control y supervisión de las unidades generadoras: se instala un sistema que controla *on-line* el estado en que se encuentran todas las unidades generadoras de la central.
- Sistema de Control y Supervisión de las subestaciones de transmisión.

Cada uno de los sistemas de control listados están compuestos a su vez de otros subsistemas de control. Estos subsistemas controlan y supervisan zonas aún más específicas en una central de generación [34], como el control de nivel del agua en el embalse, la temperatura de funcionamiento de la turbina, las vibraciones producidas durante el funcionamiento de las turbinas entre otras.

De entre todos estos sistemas de control, el sistema que reviste mayor atención en una central hidráulica de generación está dado por el conjunto de unidades generadoras, implementándose sistemas como el SCADA (*Supervisory Control and Data Acquisition*) que consiste en un sistema de control y supervisión que está formado por una red de computadoras conectadas, mediante un sistema de comunicación, a los distintos sensores instalados en las unidades generadoras y demás equipamientos relacionados de la central, de tal forma a tener un monitoreo *on-line* del estado en que se encuentran las máquinas y los equipos [35].

En este sistema, para realizar la supervisión diaria del sistema generador, los sensores envían los datos correspondientes (de acuerdo a las partes de las turbinas que controlan: como la temperatura, la presión, las vibraciones, tensiones, etc) y diversos algoritmos procesan dicha información para presentar mediante históricos, estadísticas y gráficos el estado en que se encuentran los múltiples generadores de la central hidroeléctrica [3, 35] así como otras partes de la central (las compuertas, los vertederos, el embalse, etc.). De hecho, un sistema SCADA, similar al descripto arriba, se está utilizando actualmente en la Represa Hidroeléctrica de Itaipú como sistema de control y supervisión.

El servidor, que es la computadora principal que realiza el procesamiento de las informaciones provenientes de los sensores y administra las comunicaciones entre las terminales de la red de computadoras, es una computadora de gran capacidad, y en algunos países desarrollados, por una supercomputadora (o *mainframe*) debido a su enorme capacidad de procesamiento [35].

De esta forma, los objetivos que se pretende lograr al instalar un sistema de control y supervisión del conjunto generador pueden ser enumerados de la siguiente manera [3]:

a) protección del equipo;

b) control del desgaste debido al rozamiento del rotor. En este caso las fallas pueden ser detectadas incluso en su estado incipiente;

c) prolongación de la vida útil, debido a que si se detecta la falla con anticipación se pueden detectar los defectos que posiblemente afecten al funcionamiento de las unidades generadoras;

d) se aumenta la eficiencia, debido a que se obtiene mayor tiempo de operación con menor tiempo de parada y/o mantenimiento, gracias al mantenimiento predictivo;

e) aumenta la confiabilidad. Un sistema de control y supervisión monitorea constantemente el estado de las unidades generadoras.

Sin embargo, en una represa hidroeléctrica de gran porte, las turbinas constituyen las piezas más importantes y costosas, en el proceso de generación de energía; por esa razón, son los que precisan de mayores cuidados y mantenimientos. Debido a esto, se suelen implementar grandes sistemas computacionales que controlan y supervisan el funcionamiento de las turbinas y el estado en que se encuentran las mismas. Las informaciones obtenidas gracias a este sistema de control permiten optimizar el plan de arranques y paradas de las turbinas, detectar posibles fallas mecánicas, actualizar las curvas de eficiencia, etc. El sistema que controla la generación de potencia está formada por algoritmos computacionales que realizan la optimización propiamente dicha, y los resultados obtenidos son utilizados para la generación óptima de la potencia requerida.

Este sistema de control que supervisa y administra la generación de potencia eléctrica (optimizando la generación de energía) es muy utilizado especialmente en aquellas represas en las que un gran número de turbinas están siendo utilizadas y debido a las dimensiones del sistema eléctrico que alimentan, se hace conveniente utilizar menor cantidad de caudal para generar energía [36], de tal forma a conservar siempre disponible en el embalse un caudal necesario que satisfaga demandas de potencia futuras.

2.3 ALGORITMOS DEL SISTEMA DE CONTROL DE GENERACIÓN

Las represas hidroeléctricas que poseen sistemas de control de generación utilizan técnicas matemáticas tradicionales como el método del gradiente y métodos basados en el multiplicador de Lagrange [4, 36-37], como algoritmos que procesan los datos correspondientes a los parámetros para la generación de energía. Los parámetros que tienen en cuenta estos algoritmos son los siguientes:

1. Curvas características de funcionamiento (potencia [MW] vs. Caudal [m^3/s]) de cada unidad generadora. En estas curvas características también se encuentran incluidas las posibles zonas prohibidas de generación de determinadas turbinas.
2. Altura del agua en el embalse (h).
3. La potencia que se debe producir (P_D).
4. Restricción operativa adicional, como el número de turbinas que deben dejar de operar para mantenimiento y/o reparación.

Sin embargo, la implementación de estos algoritmos en el sistema de control de generación tiene el siguiente inconveniente: los métodos numéricos utilizados tradicionalmente para la optimización en la generación de energía invierten mucho tiempo (algunos incluso meses de cálculo) para llegar a obtener buenos resultados. Una alternativa para aumentar la velocidad de procesamiento, utilizada por países desarrollados, es la de implementar dichos algoritmos en supercomputadoras por la enorme capacidad de procesamiento que poseen, lo que resulta en una implementación altamente costosa.

A su vez, otra alternativa para disminuir el tiempo de computación de los algoritmos tradicionales consiste en implementar métodos de optimización no convencionales como los algoritmos de Inteligencia Artificial. Los algoritmos de Inteligencia Artificial han demostrado tener un mejor desempeño (tanto en la velocidad de computación como en la calidad de los resultados obtenidos) que las técnicas matemáticas tradicionales. De hecho, versiones de algoritmos como el *Simulated Annealing*, los Algoritmos Genéticos y los *Fuzzy Set* (técnica de IA basada en el razonamiento difuso) ya fueron implementados para la optimización en la generación de energía [34, 37-38]. Sin embargo, dichos algoritmos fueron implementados en supercomputadoras, las que por su elevado costo son imposibles de acceder para países en vías de desarrollo, como el nuestro.

Inspirados en el auge de algoritmos paralelos asíncronos, implementados en redes de computadoras de características heterogéneas, la propuesta del presente trabajo consiste en la de aprovechar las redes de computadoras existentes en nuestro medio (formadas por computadoras personales de distintas arquitecturas y diferentes capacidades de procesamiento) desarrollando una versión paralela del Algoritmo Genético combinado con técnicas matemáticas tradicionales. De esta forma, no solo disminuirían los costos de implementación (debido a que no se utilizaría los *mainframes* de elevado costo) sino que el desempeño del algoritmo (calidad de resultados obtenidos en menor tiempo de procesamiento) podría ser similar a los obtenidos por los algoritmos implementados en las supercomputadoras.

CAPÍTULO 3: MÉTODOS NUMÉRICOS PARA LA MINIMIZACIÓN DEL CAUDAL TOTAL TURBINADO.

En este capítulo se analizarán los diversos métodos numéricos de optimización utilizados tradicionalmente en la minimización del caudal turbinado de una represa hidroeléctrica de gran porte, con diversas unidades generadoras. Primeramente se planteará el problema general de minimización del caudal total a ser turbinado, luego se abordarán diversos métodos numéricos comúnmente utilizados y se realizará una comparación entre ellos. Finalmente, se presentará el método numérico de optimización utilizado en el presente trabajo, el cual utiliza las ventajas de cada uno de los métodos de optimización vistos anteriormente.

3.1 EL PROBLEMA DE MINIMIZACIÓN DEL CAUDAL TURBINADO

En la **Figura 3.1** se muestra la representación gráfica del problema que será estudiado en este capítulo, y se muestra gráficamente el problema de la minimización del caudal turbinado. El sistema mostrado consiste en N_T unidades de generación, cada una de ellas recibiendo un caudal Q_i ($i \leq N_T$), y generando cada una de ellas una potencia eléctrica P_i. La suma de las potencias generadas por cada una de las unidades debe ser igual a la Potencia total Demandada, P_D.

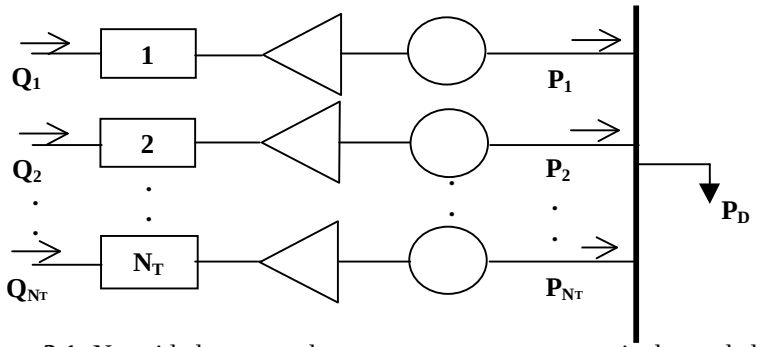


Figura 3.1: N_T unidades generadoras para generar una potencia demandada P_D.

El problema puede ser presentado concisamente como sigue: se tiene una función objetivo, Q_{Total}, que es igual a la suma de los caudales Q_i turbinado en cada unidad generadora *i* para generar la potencia total requerida, P_D. A cada uno de los caudales Q_i turbinado le corresponde una potencia P_i generada por cada turbina *i*. El objetivo del problema es encontrar el mínimo valor posible del caudal total, Q_{Total}, que deberá ser turbinado en la usina hidroeléctrica con la condición de que la suma de las potencias P_i sea igual a la potencia total demandada, P_D. Matemáticamente el problema puede ser presentado de la siguiente manera:

$$\text{Minimizar} \qquad Q_{Total} = Q_1 + Q_2 + Q_3 + \cdots + Q_{N_T} = \sum_{i=1}^{N_T} Q_i(P_i) \qquad \textbf{(3.1)}$$

sujeto a:

$$P_D = \sum_{i=1}^{N_T} P_i \qquad \textbf{(3.2)}$$

donde N_T determina la cantidad de turbinas que se encuentran en servicio.

Debe tenerse en cuenta que existen otras restricciones adicionales como la altura variable del agua (altura del embalse). Además, normalmente cada turbina posee una faja prohibida de generación en la cual la unidad generadora no debe operar; y también puede suceder que determinadas turbinas tengan que parar, para realizar operaciones de mantenimiento o de reparación, variándose de esta forma el número de máquinas en servicio.

3.2 MÉTODO DEL GRADIENTE DE PRIMER ORDEN

En el método del Gradiente la búsqueda del punto óptimo se realiza a partir de una punto factible inicial (punto inicial de búsqueda). Un punto es denominado “punto o solución factible” cuando el mismo cumple con todas las restricciones del problema [4] (función matemática) que se quiere optimizar.

La mencionada búsqueda se realiza a lo largo de una trayectoria que corresponde al gradiente de la función objetivo, en dicho punto. Todos los nuevos puntos encontrados por el método del Gradiente se encuentran sobre esta trayectoria, constituyendo así en nuevos puntos factibles. Por otro lado, si el valor de la función objetivo, Q_{total}, decrece monótonamente, entonces la solución más reciente será la mejor encontrada hasta ese momento.

En el problema presente, un punto o variable dependiente de la función objetivo Q_{Total} está representado por las potencias P_i generadas por cada una de las turbinas. A su vez, la obtención de los valores numéricos de los caudales Q_i correspondientes a cada potencia P_i se realiza utilizando un "*index-test*".

El "*index-test*", contiene los datos necesarios para formar las curvas de eficiencia de cada turbina. La misma está dada por una lista de potencias en función del caudal. Cada lista corresponde a un valor específico de altura disponible. Estas curvas de eficiencia pueden ser representadas matemáticamente por:

$$P_q{}^p = f(Q_q{}^p, h^p) \qquad q = \{1, 2, 3, ..., NoP\} \qquad p = \{1, 2, 3, ..., NoL\} \quad \textbf{(3.3)}$$

donde NoL indica la cantidad de listas de potencia en función del caudal, y NoP corresponde a la cantidad de datos pertenecientes a cada lista *p*.

Para comprender el mecanismo de búsqueda del método del Gradiente, aplicado al problema de minimización del caudal total turbinado de una usina hidroeléctrica, el punto de operación factible inicial se varía en una cantidad muy pequeña. La variación se realiza en la vecindad del punto inicial de búsqueda. Se aplica luego la serie de Taylor sobre dicho punto, obteniéndose la ecuación **(3.4)** que es una expansión en series de Taylor de la función objetivo de la ecuación **(3.1)**, realizado en la vecindad del punto inicial de búsqueda. Se incluyen en esta serie de Taylor los términos de segundo orden:

$$Q_{Total} + \Delta Q_{Total} = Q_1(P_1) + Q_2(P_2) + Q_3(P_3) + \cdots + Q_{N_T}(P_{N_T}) +$$
$$+ \frac{dQ_1}{dP_1}\Delta P_1 + \frac{dQ_2}{dP_2}\Delta P_2 + \frac{dQ_3}{dP_3}\Delta P_3 + \cdots + \frac{dQ_{N_T}}{dP_{N_T}}\Delta P_{N_T} +$$
$$+ \frac{1}{2}\left(\frac{d^2Q_1}{d^2P_1}[\Delta P_1]^2 + \frac{d^2Q_2}{d^2P_2}[\Delta P_2]^2 + \frac{d^2Q_3}{d^2P_3}[\Delta P_3]^2 + \cdots + \frac{d^2Q_{N_T}}{d^2P_{N_T}}[\Delta P_{N_T}]^2 \right) +$$
$$+ \cdots \text{términos de orden superior} \qquad \textbf{(3.4)}$$

Asumiendo que las variaciones son muy pequeñas, entonces las variaciones de segundo orden y de orden superior son tan pequeñas que pueden ser despreciadas, y tenemos finalmente que:

$$\Delta Q_{Total} = \frac{dQ_1}{dP_1}\Delta P_1 + \frac{dQ_2}{dP_2}\Delta P_2 + \frac{dQ_3}{dP_3}\Delta P_3 + \cdots + \frac{dQ_{N_T}}{dP_{N_T}}\Delta P_{N_T} = \sum_{i=1}^{N_T} \frac{dQ_i}{dP_i}\Delta P_i \qquad \textbf{(3.5)}$$

Si se aplica la expansión en series de Taylor a la ecuación **(3.2)** se obtiene una nueva ecuación de restricción:

$$\sum_{i=1}^{N_T} \Delta P_i = 0 \qquad \textbf{(3.6)}$$

Esta ecuación reduce en un grado de libertad el problema, de tal forma que una unidad generadora se constituye en una variable dependiente. Sea "*j*" el índice correspondiente a la variable dependiente. Se deduce de **(3.6)** la siguiente expresión:

$$\Delta P_j = -\sum_{i \neq j} \Delta P_i \qquad \textbf{(3.7)}$$

Las ecuaciones **(3.7)** y **(3.5)** pueden ser combinadas de tal forma a obtener una ecuación que expresa la variación de la función objetivo, Q_{Total}, como una función de la variación de la potencia de salida de cada una de las $N_T - 1$ máquinas independientes [4].

$$\Delta Q_{Total} = \sum_{i \neq j}\left[\frac{dQ_i}{dP_i} - \frac{dQ_j}{dP_j} \right]\Delta P_i = \sum_{i \neq j} \frac{\partial\, Q_{Total}}{\partial\, P_i}\Delta P_i \qquad \textbf{(3.8)}$$

El procedimiento para utilizar esta ecuación se describe en la **Figura 3.2**, el cual representa un Diagrama de Flujo del método del Gradiente. Como se puede ver en el diagrama de flujo, el método del Gradiente comienza con un punto inicial. Dicho punto inicial debe ser un punto de operación factible, es decir, que cumpla con las restricciones del problema dada por la ecuación **(3.2)**. Luego, se selecciona una variable dependiente P_j y a continuación se evalúan los coeficientes de la ecuación **(3.8)**. Los coeficientes de cada una de las $N_T - 1$ potencias i ($i \neq j$) indican el incremento en la función objetivo (Q_{Total}) que debe ser contrarrestado por la potencia P_j. Sin embargo, de acuerdo con [4] también este incremento puede ser contrarrestado por la potencia i cuyo coeficiente $\frac{\partial\ Q_{Total}}{\partial\ P_i}$ es máximo. Al índice de la potencia i que posee el máximo coeficiente (en valor absoluto) se lo designa con i_{max}, es decir $i = i_{max}$. Posteriormente a dichos cálculos se chequea si la potencia P_{imax} ajustada se encuentra fuera de rango, es decir, si se encuentra entre los límites de operación de la turbina, si es así, se vuelve a realizar las operaciones de ajuste para P_{imax} y P_j, hasta que ambos se encuentren dentro de los límites de operación de las unidades de generación correspondientes. Pero, si P_j queda fuera de los rangos de operación, se vuelve a escoger otro índice para la variable dependiente P_j, y nuevamente se realizan las operaciones de cálculo.

Cuando la potencias ajustadas P_{imax} y P_j se encuentran dentro del rango, se realiza el cáculo de la función objetivo, Q_{Total}, con los valores nuevos de las potencias ajustadas y se analiza la magnitud de ΔQ_{Total}. Se verifica antes de imprimir los resultados que $\Delta Q_{Total} \leq$ Tolerancia, si no ocurre así, entonces se vuelve a calcular los coeficientes con los nuevos valores de Q_i. Una vez que se cumpla el criterio de parada se imprimen los resultados.

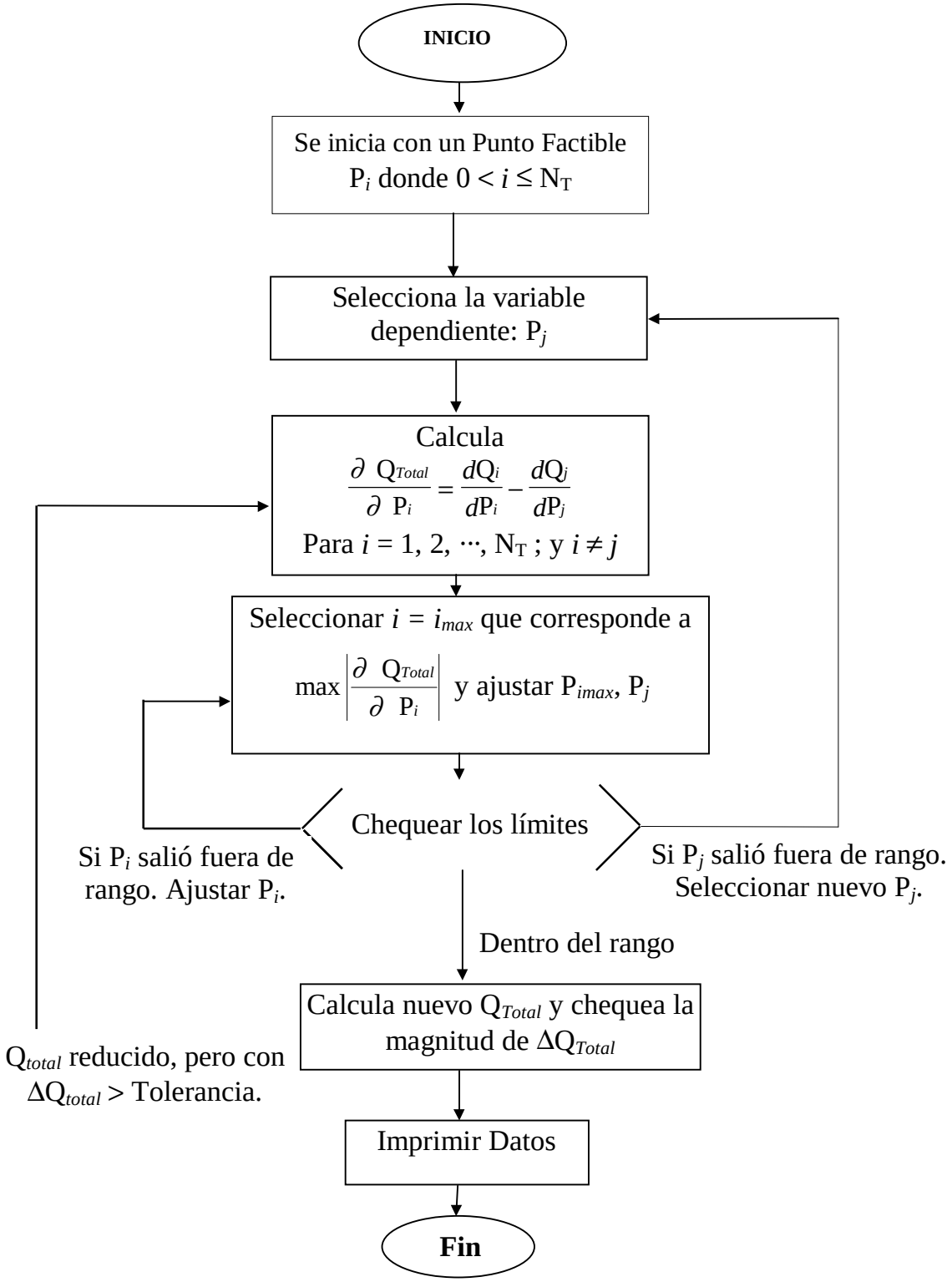


Figura 3.2: Método del Gradiente de primer orden

Este método de optimización es uno de los algoritmos más comunes utilizados para el problema de minimización del caudal total; y además, el método es relativamente fácil de implementar. Sin embargo, requiere de un gran número de iteraciones para llegar a obtener un caudal mínimo razonable, debido a que con frecuencia llega a un óptimo local sin encontrar la solución global, lo que conlleva a la utilización de mayores recursos computacionales debido al tiempo de procesamiento creciente con el consiguiente aumento de la complejidad del problema.

3.3 MÉTODO DE OPTIMIZACIÓN DE BÚSQUEDA EXHAUSTIVA

En este algoritmo, aplicado al problema de minimización del caudal, se requiere dividir el espacio de búsqueda en pequeños subespacios monótonos, formando una CUADRÍCULA. La dimensión del espacio de búsqueda está dada por la cantidad N_T de turbinas en servicio.

Si N representa al número de partes en que se divide la potencia P_i, entonces se obtiene $\Delta P_i = \frac{P_i}{N}$, que constituye la porción de la potencia P_i con la que la misma irá aumentando hasta recorrer todos los puntos del espacio de búsqueda. En el Pseudocódigo 3.1 se describe la forma en que trabaja el Algoritmo de búsqueda exhaustiva.

Como en todos los algoritmos numéricos de optimización, el algoritmo de búsqueda exhaustiva comienza a partir de un punto inicial, que corresponde a un punto de operación factible. Además de eso, se utiliza una variable llamada Q_{ant} que comúnmente comienza con un valor inicial muy grande, y mientras el algoritmo está en ejecución almacenará el mejor resultado obtenido de la función objetivo Q_{Total}.

```
Q_ant ⇐ Número grande arbitrario;
    FOR i_1 = 0 TO i_1 < N
        P_1 = P_1 + i_1 × ΔP_1;
        FOR i_2 = 0 TO i_2 < N
            P_2 = P_2 + i_2 × ΔP_2;
                .
                .
                .
            FOR i_NT−1 = 0 TO i_NT−1 < N
                P_NT−1 = P_NT−1 + i_NT−1 × ΔP_NT−1;
                P_NT = P_D − Σ_{i=1}^{N_T−1} P_i
                IF (P_NT dentro de rango) THEN
                    Q ← 0;
                    FOR i = 1 TO i ≤ N
                        Q_i = Q_i (P_i);
                        Q = Q + Q_i;
                        IF (Q < Q_ant) THEN
                            Q_ant = Q;
                        END IF
                    END IF
            END FOR
                .
                .
                .
        END FOR
    END FOR
```

Pseudocódigo 3.1: Algoritmo de Búsqueda Exhaustiva.

En este algoritmo la precisión en los resultados está dado por la porción ΔP_i. Cuando se desea mayor precisión para las soluciones, se disminuye el valor de ΔP_i lo que aumenta el número de puntos en el espacio de búsqueda y los subdominios en que fue dividido el espacio de búsqueda se vuelven más pequeños. Por consiguiente, a pesar de que el algoritmo de búsqueda exhaustiva sea mucho más fácil de implementar que el Método del Gradiente, invierte más tiempo de procesamiento para llegar a buenas soluciones, debido a que la complejidad del problema aumenta con el número de turbinas en servicio (N_T), y con la precisión deseada para los resultados.

3.4 COMPARACIÓN ENTRE AMBOS MÉTODOS DE OPTIMIZACIÓN

En esta sección describiremos brevemente las principales ventajas y desventajas de los métodos vistos anteriormente. El objetivo de analizar las diferencias de ambos métodos de optimización es comprender la razón del porque actualmente se combinan diferentes métodos de optimización, aprovechando las buenas cualidades de ambos, para la aplicación del algoritmo híbrido resultante en complejos problemas tecnológicos.

En el método del Gradiente el mejor valor encontrado se encuentra en la vecindad del punto inicial de búsqueda y generalmente no coincide con el óptimo global buscado sino que corresponde a un óptimo local. Otra desventaja que presenta este tipo de método de optimización, es que no hay un claro criterio de parada. Normalmente, el método de búsqueda del gradiente continúa la búsqueda hasta que el procedimiento de búsqueda encuentre un punto en el cual el valor del gradiente sea menor que un número $\varepsilon > 0$, suficientemente pequeño, o bien ya no se encuentre una "ganancia" significativa en la función objetivo o hasta que el número de iteraciones sobrepase algún límite numérico suficientemente alto.

Aunque el método del gradiente descrito anteriormente puede ser refinado utilizando el método del gradiente de segundo orden, en el cual ya se tienen en cuenta los términos de segundo orden de la serie de Taylor [4, sección 3.5], el algoritmo requiere, un gran número de iteraciones para llegar a buenos resultados.

En el algoritmo de búsqueda exhaustiva el espacio de búsqueda se divide en pequeños espacios monótonos, formando un cuadriculado. Además de su sencilla implementación [24, Cap. 1], tiene las siguientes propiedades:

- la búsqueda comienza a partir de un punto inicial;
- la estrategia de búsqueda está basada en la comparación del punto analizado con la mejor solución anterior.

La ventaja que presenta el algoritmo de búsqueda exhaustiva es que se analizan todos los puntos del dominio de definición encontrándose un óptimo que generalmente estará más próximo al óptimo global buscado: Sin embargo, se requiere de grandes recursos computacionales a medida que la precisión deseada aumenta, y a medida que aumenta también la dimensión del espacio de búsqueda, invirtiendo más tiempo de computación que el requerido por el método del Gradiente.

3.5 MÉTODO NUMÉRICO UTILIZADO

Para problemas de elevada complejidad, como el problema de minimización presentada en este trabajo, se combinan las ventajas de métodos arriba presentados. El dominio en el cual el método de búsqueda exhaustiva realizará el análisis global, se divide en subespacios, formando una CUADRÍCULA. Aquí también la dimensión del espacio de búsqueda es igual al número N_T de turbinas en servicio.

En la CUADRÍCULA, los puntos son analizados sistemáticamente, y se consideran sólo aquellos puntos que atienden a la restricción **(3.2)** del problema. En la **Figura 3.3** se presenta un flujograma que describe la forma en que el mencionado método numérico trabaja.

En el procedimiento **Lee_rango** del flujograma de la **Figura 3.3**, se lee los dos valores extremos de la potencias P_i que se tienen como datos, para los cuales hay datos de caudal. Las variables utilizadas para asignar los valores máximos y mínimos son: $Psup_i$ y $Pinf_i$, respectivamente.

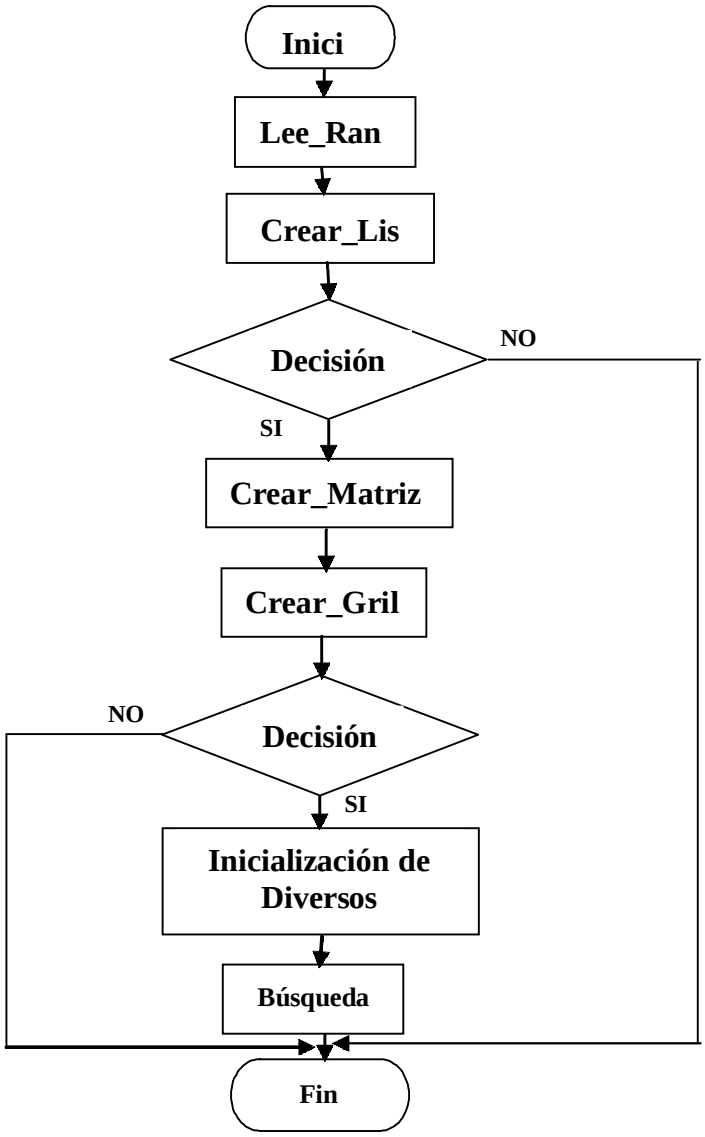


Figura 3.3: Flujograma general del Método Numérico utilizado.

Luego, en **Crear_lista**, se realiza la lectura de disponibilidad de máquinas, para que el algoritmo detecte que máquinas se encuentran en servicio. Una vez formada la lista de disponibilidad de máquinas, se realizan las siguientes preguntas:

1. Con las máquinas disponibles, ¿es posible atender la demanda de potencia?.
2. Con las unidades disponibles, ¿la potencia demandada es muy reducida?.

Computacionalmente, las preguntas hechas arriba se realizan en un procedimiento llamado **Decisión 1**, si las respuestas a las preguntas anteriores son negativas entonces el algoritmo termina la ejecución del programa.

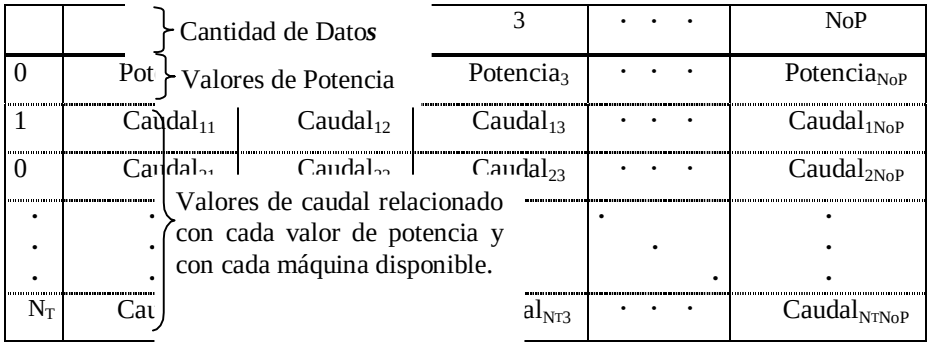


Figura 3.4: Matriz de datos interpolados con respecto al salto h.

Normalmente, el valor del parámetro h (salto disponible) no coincidirá con las alturas contenidas en el "*index-test*". Por lo tanto, en estos casos se realiza una interpolación con respecto a la altura h, para obtener datos de potencia y caudal relacionados con esa altura. Esta operación de interpolación se realiza en el procedimiento **Crear_Matriz_F**, obteniéndose una matriz de datos como muestra la **Figura 3.4**.

Como los valores de potencia que se encuentran en la Matriz de datos interpolados no están uniformemente espaciados, en el procedimiento **Crear_Grilla** se obtiene mediante interpolación, una matriz **G** (ver **Figura 3.5**) con datos de caudales correspondientes a potencias que están uniformemente espaciadas dentro del rango de operación de las máquinas.

Para ello se divide el rango de operación en un número de espacios iguales, formando una CUADRÍCULA. El número utilizado para dividir en porciones iguales el rango de operación de las unidades generadoras, correspondientes a cada dimensión, recibe el nombre de *intensidad de búsqueda*.

	1	2	3	· · ·	NoP
0	Po		encia_3^*	· · ·	$\text{Potencia}_{\text{NoP}}^*$
1	Ca		dal_{13}^*	· · ·	$\text{Caudal}_{1\text{NoP}}^*$
0	Caudal_{21}^*	Caudal_{22}^*	Caudal_{23}^*	· · ·	$\text{Caudal}_{2\text{NoP}}^*$
·	·			·	·
·	·			·	·
·	·			·	·
N_T	Cau		$\text{dal}_{\text{NT3}}^*$	· · ·	$\text{Caudal}_{\text{NTNoP}}^*$

Valores de Potencia espaciados uniformemente

Valores de Caudal espaciados uniformemente.

…on potencias espaciadas uniformemente.

Sabiendo que N es el número con el cual se divide en porciones iguales el rango de operación (Rango de operación: entre $P_{i\text{sup}}$ y $P_{i\text{inf}}$) de las turbinas, se tiene lo siguiente:

$$\textit{intensidad de búsqueda} = N + 1. \qquad \textbf{(3.8)}$$

Cuando se introduce como parámetro un valor de h para el cual no hay datos de potencia y caudal, entonces el algoritmo termina la ejecución. El chequeo correspondiente se realiza en el procedimiento **Decisión 2**.

En el procedimiento **Búsqueda** se realiza la optimización propiamente dicha. Para una mejor comprensión de la manera en como trabaja el proceso de optimización, se recurrirá a un ejemplo.

Sea N = 4 que corresponde al número de subespacios en que se dividirá el rango de operación de cada unidad generadora. Por consiguiente, la *intensidad de búsqueda* = 5. También se tienen 2 máquinas en servicio (por consiguiente $N_T = 2$). La mínima potencia que ambas máquinas pueden generar es igual a 300 MW, y la máxima potencia capaz de generar es de 700 MW. De esta forma, luego de utilizar la *intensidad de búsqueda* se tiene la siguiente matriz **G** de datos interpolados respecto a potencias uniformemente espaciadas:

	1	2	3	4	5
0	300	400	500	600	700
1	Q_{11}	Q_{12}	Q_{13}	Q_{14}	Q_{15}
2	Q_{21}	Q_{22}	Q_{23}	Q_{24}	Q_{25}

Figura 3.6: Matriz **G** formada para el ejemplo presentado.

Se realiza una búsqueda sistemática (de abajo hacia arriba y de izquierda a derecha) de puntos en el espacio de búsqueda. Todos los puntos que se encuentran en la franja sombreada del espacio de búsqueda cumplen con la siguiente condición:

$$\frac{G[0,i+1]-G[0,i]}{2} \qquad \textbf{(3.9)}$$

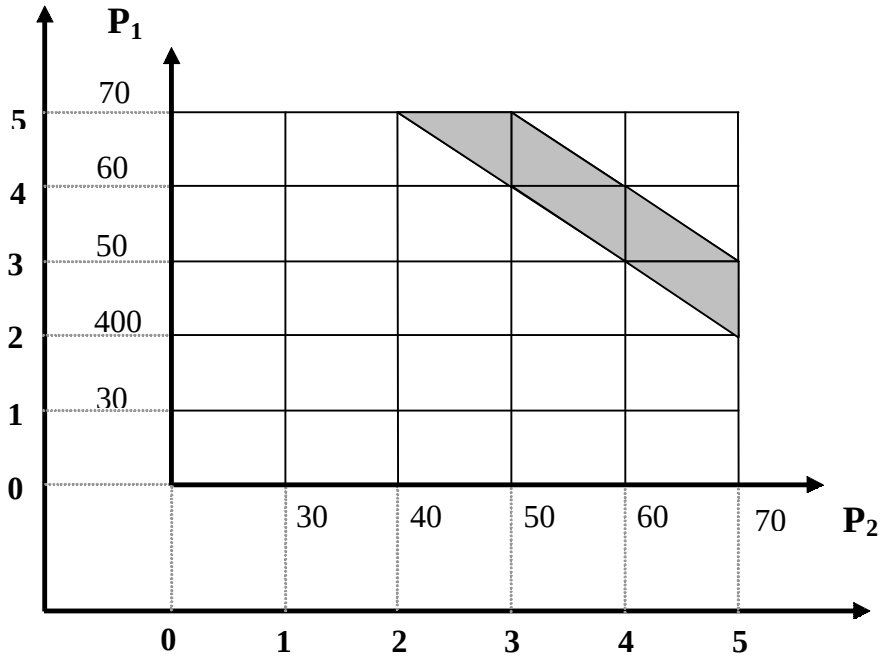


Figura 3.7: Representación del espacio de búsqueda.

Los puntos que se encuentran dentro de la franja sombreada (ver **Figura 3.7**), son los puntos prometedores sobre los cuales se aplica el método del Gradiente. La manera

en como trabaja el procedimiento **Búsqueda**, luego de que el espacio de búsqueda esté dividido en subespacios monótonos, se describe en la Figura 3.8 en donde se muestra un Flujograma del procedimiento **Búsqueda**.

Como se puede ver en la **Figura 3.8**, el resultado del Caudal Q_{Total} obtenido por el método del Gradiente se compara con el caudal Q_{min} que representa el menor caudal obtenido hasta el momento.

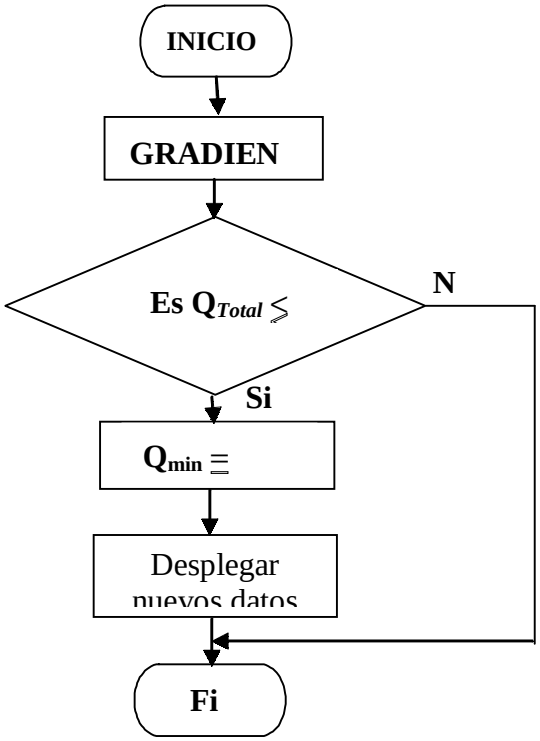


Figura 3.8: Flujograma de la llamada al Gradiente.

En la **Figura 3.9** se muestra el Flujograma del método del Gradiente. El algoritmo comienza calculando P_t (generalmente $P_t \neq P_D$) que es igual a la suma de las potencias P_i que cumplen con la condición **(3.2)**. A cada valor de potencia P_i calculada le corresponde un valor de caudal Q_i que se obtiene del "*index-test*".

Los nuevos puntos obtenidos al utilizar el gradiente pueden encontrarse fuera de rango. Para ello, se utiliza la suma de los gradientes calculados para compensar equitativamente el valor de la potencia en dichos nuevos puntos de tal manera que

nuevamente pueda cumplirse con la condición **(3.2)**. Luego, se calcula el nuevo caudal total Q_{Total} a partir de los nuevos valores de potencia obtenidos.

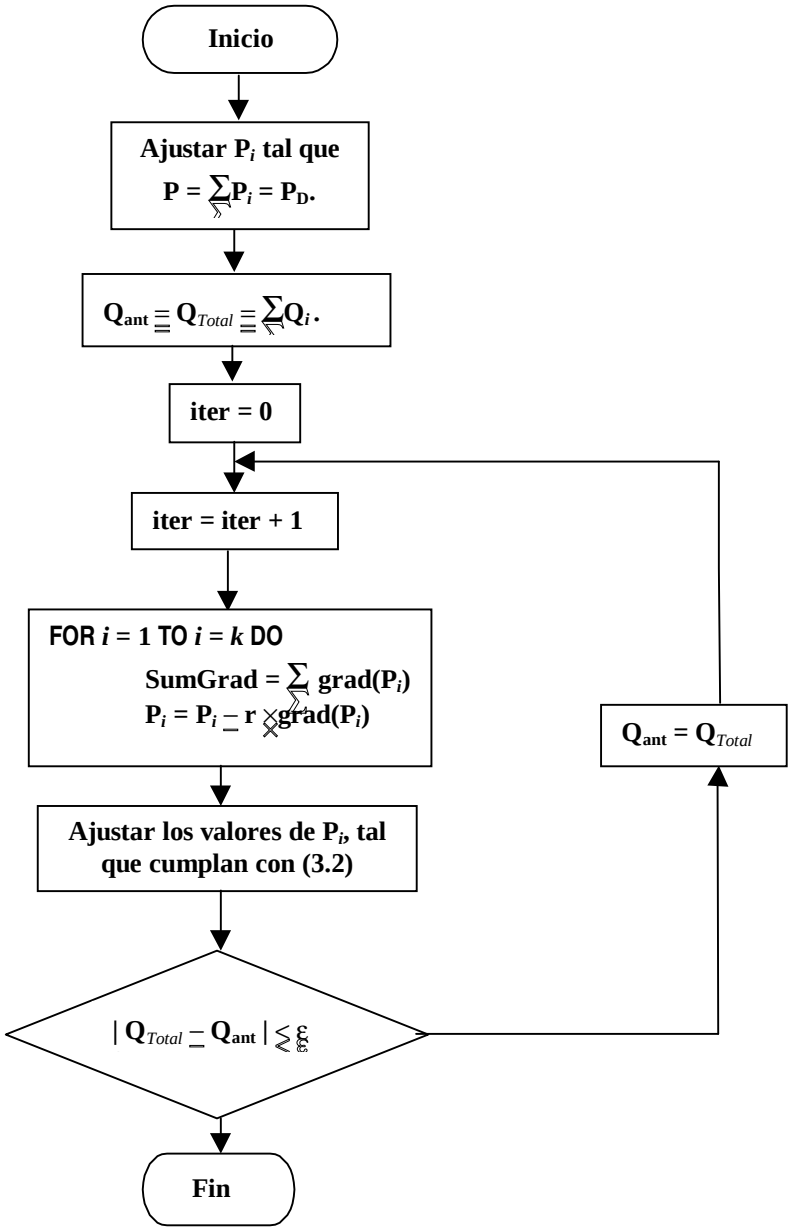


Figura 3.9: Flujograma del método del Gradiente.

Para las N_T unidades generadoras disponibles se calcula la suma de los gradientes de cada potencia P_i, y con los valores obtenidos de los gradientes de cada turbina se determinan los nuevos puntos P_i (como muestra la **Figura 3.9**). Para el cálculo de los nuevas potencias se utiliza un factor de relajación, que en la **Figura 3.9** es llamado "**r**", que permite usar solo una parte del valor numérico del gradiente con el fin de ir avanzando en forma monótona hasta el valor óptimo buscado.

El criterio de parada considerado compara el nuevo valor de caudal Q_{Total} con el caudal Q_{ant} verificando si el valor absoluto de la diferencia entre ambos es menor que un valor ε ($\varepsilon > 0$) arbitrariamente pequeño.

En resumen, el método exhaustivo se utiliza para definir los puntos del espacio de búsqueda, recorriendo cada subespacio, optimizando localmente los puntos prometedores mediante el uso del método del Gradiente.

CAPÍTULO 4: ALGORITMOS GENÉTICOS

En este capítulo se dará una definición de los Algoritmos Genéticos (AG), de donde provienen, y cuales son las principales características que lo diferencian de los demás métodos numéricos tradicionales de optimización. Se describirán también los principales operadores utilizados en los Algoritmos Genéticos. Se describirá el concepto del *esquema*, con el cual se puede realizar un modelo matemático del comportamiento del AG y comprender mejor el funcionamiento del mismo.

4.1 CONCEPTO DE LOS ALGORITMOS GENÉTICOS

Los Algoritmos Genéticos son algoritmos de optimización basados en los mecanismos naturales de Selección y Genética [24]. Los Algoritmos Genéticos son las técnicas de optimización más difundidas y estudiadas de la Computación Evolucionaria [21, 24], debido a su flexibilidad, relativa simplicidad de implementación, y eficacia en realizar búsqueda global en problemas complejos.

De una manera general, la evolución natural implementa mecanismos adaptativos de optimización que, lejos de ser una forma de búsqueda aleatoria, con seguridad utilizan la aleatoriedad como herramienta de búsqueda. Es decir, los mecanismos de búsqueda en la naturaleza son procesos *inteligentes* pero no determinísticos, que los Algoritmos Genéticos intentan imitar.

A finales del siglo pasado, Charles Darwin tenía formulado la Teoría de la Evolución de las Especies [39], pero sólo recientemente se intentó idealizar un modelo matemático del proceso evolutivo.

En los años 60, John Holland, sus colegas y estudiantes de la Universidad de Michigan [24] comenzaron a definir las bases de los algoritmos de optimización de inspiración genética. El objetivo de sus investigaciones fue doble: (1) resumir y explicar rigurosamente los procesos adaptativos de los sistemas naturales, y (2) diseñar paquetes de software que simulen los mecanismos evolutivos de los sistemas naturales. Las primeras investigaciones realizadas culminaron con la publicación del libro *Adaptation in Natural and Artificial Systems* [40], presentando importantes descubrimientos para la Ciencia de los sistemas naturales y artificiales.

El tema central de las investigaciones en Algoritmos Genéticos ha sido la *robustez*, es decir la capacidad del algoritmo para optimizar satisfactoriamente una amplia gama de problemas y obtener en todos los casos muy buenos resultados. Las implicaciones de la *robustez* para los sistemas artificiales (algoritmos) son múltiples: si los sistemas artificiales llegaran a ser más robustos, los costos de rediseños se reducirán o incluso se eliminarán, es decir, no se necesitará acondicionar totalmente el algoritmo para un nuevo conjunto de parámetros de entrada para un nuevo problema. Además, si se llegan a niveles altos de adaptación, los sistemas robustos existentes pueden desempeñar mucho mejor sus funciones y durante mucho más tiempo [24].

Actualmente, los AG se aplican a problemas tecnológicos de diversas áreas de la ciencia y son utilizados para encontrar óptimos globales de funciones matemáticas que modelan sistemas físicos complejos. Por consiguiente, el Algoritmo Genético puede definirse como un algoritmo de optimización en el cual una población de posibles soluciones evoluciona utilizando operadores probabilísticos que imitan de alguna manera el proceso evolutivo de las especies, de modo que exista una tendencia de que los individuos representen soluciones cada vez mejores a medida que la población avanza de una generación a la siguiente [21].

4.2 DESEMPEÑO DE LOS MÉTODOS NUMÉRICOS TRADICIONALES

Una gran parte de los problemas tecnológicos actuales pueden ser formulados como problemas de búsqueda y optimización, mediante la representación de sistemas físicos por medio de funciones matemáticas. Las variables independientes, que determinan el valor numérico de la función matemática, representan los factores que actúan sobre el sistema físico modificando su desempeño.

A su vez, el comportamiento del sistema físico considerado puede estar sujeto a restricciones. Esto significa que de todos los valores posibles que puede adquirir la función matemática dentro del dominio de definición de la función sólo algunos valores cumplen con las condiciones de comportamiento del sistema.

Matemáticamente hablando, el conjunto de todos los puntos[1] que definen todos los valores permitidos para la función matemática, que modela al sistema físico analizado, constituye el espacio de búsqueda de la función.

En términos generales, dada la función $f \mid \Re^n \rightarrow \Re$ que modela un sistema físico y un espacio de búsqueda $S \subseteq \Re$, entonces el problema de optimización puede ser formulada de la siguiente manera:

$$\text{optimizar:} \qquad f(\boldsymbol{x}) \mid \boldsymbol{x} \in S \qquad \textbf{(4.1)}$$

Si $\boldsymbol{x}^*$ ($\boldsymbol{x}^* \in S$) constituye la variable independiente para el cual se obtiene el valor óptimo de $f(\boldsymbol{x})$, el problema de optimización formulado en **(4.1)** consiste en resolver el siguiente problema de búsqueda:

$$\text{encontrar:} \qquad \boldsymbol{x}^* \mid f(\boldsymbol{x}^*) \geq f(\boldsymbol{x}), \forall\, \boldsymbol{x} \in S \qquad \textbf{(4.2)}$$

[1] En adelante, el punto estará referido a un vector cuyas componentes corresponden a las variables independiente que definen el valor numérico de la función matemática.

La función $f(\cdot)$ puede ser derivable o no, unidimensional o multidimensional, y el espacio de búsqueda S puede ser continuo o discreto, finito o infinito, cóncavo o convexo.

Al respecto, el problema de la búsqueda de los puntos extremos de una función matemática ya ha sido ampliamente estudiado [41], y en la actualidad se cuentan con varios métodos numéricos de probada eficacia. La bondad de un método de optimización está determinada principalmente por la capacidad que éste posea para encontrar el óptimo de las funciones en una amplia gama de problemas. Esta característica determina la *robustez* del método.

Los métodos de búsqueda de optimización tradicionales pueden agruparse en tres tipos principales: los métodos basados en el cálculo, esquemas enumerativos y algoritmos de búsqueda aleatoria.

a) Métodos basados en el Cálculo: Estos métodos tiene una metodología de búsqueda que requiere de la información proporcionada por el gradiente de la función analizada. Estos métodos, a pesar de ser rápidos, presentan dos inconvenientes principales. Primero, generalmente obtienen como resultado del proceso de búsqueda un óptimo local, es decir, el máximo o mínimo situado en la vecindad del punto a partir del cual se inició la búsqueda, el cual generalmente no corresponde a un óptimo global. En segundo lugar, dependen de que la función sea continua y derivable en su dominio de definición.

Existen dos tipos de métodos basados en el cálculo [24]: los *indirectos* y los *directos*. Los métodos *indirectos* buscan el óptimo local de una función matemática de una o varias variables mediante la resolución de un sistema de ecuaciones no lineales iguales a cero, formada por las derivadas parciales de dicha función. Esta es una generalización multidimensional de la noción de Cálculo elemental para búsqueda de puntos óptimos.

Por otro lado, están los métodos *directos* que buscan el punto óptimo avanzando en la dirección del gradiente de la función en el punto considerado. Dicho avance es proporcional a una porción del valor numérico del gradiente Esto corresponde a la noción del método de búsqueda llamado *hill-climbing* [24, 42 Cap.4]: en el cual, por analogía, los valores máximos de la función están asociados con una "*colina*" o "*cerro*", y el gradiente de la función en el punto considerado con un "*sendero cuesta arriba*". Por consiguiente, en estos métodos se trata de recorrer la mayor distancia posible en cada paso, por el "*sendero cuesta arriba*", para llegar más rápidamente a la cima de la "*colina*".

Los métodos basados en el cálculo han sido mejorados, extendidos y revisados con el tiempo. Pero aún poseen una baja *robustez*. Esto se debe a que los métodos presentan un muy buen desempeño cuando la función a optimizar es unimodal[2], pero cuando la función es multimodal el desempeño de estos métodos decrece considerablemente.

b) Esquemas Enumerativos: Buscan el óptimo analizando el valor que adquiere la función objetivo en cada punto de un espacio de búsqueda discretizado. Es decir, el espacio de búsqueda está dividido en subespacios monótonos. Cuanto menor es el tamaño de estos subespacios mayor es la cantidad de puntos a ser analizados en el espacio de búsqueda, y de esta manera aumenta la precisión de los resultados obtenidos. Este método de búsqueda de "punto por punto", si bien es simple y fácil de implementar, tiene el inconveniente de que puede llegar a invertir grandes tiempos de procesamiento a medida que aumenta la precisión deseada.

[2] Se dice que una función $f(x):S \in \Re^n \rightarrow \Re$ es unimodal cuando tiene un solo óptimo global en el dominio de definición de la función.

Estos métodos numéricos de optimización son muy aplicables a funciones discretas o funciones multimodales. Debido a que la ventaja que posee es que analiza todos los puntos del espacio de búsqueda, evitando encontrar óptimos locales. Sin embargo, al aumentar la complejidad del problema (dado por la precisión y el número de variables independientes de la función) los recursos computacionales requeridos aumentan exponencialmente.

c) **Algoritmos de Búsqueda Aleatoria:** Estos métodos de optimización carecen de una estrategia o dirección de búsqueda. La metodología básica de funcionamiento es generar aleatoriamente puntos pertenecientes al espacio de búsqueda considerado, y comparar los valores que adquiere la función objetivo con los obtenidos anteriormente; sin embargo, han logrado creciente popularidad como algoritmos de búsqueda que tratan de salvar los defectos existentes en los Métodos basados en el Cálculo y en los Esquemas Enumerativos. Pero aún así, estos algoritmos no deben ser considerados robustos debido a su técnica de búsqueda, porque a la larga no son mejores que los Esquemas Enumerativos.

De lo expuesto anteriormente se concluye que los métodos tradicionales de optimización no son robustos. Esto no implica que los métodos tradicionales no sean útiles, puesto que los métodos arriba descriptos así como las incontables combinaciones híbridas de los mismos [24, Cap.1] han sido utilizado en numerosas aplicaciones.

Sin embargo, para problemas mucho más complejos ya es necesario la utilización de otras técnicas de optimización más robustas, debido a que las técnicas tradicionales ya no presentan un buen desempeño.

Al respecto, el Algoritmo Genético representa una interesante técnica de búsqueda de óptimos globales, debido a que utiliza la aleatoriedad como herramienta para guiar la exploración de nuevas zonas del espacio de búsqueda que representen soluciones cada vez mejores.

4.3 CARÁCTERÍSTICAS DE LOS ALGORITMOS GENÉTICOS

A continuación se describirán las características principales que vuelven atractivos a los Algoritmos Genéticos para ser utilizados en problemas de optimización de gran envergadura [24]:

a) Los AG trabajan con la codificación de las variables independientes de la función matemática a optimizar. Las variables independientes de la función se codifican utilizando generalmente un sistema binario {1,0}, formando una cadena de caracteres con dicho sistema numérico. En la literatura, a la cadena de caracteres, que contiene la representación binaria de todas las variables independientes de la función, se la denomina *cromosoma*.

b) En los Algoritmos Genéticos la búsqueda se realiza a partir de una población de posibles soluciones (puntos del espacio de búsqueda), y no a partir de un solo punto. La búsqueda así realizada tiene un paralelismo implícito [24]. En Algoritmo Genético los puntos del espacio de búsqueda son denominados *individuos*, haciendo una analogía con la naturaleza en la que cada uno de los integrantes de una especie también es denominado un individuo.

c) Solo necesitan del valor numérico de la función matemática para guiar la búsqueda, a diferencia de los métodos tradicionales que utilizan el gradiente u otras informaciones adicionales para encontrar el punto óptimo.

d) Utilizan reglas de transición probabilística (operadores probabilísticos de búsqueda), y no determinísticas.

Por consiguiente, el AG sólo necesita del valor de la función matemática para explorar nuevas regiones del espacio de búsqueda, y realiza la búsqueda del punto óptimo a partir de una determinada cantidad de puntos aplicando sobre los mismos los operadores probabilísticos. Estas cualidades hacen del AG una técnica de búsqueda robusta permitiendo un muy buen desempeño ante un número considerable de problemas.

4.4 OPERADORES DEL ALGORITMO GENÉTICO

Los mecanismos de búsqueda de un Algoritmo Genético son notablemente simples, pero poderosos en la optimización de funciones complejas. Por consiguiente, la simplicidad de operación y la *robustez* de su aplicación son las dos principales razones por las que el Algoritmo Genético es muy requerido en numerosos problemas tecnológicos.

En esta sección, se describirán los operadores probabilísticos (llamados también: *operadores genéticos* [24]) utilizados por el AG, para realizar la optimización de una función matemática.

4.4.1 OPERADORES PROBABILÍSTICOS

Un Algoritmo Genético está compuesto por tres operadores probabilísticos: el operador de Selección, de Cruzamiento y de Mutación.

a) El operador de Selección: simula el proceso de selección natural, en el que el más fuerte tiene mayor capacidad de supervivencia. En un AG la capacidad de supervivencia de un *individuo* está relacionado con el valor numérico de la función objetivo (*fitness*).

Como el Algoritmo Genético es un algoritmo de carácter iterativo al igual que los métodos numéricos de optimización, se aplica en cada iteración, sobre una población de N *individuos*, el operador de Selección para escoger los mejores N *individuos* para la siguiente iteración.

Por consiguiente, en los N *individuos* escogidos para la siguiente iteración, pueden encontrarse dos o más *individuos* idénticos debido a que los *individuos* con bajos *fitness* tienen menos posibilidades de ser escogidos, en cambio, los que poseen altos *fitness* tienen más chances de ser frecuentemente seleccionados.

Quizás la forma más fácil de comprender al operador de Selección consista en crear computacionalmente una ruleta [24 Cap.1], en la que él número de partes en que se divida la ruleta es igual a N, y el tamaño de cada parte es proporcional al *fitness* de cada *individuo*.

Es de esperar que al hacer "girar la ruleta"[3] tantas veces como *individuos* existan en la población, se obtenga una población de N *individuos* con mayor cantidad de *individuos* con altos *fitness*. A continuación, se presenta el pseudocódigo del operador de Selección.

[3] Computacionalmente hablando, "girar la ruleta" en Algoritmo Genético significa obtener aleatoriamente un número comprendido entre el cero y la sumatoria de los *fitness* de la población.

RAND ← número aleatorio entre 0.0 y 1.0;

Suma ← 0;

SumaParcial ←RAND × $\sum_{i=1}^{N} f_i$ /*RAND × Sumatoria de *fitness**/

FOR *j* = 1 **TO** *j* = N /*N, tamaño de la población*/

Suma ← Suma + $\sum_{i=1}^{N} f_i$

IF (Suma ≥ SumaParcial**) THEN** break;

END FOR

RETURN *j* ; /*Devuelve índice del *individuo**/

Pseudocódigo 4.1: Operador de Selección.

b) El operador de Cruzamiento: Una vez que los *individuos* fueron seleccionados se aplica a cada par de *individuos* escogidos, el operador de Cruzamiento. Este proceso es equivalente a la búsqueda en la vecindad del par de puntos seleccionados.

En primer lugar, se escoge aleatoriamente un punto de corte que se aplicará al par de *cromosomas* de los *individuos* seleccionados. Luego, los caracteres más significativos, a partir del punto de corte, se conservan en sus posiciones relativas en el nuevo par de *cromosomas*, y los caracteres restantes son intercambiados de los *cromosomas* progenitores a los nuevos dos *cromosomas* obtenidos. Como ejemplo, se consideran dos *cromosomas* A_1 y A_2 de un par de *individuos* sobre los que se aplicará el operador de Cruzamiento para obtener dos nuevos *cromosomas*:

$$A_1 = 0\ 1\ 1\ 0\ |\ 1$$

$$A_2 = 1\ 1\ 0\ 0\ |\ 0$$

Donde el símbolo "|" indica el punto de corte. Los caracteres a la izquierda del punto de corte son los más significativos. Por lo tanto, los caracteres más significativos permanecen en sus posiciones correspondientes en los nuevos *cromosomas*, mientras que los caracteres a la derecha del punto de corte son cruzados de los *cromosomas* progenitores, como muestra el ejemplo:

$$A'_1 = 0\ 1\ 1\ 0\ 0$$

$$A'_2 = 1\ 1\ 0\ 0\ 1$$

En el ejemplo anterior, A'_1 y A'_2, representan a los dos nuevos *cromosomas* que resultan de aplicar el operador de Cruzamiento al par de *cromosomas* progenitores. En el Pseudocódigo 4.2 se describe al operador de Cruzamiento. En dicho Pseudocódigo, se tiene que ℓ representa a la longitud del *cromosoma*, jcross constituye el punto de corte, y los vectores child1[ ] y child2[ ] equivalen a la representación binaria de una pareja de puntos.

```
IF (Cruzamiento permitido de acuerdo a una probabilidad) THEN
        jcross ← número aleatorio entre 1 y ℓ – 1;
ELSE
        jcross ← ℓ.
END IF
FOR  j = 1 TO j = jcross DO
        child1[ j ] ← transmitir código genético del progenitor 1.;
        child2[ j ] ← transmitir código genético del progenitor 2.;
END FOR
IF ( jcross es distinto de ℓ ) THEN
        FOR  j = jcross + 1 TO j = ℓ DO
                child1[ j ] ← transmitir código genético del progenitor 2.
                child2[ j ] ← transmitir código genético del progenitor 1.
        END FOR
END IF
```

Pseudocódigo 4.2: Operador de Cruzamiento.

c) **El operador de Mutación:** el proceso de mutación es básicamente una búsqueda aleatoria. Se selecciona aleatoriamente una posición específica dentro de un *cromosoma*, para cambiar luego el valor contenido en dicha posición.

Como en la naturaleza, la probabilidad de que ocurra mutación en los individuos de una especie es muy pequeña, en condiciones normales de vida, en el Algoritmo Genético se trata de representar lo mismo con un valor de probabilidad muy baja para la aplicación del operador de Mutación.

4.4.2 IMPLEMENTACIÓN BÁSICA DEL ALGORITMO GENÉTICO

El primer paso para la aplicación de los Algoritmos Genéticos a un problema determinado (función que se quiere optimizar) es la representación de cada posible punto $\boldsymbol{x}$ del espacio de búsqueda mediante una secuencia de símbolos A de un dado alfabeto finito. Generalmente, el alfabeto utilizado, para la representación de la variable independiente $\boldsymbol{x}$, es el sistema binario {1,0}, el que también es utilizado en el presente trabajo.

Cada punto del espacio del espacio de búsqueda, que define el valor de la función a optimizar, se denomina *individuo*. En la mayor parte de los Algoritmos Genéticos, cada *individuo* está constituido de un único *cromosoma* en el que están representados todas las variables de la función objetivo [21]. Una vez definida la representación binaria para los puntos del problema sobre el cual se trabajará, se genera en forma aleatoria un conjunto de posibles soluciones (puntos del espacio de búsqueda).

En un Algoritmo Genético un *individuo* está representado por una estructura de datos, la que está formada por las siguientes variables**:**

- la codificación de las variables independientes que definen el valor de la función matemática a optimizar. A la codificación de las variables independientes llamamos *cromosoma*;
- los valores reales de dichas variables independientes.
- El valor de la función objetivo, $f(\boldsymbol{x})$. Con el valor numérico de $f(\boldsymbol{x})$ se tiene una medida de que tan bien adaptado al ambiente se encuentra el *individuo* correspondiente; o sea, cuanto mayor es el valor de la función objetivo mayores son las chances del *individuo* de sobrevivir en el ambiente y reproducirse,, pasando parte del buen material genético a generaciones posteriores, haciendo una analogía con la biología.

En la mayoría de las aplicaciones del Algoritmo Genético, la población inicial es generada en forma aleatoria, aplicando sucesivamente sobre la población los tres operadores probabilísticos, descriptos en la sección anterior. Para encontrar una buena solución, es importante que la población contenga suficiente variedad genética de forma a evitar que el AG se estanque prematuramente en óptimos locales. Este criterio está basado en la Teoría de la Selección Natural [21].

Los Algoritmos Genéticos son algoritmos iterativos, y en cada iteración la población es modificada. Cada iteración de un Algoritmo Genético se denomina *generación*, por consiguiente cada *individuo* de la población actual será un "descendiente" de los *individuos* de la población anterior. Designando cada *generación* por un índice *t*, el procedimiento de búsqueda del Algoritmo Genético se ilustra en el Pseudocódigo 4.3.

Como se puede observar en el Pseudocódigo 4.3, una vez inicializado la población se realiza una evaluación de la misma. La evaluación consiste en determinar el valor numérico de la función objetivo (*fitness*) de cada *individuo*. Además de lo anterior, en la evaluación se calcula el valor medio de los *fitness* de la población, la sumatoria de *fitness* y la varianza de la población. Todos estos valores sirven como indicador para observar la evolución del algoritmo desde una generación a la otra.

```
t ← 0;
Iniciar Población ( t );
Evaluar Población ( t );
DO WHILE ( No se cumpla Criterio de Parada )
        t ← t + 1;
        Seleccionar Población ( t ) a partir de Población ( t – 1 );
        Cruzar y mutar Población ( t );
        Evaluar Población ( t );
END DO
```

Pseudocódigo4.3: Algoritmo Genético.

La varianza indica que tan dispersos se encuentran los valores de la función en la población con respecto al valor medio. El valor obtenido de la varianza se puede utilizar como criterio de parada del algoritmo, debido a que cuando la misma adquiere valores muy pequeños a medida que transcurren las generaciones, esto significa que la gran mayoría de los individuos llegaron a los mismos valores altos de adaptación en la población. Otro criterio de finalización del algoritmo, consiste generalmente en una máxima cantidad de iteraciones prefijada.

4.5 FUNDAMENTO MATEMÁTICO DE LOS AG

Esta sección está orientada a dar un enfoque teórico-matemático a los operadores probabilísticos del AG, con el fin de comprender mejor la forma en como éstos trabajan en el AG y poder predecir de alguna manera sus resultados. La herramienta introducida para poder realizar este análisis teórico se denomina ***esquema***, y es conceptualizada en la siguiente sección.

4.5.1 CONCEPTO DE ESQUEMA

Un ***esquema*** podría ser pensado como un "molde" representando uno o un conjunto de *cromosomas*. Considerando un alfabeto binario, utilizado en el presente trabajo para la codificación de las variables independientes de la función objetivo, y un *cromosoma* de longitud[4] ℓ, se representa un ***esquema*** mediante cualquier secuencia de ℓ caracteres generados a partir del siguiente alfabeto extendido {`0`, `1`, `*`}, donde "`*`" representa un carácter "comodín" el cual puede adquirir cualquiera de los valores del alfabeto binario mencionado.

Para representar un ***esquema*** recurrimos al símbolo `*`, con el cual creamos cadenas de caracteres con el siguiente alfabeto ternario {`1`, `0`, `*`}, en lugar de utilizar el alfabeto binario {`1`, `0`}. Como ya se había adelantado, el símbolo `*` indica que esa posición puede ser ocupado por uno de los caracteres del sistema binario. En un ***esquema***, los caracteres del alfabeto binario se denominan *caracteres fijos*. Las posiciones ocupadas por los *caracteres fijos* se denominan posiciones fijas.

De esta forma, en un ***esquema*** están incluidos un determinado número de *cromosomas* cuya cantidad depende de la longitud ℓ del *cromosoma* y del número de símbolos * utilizados.

[4] La longitud ℓ de un *cromosoma* está dada por la cantidad de caracteres del alfabeto utilizado para representar las variables independientes de la función objetivo.

Por ejemplo, en el siguiente ***esquema*** `* 0 0 0 0`, están incluidos dos *cromosomas*, que pertenecen al siguiente conjunto {`1 0 0 0 0`, `0 0 0 0 0`}; otro ejemplo está dado por el ***esquema*** `* 1 1 1 *`, que representa al siguiente conjunto de *cromosomas* {`0 1 1 1 0`,`0 1 1 1 1`,`1 1 1 1 0`,`1 1 1 1 1`}.

El concepto de ***esquema*** otorga un medio conveniente, poderoso y compacto para referirnos a cualquier *cromosoma* de longitud finita correspondiente a un alfabeto determinado. El símbolo `*` es denominado un metasímbolo (un símbolo que se refiere a otros símbolos); en el Algoritmo Genético este símbolo nunca es procesado. Consiste únicamente en un dispositivo de notación que permite la descripción de todas las posibles similitudes entre *cromosomas*.

El efecto del operador de Selección sobre un ***esquema*** es fácil de interpretar, debido a que los *cromosomas* más adaptados (con *fitness* más altos) tiene más posibilidades de ser seleccionados. Por lo tanto, se espera un número creciente de muestras reproducidas para la siguiente generación, correspondientes a buenos "*cromosomas*". Sin embargo, la Selección por si sola no crea nuevos puntos en el espacio de búsqueda.

Cuando se introduce el operador de Cruzamiento, el ***esquema*** puede ser o no dividido en partes. Cuando el ***esquema*** es particionado éste queda destruido. Por ejemplo, consideramos los dos esquemas siguientes: `1 * * * 0` y `* * 1 1 *`. El primero tiene más probabilidades de ser partido debido a que los caracteres correspondientes a las posiciones fijas están muy distanciados entre si. El segundo, es relativamente menos probable que sea partido, y por lo tanto, de alguna manera asegura que este esquema será varias veces seleccionado y reproducido para la siguiente generación; esto es debido a que los caracteres correspondientes a las posiciones fijas están muy cerca uno del otro.

La Mutación es normalmente aplicada muy pocas veces, es decir su probabilidad es muy baja. Pero cuando el operador de mutación actúa, modifica un carácter del *cromosoma* cambiándolo a su valor opuesto, alterando de esta manera la adaptabilidad del *cromosoma* haciendo que su *fitness* sea alto y por consiguiente con más frecuencia seleccionado, o de adaptabilidad baja haciendo que dichos *cromosomas* sean cada vez menos seleccionados.

Por consiguiente, visto el análisis empírico anterior, podemos adelantar que los ***esquemas***, en la que los caracteres que ocupan posiciones fijas están próximas entre sí y representen *cromosomas* de alta adaptabilidad (alto *fitness*), serán reproducidos de generación en generación e irán en aumento.

4.5.2 ANÁLISIS MATEMÁTICO DE LOS OPERADORES GENÉTICOS

En la sección anterior se ha dado una explicación intuitiva de la forma en que los operadores genéticos actúan sobre una población de puntos (*individuos*). En esta sección, se extenderán los conocimientos teóricos acerca de los operadores utilizados en el Algoritmo Genético incluyendo en los análisis al ***esquema***.

De los ejemplos de ***esquemas*** dados en la sección anterior, existen un total de 3^{ℓ} ***esquemas*** por cada *cromosoma* artificial de longitud finita ℓ utilizando un alfabeto binario. Generalizando para un alfabeto de K caracteres para representar en un *cromosoma* a las variables independientes de la función objetivo a optimizar, se tiene un total de $(K + 1)^{\ell}$ ***esquemas***.

Los parámetros que definen las características de los ***esquemas*** son: el *orden* y la *longitud*.

El *orden* de un ***esquema*** está dado por el número de posiciones fijas que éste posee, y la *longitud* del ***esquema*** está definida como la diferencia entre la última posición fija y la primera del ***esquema*** correspondiente, a contar desde los caracteres menos significativos.

Por ejemplo, el *orden* de un ***esquema*** como $H_1 =$ `0 1 1 * 1 * *`, es igual a 4 porque existen 4 *caracteres fijos* en H_1, y se lo representa como $O(H_1) = 4$. El *orden* del ***esquema*** $H_2 =$ `0 * * * * * *`, es igual a 1, lo que significa que $O(H_2) = 1$.

Asimismo, la *longitud* del ***esquema*** $H_1 =$ `0 1 1 * 1 * *`, es igual a 4, porque el último *carácter fijo* ocupa la posición 7 del ***esquema***, y el primer *carácter fijo* ocupa la posición 3; por lo tanto, la diferencia entre estas dos posiciones es igual a 4, por consiguiente tenemos que $\delta\ (H_1) = 4$. Así también tenemos que para el ***esquema*** $H_2 =$ `0 * * * * * *`, se tiene que $\delta\,(H_2) = 0$, porque la primera posición fija está en 8 y la posición del último *carácter fijo* es también igual a 8.

Comenzando con el análisis teórico de los operadores probabilísticos del AG, se considera una población formada por *N individuos* (puntos del espacio de búsqueda). A su vez, se considera también que en un instante *t*, en la población, un ***esquema*** genérico *H* representa a *m cromosomas*. Luego, se aplica el operador de Selección para obtener la siguiente población en el instante $t + 1$.

Con el operador de Selección se escogen *N individuos* de acuerdo a una probabilidad proporcional al *fitness* de cada *individuo* de la población, o sea, la probabilidad de selección de un *individuo* (o *cromosoma*) está dada por la siguiente fórmula:

$$p_i = \frac{f_i}{\sum f_i} \qquad \textbf{(4.3)}$$

donde f_i indica el *fitness* o valor numérico de la función objetivo correspondiente al *cromosoma i*.

Luego de seleccionar los *individuos* de la población, se espera reproducir para la siguiente generación un total de $m(H, t + 1)$ *cromosomas* representados por el ***esquema*** *H*, en la generación $t + 1$, dado por la siguiente ecuación:

$$m(H, t+1) = m(H, t) \times \frac{N \times f(H)}{\sum f_i} \qquad \textbf{(4.4)}$$

donde $f(H)$ es el promedio de los *fitness* correspondientes al ***esquema*** *H*.

Como $\Sigma f_i / N = f_{prom}$, representa el promedio de los *fitness* de la población la ecuación anterior puede ser reescrita de la siguiente manera:

$$m(H, t+1) = m(H, t) \times \frac{f(H)}{f_{prom}} \qquad \textbf{(4.5)}$$

La ecuación anterior nos dice que la cantidad de *cromosomas* representados por un ***esquema*** genérico, *H*, crece en función al cociente entre el promedio de los *fitness* del ***esquema*** y el promedio de los *fitness* de la población. Si el promedio de los *fitness* del ***esquema*** es mayor que el promedio de los *fitness* de la población, entonces se espera un crecimiento en el número de *cromosomas* para la siguiente generación. Pero, si el *fitness* del esquema es menor que el promedio de los *fitness* de la población, el resultado es un decremento en el número de *individuos* representados por el ***esquema*** correspondiente.

Sin embargo, la sola aplicación del operador de Selección, para luego reproducir en la siguiente generación los *individuos* más adaptados, no ayuda a explorar nuevos puntos en el espacio de búsqueda. Es evidente que para explorar nuevas regiones en el espacio de búsqueda es necesario crear nuevos *individuos*.

Para crear nuevos puntos (*individuos*) en el espacio de búsqueda se utiliza el operador de Cruzamiento, el cual trabaja con cada par de *individuos* seleccionados generando un nuevo par de puntos a los que comúnmente se los llama *descendientes*, haciendo una analogía con la biología. Para determinar cuáles ***esquemas*** serán afectados por el Cruzamiento y cuales no, consideremos un *cromosoma A* de $\delta(A) = 7$, el cual está incluido en dos esquemas, H_1 y H_2:

$$A = 0\ 1\ 1\ 1\ 0\ 0\ 0$$
$$H_1 = *\ 1\ *\ *\ *\ *\ 0$$
$$H_2 = *\ *\ *\ 1\ 0\ *\ *$$

Se considera que el *cromosoma A* ha sido escogido como pareja para el cruzamiento. A continuación, se escoge aleatoriamente un número entero comprendido entre 1 y $\ell - 1$. Dicho número, aleatoriamente seleccionado, recibe el nombre de "punto de corte", p_c. En nuestro ejemplo hay 6 posibles valores numéricos para el punto de corte.

En el Algoritmo Genético, el punto de corte se utiliza para dividir al par de *cromosomas* generalmente en dos partes. Los caracteres más significativos a partir del punto de corte se mantienen en sus posiciones relativas en los *cromosomas descendientes*, y los caracteres menos significativos son intercambiados de los *cromosomas progenitores* a los nuevos *cromosomas*.

En el ejemplo que sigue, se supone que el punto de corte, escogido aleatoriamente, es igual a 4 y el cual es simbolizado por el carácter "|":

$$A = 0\ 1\ 1\ |\ 1\ 0\ 0\ 0$$
$$H_1 = *\ 1\ *\ |\ *\ *\ *\ 0$$
$$H_2 = *\ *\ *\ |\ 1\ 0\ *\ *$$

Se observa que el ***esquema*** H_1 será destruido luego del cruzamiento, porque el *carácter fijo* "1", que está en la posición 6 de H_1, y el carácter "0" que se encuentra en la posición 1 de H_1 están en lados distintos respecto del punto de corte, y por consiguiente, estos *caracteres fijos* serán separados en la siguiente generación. Por otro lado, el ***esquema*** H_2 conservará su estructura luego del cruzamiento porque tanto el *carácter fijo* "1" como el *carácter fijo* "0" de H_2 están del mismo lado del punto de corte. Por consiguiente, se puede decir que el ***esquema*** H_2 no se destruirá en la siguiente generación.

Cualitativamente se puede decir que aquellos ***esquemas*** cuyos *caracteres fijos* se encuentren en distintos lados del punto de corte, tienen más posibilidades de ser destruidos que aquellos otros ***esquemas*** cuyos *caracteres fijos* se encuentren del mismo lado del punto mencionado.

Para explicar en forma más detallada el punto de vista anterior, se observa que el ***esquema*** H_1 tiene una longitud de $\delta(H_1) = 5$. Si el punto de corte es seleccionado aleatoriamente para el ***esquema*** H_1, existen $\ell - 1 = 7 - 1 = 6$ lugares posibles. Por lo tanto, $p_d = \delta(H_1) / (\ell - 1) = 5 / (7 - 1) = 5 / 6$ corresponde a la probabilidad de que el esquema H_1 sea destruido, y la posibilidad que tiene de sobrevivir está determinado por la ecuación $p_s = 1 - p_d = 1 / 6$. Asimismo el ***esquema*** H_2, que tiene una longitud $\delta(H_2) = 1$, es destruido con una probabilidad de $p_d = \delta(H_2) / (\ell - 1) = 1 / (7 - 1) = 1 / 6$, y sobrevive con una probabilidad de $p_s = 1 - p_d = 5 / 6$.

Generalizando, tenemos que cuando el punto de corte corresponde a un valor numérico mayor que la longitud de un determinado ***esquema***, éste tiene mayores chances de sobrevivir. Pero, cuando el valor numérico del punto de corte es menor que la longitud del ***esquema***, éste tiene mayores probabilidades de ser destruido para la siguiente generación. Por consiguiente, la probabilidad de supervivencia de cualquier ***esquema*** está dado por la siguiente ecuación:

$$p_s = 1 - \frac{\delta(H)}{\ell - 1} \qquad \textbf{(4.6)}$$

Sin embargo, la ecuación anterior es válida sólo cuando se aplica el operador de Cruzamiento en cada generación. Además, el operador de Cruzamiento no se aplica todas las veces sino que se aplica un número determinado de veces, el cual estará fijado por la probabilidad de cruzamiento p_c ($p_c \leq 1$). Esto es, si se tiene una población de N = 100 *individuos* (puntos del espacio de búsqueda) y sabiendo que el cruzamiento se aplica por cada par de *individuos* de la población (cuando p_c = 1), entonces se aplicará 50 veces el operador de Cruzamiento en cada iteración (generación); pero si p_c = 0.6, el operador de Cruzamiento será aplicado aproximadamente 30 veces.

Entonces, la probabilidad de supervivencia de un ***esquema***, considerando la probabilidad de cruzamiento, está dada por la siguiente ecuación:

$$p_s \geq 1 - p_c \times \frac{\delta(H)}{\ell - 1} \qquad \textbf{(4.7)}$$

Ahora ya se puede considerar el efecto combinado de la reproducción y el cruzamiento. Asumiendo una independencia en las operaciones de reproducción y cruzamiento, se puede calcular el número esperado de *cromosomas*, para la siguiente generación, representados por un ***esquema*** particular por medio de la siguiente ecuación:

$$m(H, t+1) \geq m(H, t) \times \frac{f(H)}{f_{prom}} \times \left[1 - p_c \times \frac{\delta(H)}{\ell - 1}\right] \qquad \textbf{(4.8)}$$

De la ecuación anterior vemos que un esquema crece o decrece dependiendo de los factores de multiplicación. Esos factores dependen de que el promedio de los *fitness* del esquema esté por encima o por debajo del promedio de los *fitness* de la población y que la longitud del ***esquema*** sea corta o larga.

En otras palabras, los ***esquemas*** cuyos *fitness* promedio están por encima de la media de la población y tienen una longitud corta, tendrán una razón de crecimiento que obedecerá a un aumento exponencial del número esperado de *cromosomas*.

El operador de Mutación es el último operador que falta por analizar. Este operador, como ya se sabe, tiene por objetivo modificar el valor de una posición cualquiera en el *cromosoma*. El operador de Mutación está restringido a actuar de acuerdo a una probabilidad de mutación muy baja, p_m.

Un ***esquema*** sobrevive si todos los valores correspondientes a los *caracteres fijos* sobreviven. La probabilidad de que cada posición fija del *cromosoma* sobreviva está de acuerdo con $1 - p_m$. Por lo tanto, para obtener la posibilidad de supervivencia de un ***esquema*** de orden $O(H)$, multiplicamos la posibilidad de supervivencia $O(H)$ veces, y obtenemos $(1 - p_m)^{O(H)}$. Si $p_m << 1$, la probabilidad de supervivencia del ***esquema*** puede ser aproximada por $1 - O(H) \times p_m$.

Así tenemos que, un ***esquema*** particular H representa a un número esperado de *cromosomas* para la siguiente generación de acuerdo con la ecuación **(4.9)**, considerando ya el efecto combinado de la reproducción, cruzamiento y mutación:

$$m(H, t+1) \geq m(H, t) \times \frac{f(H)}{f_{prom}} \times \left[1 - p_c \frac{\delta(H)}{\ell - 1} - O(H)\, p_m \right] \qquad \textbf{(4.9)}$$

De la ecuación anterior podemos decir que el número esperado de *cromosomas* representados por un esquema particular, H, tendrá un crecimiento exponencial cuando el *fitness* promedio del ***esquema*** es superior al promedio de los *fitness* de la población, y también cuando la longitud del ***esquema*** es corta además de tener un bajo *orden*. El análisis matemático desarrollado anteriormente es conocido en la literatura como el *Teorema Fundamental de los Algoritmos Genéticos* [24].

4.6 ALGORITMOS GENÉTICOS PARALELOS

Los Algoritmos Genéticos han demostrado ser una muy buena herramienta para la búsqueda de óptimos globales en problemas tecnológicos complejos, debido a su relativa sencillez de implementación y a la robustez de su aplicación. Sin embargo, en la práctica los Algoritmos Genéticos necesitan de mucho tiempo de procesamiento para llegar a buenos resultados.

Una solución a este inconveniente es la paralelización del Algoritmo Genético, dividiendo el problema global en varios subproblemas que son asignados a varios procesadores pertenecientes a una red de computadoras. De esta forma, cada procesador realizará la computación del problema asignado obteniendo resultados parciales los cuales serán transmitidos a los otros procesadores de la red de computadoras colaborando todos para llegar a la solución del problema global [5].

4.6.1 IMPLEMENTACIONES SÍNCRONAS Y ASÍNCRONAS

Como ya se mencionó, para aumentar la velocidad de procesamiento de un dado problema se realiza la paralelización del mismo, dividiendo el problema global en diversos subproblemas. Cada subproblema se asigna a cada procesador disponible de una red de computadoras. De esta forma, cada procesador resuelve el problema asignado a él y transmite los resultados parciales a los demás procesadores, colaborando todos los procesadores del sistema para resolver el problema principal.

De acuerdo a como el algoritmo está diseñado para se realizar la transmisión de los resultados parciales los algoritmos paralelos puede ser síncronos o asíncronos. En los algoritmos paralelos síncronos, todas las transmisiones entre procesadores están sincronizados. En el momento en que se debe realizar la transmisión, todos los procesadores del sistema detienen la computación del problema a ellos asignados para realizar la comunicación de sus resultados parciales a los demás procesadores.

La implementación paralela síncrona no resulta conveniente para redes formadas por computadoras de diferentes arquitecturas y distintas velocidades de procesamiento, porque cuando llega el momento de realizar la comunicación entre procesadores, aquellas computadoras más rápidas deben esperar a que los procesadores más lentos terminen la computación del subproblema a él asignado. Por esta razón, los algoritmos paralelos síncronos requieren un perfecto *balanceamiento de cargas*; es decir, requieren que el tamaño de los subproblemas sean proporcionales a la capacidad de las computadoras a los cuales están asignados, con el fin de evitar los tiempos muertos de espera conocidos, también como *tiempo de sincronización.*

Otra alternativa para evitar el *tiempo de sincronización*, debido a la imposibilidad de realizar en la práctica un perfecto *balanceamiento de cargas*, consiste en no sincronizar las comunicaciones dejando a cada procesador realizar tantas iteraciones (generaciones) como sean necesarias, de acuerdo a su capacidad.

En una implementación paralela asíncrona, los procesadores del sistema pueden ser heterogéneos, trabajando cada uno de ellos de acuerdo a su velocidad de procesamiento. La principal ventaja de estas implementaciones asíncronas radica en el máximo aprovechamiento del tiempo de procesamiento, aún en presencia de retardos aleatorios [5]. Por consiguiente, al llegar el momento de la comunicación entre los procesadores del sistema las máquinas más rápidas no necesitan esperar a que los procesadores más lentos terminen la computación del problema a él asignado, evitando de esta manera los tiempos muertos de espera.

Sin embargo, este tipo de algoritmo requiere un diseño más complejo, pues en el contexto asíncrono, se pierde el concepto global de iteración debido a que algunos procesadores podrán realizar más iteraciones que otros en el mismo periodo de tiempo, y por lo tanto, una iteración en el procesador 1 no será necesariamente equivalente a la misma iteración en el procesador 2.

4.6.2 CLASIFICACIÓN DE ALGORITMOS GENÉTICOS PARALELOS

Una de las principales características del Algoritmo Genético es el paralelismo implícito en su mecanismo de búsqueda para llegar al óptimo global; y por esta razón, el paso del Algoritmo Genético secuencial a la implementación paralela del Algoritmo Genético fue relativamente rápida. Desde que se presentaron las primeras ideas acerca de la paralelización de los Algoritmos Genéticos hasta nuestros días, se han realizado numerosas implementaciones en este campo. Pero debido a la gran diversidad de trabajos relacionados con los Algoritmos Genéticos Paralelos (AGP) se hizo necesaria la continua clasificación de los mismos y la descripción de sus características, para que de esta forma se pueda organizar las informaciones que se tenía hasta el momento con la posibilidad de determinar las áreas aún abiertas de investigación [43].

A continuación, se presenta una clasificación de los Algoritmos Genéticos Paralelos acompañada de sus características más atractivas con los trabajos más relevantes que han sido publicados al respecto:

a) **Paralelización Global del Algoritmo Genético:** En este tipo de implementación, la evaluación de los *individuos* de la población es explícitamente paralelizada. La evaluación de la población consiste en la obtención del *fitness* de cada *individuo*. Es por eso que, cuando el cálculo del valor de la función objetivo es muy pesado desde el punto de vista computacional se recurre a este tipo de implementación.

   Cabe destacar también que en este tipo de implementación no existe comunicación entre los procesadores de la red mientras se realiza la evaluación de los *individuos* de la población. La comunicación sólo ocurre al comienzo y al final de la fase de evaluación.

La paralelización puede ser realizada en Computadoras de Memoria Compartida, así como también en Computadoras de Memoria Distribuida o bien sobre una red de computadoras.

Una computadora de Memoria Compartida está formada internamente por un procesador central, denominado con frecuencia *administrador*, que a su vez está comunicado con un determinado número de procesadores. La memoria de trabajo del *administrador* se denomina generalmente *memoria global*. Cada uno de los otros procesadores del sistema comparte parte de la memoria de trabajo del *administrador* para realizar tareas relativamente más simples. Sin embargo, no existe comunicación entre los otros procesadores entre sí, pero si lo hay entre cada procesador del sistema con el *administrador*.

En la implementación en Computadoras de Memoria Compartida, los *individuos* de una población pueden ser almacenados en la memoria global. Cada procesador puede leer los *individuos* asignados a él y entregar luego los resultados de la evaluación sin ningún conflicto. Por supuesto, para que esto ocurra es necesario que exista una sincronización entre las generaciones (iteraciones). Es una implementación netamente síncrona.

Sin embargo, en Computadoras de Memoria Distribuida, en donde todos los procesadores de la Computadora están todos comunicados entre sí, la población puede ser almacenada en un procesador para simplificar la aplicación de los Operadores probabilísticos del Algoritmo Genético. Este procesador, denominado procesador *Master*, es el responsable de enviar a los otros procesadores, los *individuos* asignados a ellos. En cada uno de estos procesadores se realiza la evaluación de los *individuos*, que fueron enviados desde el *Master*.

En la mayoría de los trabajos más relevantes de paralelización realizados con este tipo de implementación, los resultados numéricos obtenidos indican *speedup*[5] sublineales, debido que al aumentar la cantidad de procesadores utilizados aumenta también la capacidad de computación y el "caudal" de comunicación.

b) Algoritmos Genéticos Paralelos de *granularidad gruesa*: Antes de empezar la descripción de este tipo de AGP, es conveniente explicar el significado de *granularidad* en parelelismo. La *granularidad* en paralelismo se refiere a la razón entre el tiempo empleado en computación y el tiempo empleado en la comunicación de datos entre procesadores, la cual puede ser expresada en forma matemática de la siguiente manera:

$$granularidad = \frac{Tiempo\ de\ Computacion}{Tiempo\ de\ Comunicacion} \qquad \textbf{(4.10)}$$

En este tipo de implementación la *granularidad* > 1, debido a que el tiempo de computación es mayor que el tiempo empleado en la comunicación. Está caracterizado además por la presencia de unas pocas subpoblaciones de *individuos* relativamente grandes y la introducción de un nuevo operador: el operador de Migración. Además, el número de subpoblaciones es igual al número de procesadores que se utilizarán.

El operador de Migración, que indica la forma en como se organizará y se distribuirá la comunicación de los *individuos* entre los diversos procesadores disponibles de la red, introduce nuevos conceptos que se describen a continuación:

[5] *Speedup* o aceleración, en todo el presente trabajo, se define como el cociente entre el mejor tiempo secuencial y el tiempo de la ejecución paralela del tipo de AGP considerado.

1) *Intervalo de Migración*. Establece cada cuantas generaciones se realizará la migración de una cierta cantidad de *individuos* desde una subpoblación a la otra.

2) *Tasa de Migración*. Indica cuantos *individuos* han de comunicarse a la otra subpoblación, cuando se cumple el intervalo de migración.

3) *Criterio de Selección de los Migrantes*. Determina la política que se aplicará para la selección de los *individuos* que han de migrar. Siguiendo la filosofía eminentemente aleatoria de los Algoritmos Genéticos, se puede establecer que los migrantes sean elegidos al azar. Sin embargo, es posible ayudar a las otras subpoblaciones seleccionando los *individuos* de mejor *fitness*. Esta última opción parece más natural desde el punto de vista biológico, pues en la Naturaleza, las migraciones involucran grandes esfuerzos a los migrantes, por lo cual, serán los mejores los que terminen la travesía.

4) *Topología de Comunicación*. Define el sentido de las migraciones, es decir, como estará organizada la conexión entre los procesadores a través de los cuales se realizará la transmisión de los resultados parciales. Por ejemplo, en DGENESIS [44] se propone un modelo sencillo que ofrece gran flexibilidad para la representación de diversas topologías tales como anillos y mallas.

La mayoría de los trabajos realizados en este campo utilizaron computadoras paralelas de Memoria Distribuida [43], y también fueron desarrolladas sobre redes de computadoras. Los esfuerzos realizados en esta área fueron los más importantes en el campo de las implementaciones paralelas de Algoritmos Genéticos.

En las primeras implementaciones, los investigadores se preocuparon más de la frecuencia en que ocurrían las migraciones y de cuantos *individuos* migraban a los otros procesadores. Asimismo, se utilizaron diversas topologías de comunicación. De los primeros esfuerzos realizados en esta área, se obtuvieron importantes observaciones relacionadas con el comportamiento de los Algoritmos Genéticos Paralelos:

- En 1987, Reico Tanese [45], implementó el AGP en una topología de hipercubo de cuatro dimensiones. En este algoritmo los migrantes eran escogidos de entre los *individuos* probabilísticamente mejores de la población, y reemplazaban a los peores de la subpoblación que los recibía. Obtuvo *speedups* (aceleración) cercanamente lineales.

- En 1989, Grosso [46] dividió la población total en cinco subpoblaciones. Realizó, además de sus primeras aplicaciones, un análisis exhaustivo probando con diversos números de migrantes. Comprobó que para una muy baja cantidad de migrantes el tiempo de procesamiento del AGP disminuye aún más pero los valores promedios de los *fitness* obtenidos fueron menores que los obtenidos por el Algoritmo Genético Secuencial. Al aumentar considerablemente el número de los *individuos* que han de migrar hacia las otras subpoblaciones, el comportamiento del AGP era casi idéntico al de un AG secuencial.

- En 1989, Tanese [47] continuó sus investigaciones realizando un estudio exhaustivo sobre el intervalo de migración y la tasa de migración. Encontró que existe una razón de migración crítica por debajo del cual las subpoblaciones se comportan como si estuvieran aisladas, y por arriba del cual el comportamiento era muy parecido al de un AG secuencial.

En trabajos posteriores se intentó llegar a un modelo matemático del AGP para describir su comportamiento. Sin embargo, esto sólo derivó en unas cuantas conclusiones mas bien empíricas [43]. Por lo tanto, una de las áreas aún hoy abiertas de investigación, y posiblemente uno de los mayores retos en los Algoritmos Genéticos sea el fundamento matemático que explique el comportamiento de un AGP.

- En un trabajo presentado en 1991, Mühlenbein [8] demostró el potencial de los Algoritmos Genéticos Paralelos utilizando problemas grandes y complejos. En su implementación incluyó un optimizador local, distinto del Algoritmo Genético. De ahí que no es muy claro si los muy buenos resultados obtenidos en la experimentación fueron debidos al AG o al optimizador local implementado.

En todos los trabajos realizados de AGP de *granularidad gruesa*, el operador de Migración ha sido un tema de estudio continuo y hasta la fecha ha prevalecido la experiencia. Esto es, todos los valores asignados a los parámetros que maneja la Migración han sido determinados en forma empírica.

Además, un aspecto que tradicionalmente no se considera con la debida profundidad en el momento de realizar una implementación de este tipo de AGP está relacionado con la *topología de comunicación* [43]. La *topología de comunicación* es un factor importante en el rendimiento de los AGP, porque determina que tan rápido (o que tan lento) buenas soluciones se diseminan a las otras subpoblaciones. Si la *topología de comunicación* tiene una conectividad densa (cuando cada uno de los procesadores del sistema se comunica con el resto de los procesadores, y lo hacen con frecuencia), buenas soluciones se difundirán más rápidamente a todas las demás subpoblaciones.

Sin embargo, si las subpoblaciones están escasamente conectadas (las subpoblaciones prácticamente están aisladas) buenas soluciones se difundirán más lentamente a todos los demás procesadores, y podría presentarse en cada uno de ellos soluciones muy diferentes en comparación con los obtenidos en los otros procesadores.

En un trabajo realizado por Cantú Paz y Mejía Olvera [48], se consideró diversas topologías con la que demostraron experimentalmente que subpoblaciones densamente conectadas encuentran la solución global más rápidamente que aquellas muy aisladas. Realizaron implementaciones con topologías conectadas completamente en hipercubos de 4D, una red 4×4 toroidal y anillos unidireccionales y bidireccionales.

c) Algoritmo Genético Paralelo de *granularidad fina*: En estos algoritmos, se divide la población global en un gran número de subpoblaciones. Este tipo de implementación requiere la utilización de Computadoras Masivamente Paralelas.

Un caso ideal para este tipo de implementación del AGP, es llegar a tener en cada uno de esos procesadores del sistema un solo *individuo*. En esta idea, propuesta por Mühlenbein [21, 43], una población es distribuida en una superficie geográfica virtual. Dicha superficie virtual está dada por los múltiples procesadores del sistema que albergarán a un solo *individuo* de la población. La topología de comunicación formada entre los procesadores corresponde a una malla, formando como un cuadriculado en la que cada punto de la cuadrícula corresponde a un procesador del sistema.

De esta forma cada procesador puede seleccionar un *individuo* de entre los procesadores de su vecindad, luego ejecuta las operaciones de cruzamiento y mutación, escogiendo posteriormente al mejor de entre los descendientes resultantes. El *individuo* con más alto *fitness* sustituirá al anterior *individuo* del procesador correspondiente. Como no hay necesidad de globalizar el procesamiento, la búsqueda se torna asíncrona y la población toda tiende ha mantener un buen nivel de diversidad.

d) **Algoritmos Genéticos Paralelos Mixtos:** Este tipo de paralelización consiste en combinar en una implementación los algoritmos vistos anteriormente. Un forma de paralelización mixta consistiría en utilizar una *paralelización global* en cada una de las subpoblaciones de un AGP de *granularidad gruesa* [43]. De hecho un ejemplo concreto de dicha implementación puede observase en el siguiente trabajo [49], en donde los autores primeramente realizaron una paralelización global del problema, en la que paralelizaron la evaluación de la función objetivo. Luego, el mismo problema se partió y se colocó en diversos procesadores con el fin de que cada uno de ellos aplique un AG para la optimización de la función objetivo (esta implementación corresponde a un AGP de *granularidad gruesa*). Por último se realizó la paralelización mixta, en la que combinaron las dos estrategias anteriores.

CAPÍTULO 5: ASYNCHRONOUS TEAM (*A-TEAM*)

5.1 INTRODUCCIÓN

El impresionante avance tecnológico de los últimos años trajo consigo la necesidad de resolver problemas cada vez más complejos y con mayores exigencias en cuanto al tiempo de respuesta y la confiabilidad del sistema. De esta forma surge una creciente necesidad de incrementar la capacidad de cómputo efectivo, lo que sirvió de catalizador para el extraordinario desarrollo de los *sistemas distribuidos*. Es así como los ambientes computacionales distribuidos dominan el nuevo contexto a nivel mundial. Redes de computadoras, cada vez más eficientes, están hoy disponibles para la gran mayoría de los profesionales que trabajan con tecnología de punta, inclusive en países menos desarrollados como el Paraguay.

La razón por la cual los sistemas distribuidos [6] son requeridos cada vez más en problemas de gran porte, se debe principalmente a:

- La utilización de computadoras de bajo costo conectadas entre sí formando una red, resultado esta implementación más económica que los tradicionales sistemas uniprocesadores (supercomputadoras);
- La simplicidad con que los sistemas distribuidos se adaptan a la distribución natural de personas, información y recursos en general;
- La confiabilidad y disponibilidad proporcionados por mecanismos de tolerancia a fallas.

A pesar de todas las ventajas mencionadas arriba, los sistemas distribuidos aún son muy poco utilizados debido principalmente a las dificultades en adaptar las tradicionales implementaciones secuenciales a los nuevos ambientes distribuidos, con alta capacidad de paralelización. Razón por la cual, actualmente se están invirtiendo grandes esfuerzos en investigación para conseguir aprovechar eficientemente las ventajas proporcionadas por el procesamiento distribuido.

5.2 CONCEPTO DE LOS *ASYNCHRONOUS TEAMS*

La idea para aprovechar eficientemente un sistema distribuido consiste en la descomposición de un problema complejo en subproblemas menores, asignando luego cada uno de ellos a un número de procesadores disponibles de un sistema distribuido de forma tal que cada procesador resuelva uno o más subproblemas cuidando que el conjunto resuelva el problema global. Para la resolución de cada subproblema se puede elegir el método más adecuado, lo que lleva a tener una combinación de métodos variados siendo ejecutados en unidades de procesamiento diferentes y comunicando los resultados parciales de tal forma que todos colaboran para la resolución global del problema (*Team Algorithm*) [6].

Los algoritmos adecuados a cada subproblema a resolver, y que son asignados a los diferentes procesadores disponibles del sistema distribuido, pueden ser diseñados de tal forma que la comunicación entre dichos procesadores no esté sincronizada, permitiendo de esta forma aprovechar los recursos computacionales al máximo en redes de computadoras posiblemente heterogéneas (diferentes desempeños, recursos y velocidades de procesamiento) sin necesidad de invertir en recursos computacionales costosos para la sincronización de tareas. Este tipo de implementación paralela se conoce en la literatura como *A-Teams* (*Asynchronous Teams* [9, 50]).

El número de procesadores disponibles en la red, al igual que el número de algoritmos utilizados puede ser arbitrariamente grande. Además, dichos algoritmos pueden ser distribuidos sobre un área de comunicación arbitrariamente grande [50]. Cada algoritmo puede ser completamente autónomo (cada algoritmo decide con qué resultados trabajará y cuando).

De esta forma, los *A-Teams* han sido considerados como extremadamente eficientes para la resolución de problemas muy difíciles para los cuales muchos métodos tradicionales utilizados en forma independiente no tienen un desempeño satisfactorio. Algunos métodos pueden ser muy precisos en sus resultados pero a la vez son muy lentos, otros son rápidos en el procesamiento pero muy poco precisos en la solución obtenida. Ocasionalmente, puede suceder que ninguno de los métodos disponibles consigue resolver el problema en cuestión, cuando son utilizados en forma independiente [9].

Así, un *A-Team* combina estos algoritmos de tal forma a obtener un mejor desempeño en la resolución de problemas complejos, mejorando la calidad de los resultados obtenidos en comparación a los resultados hallados por los métodos tradicionales, aumentando además la velocidad de procesamiento.

Actualmente los *A-Teams* ya han sido aplicados en numerosos problemas difíciles e importantes, como [50]: Diseño de Edificios de gran altura, diagnósticos de fallas en redes eléctricas, control en redes eléctricas, El Problema del Transporte de Aceite a través de tuberías [51], aplicación de *A-Teams* a la industria del Acero en pequeñas Acerías [52] entre otros.

5.3 ALGORITMOS GENÉTICOS COMBINADOS

Los Algoritmos Genéticos han demostrado ser métodos de búsqueda robustos debido a que poseen un muy buen desempeño ante una amplia gama de problemas, y la metodología de optimización del AG conduce siempre al óptimo global. Esto es así porque la búsqueda del óptimo global se realiza a partir de una población de posibles soluciones (paralelismo implícito) y no a partir de un solo punto inicial.

Sin embargo, a pesar de las evidentes ventajas que presenta el Algoritmo Genético frente a tradicionales métodos numéricos de optimización, el algoritmo invierte un tiempo considerable para encontrar buenas soluciones. Mientras que los métodos numéricos de optimización son rápidos pero muy poco robustos. Este antagonismo ha llevado a pensar en algoritmos híbridos que aprovechan las ventajas de cada método y demuestran experimentalmente sus beneficios.

La idea de combinar el Algoritmo Genético con métodos numéricos de optimización y aprovechar las ventajas de ambos métodos proviene de los *Team Algorithms* (TA) [9]; en ellos un problema complejo se descompone en diversos subproblemas menores aplicando a cada uno de ellos diferentes métodos numéricos adecuados al subproblema a resolver. De esta forma, se salva la dificultad de que un método no tenga un buen desempeño para un determinado problema complejo, o no lo pueda resolver.

En la mayoría de los trabajos en la que se combina el AG con algún método numérico de optimización, el optimizador local empleado se utiliza como un operador del Algoritmo Genético [8,10]. Por consiguiente, el AG combinado utiliza un operador de *optimización local* que trabaja de la siguiente forma:

1) Se selecciona un *individuo* de la población. El *individuo* puede ser seleccionado en forma aleatoria. Aunque también puede ser escogido siempre el *individuo* de más bajo *fitness*, con el fin de aumentar el valor numérico de la función objetivo después de aplicar el método numérico.
2) Se aplica el método numérico de optimización asignado, optimizando al *individuo* escogido posiblemente hasta un óptimo local;
3) Reemplaza al *individuo* escogido por su versión optimizada.

Una vez que el *individuo* es optimizado por el método numérico, el *individuo* optimizado sustituye al *individuo* seleccionado, en la población anterior. A continuación se presenta un pseudocódigo (ver Pseudocódigo 5.1), en el que se muestra como se aplica el algoritmo de optimización local en el AG:

```
t ← 0;
Iniciar Población ( t );
Evaluar Población ( t );
DO WHILE ( No se cumpla Criterio de Parada )
  t ← t + 1;
  Seleccionar Población ( t ) a partir de Población ( t – 1 );
  Cruzar y mutar Población ( t );
  IF ( t ≥ numMax ) THEN Optimizador_Local;
  Evaluar Población ( t );
END DO
```

Pseudocódigo 5.1: Algoritmo Genético Combinado.

En el Pseudocódigo 5.1, se observa que el método numérico utilizado es aplicado, en general, a partir de un cierto número de generaciones en adelante. La cantidad de generaciones a partir del cual se aplica el optimizador local es habitualmente empírica.

El objetivo de esta estrategia es evitar la aplicación del método numérico en las primeras generaciones, en las cuales los *individuos* aún tienen muy bajos *fitness*, con lo que la aplicación del método numérico no sería de mucho beneficio debido a que el punto obtenido no tiene aún información que lo relacione con el óptimo global [8].

Este tipo de implementación híbrida ha sido aplicado con ventajas, presentando mejores soluciones en menores tiempo de ejecución comparados con tradicionales métodos numéricos de optimización [5, 9-10].

5.4 ALGORITMOS GENÉTICOS PARALELOS Y COMBINADOS

Uno de los métodos más naturales para mejorar los tiempos de respuesta de los algoritmos secuenciales es la paralelización de los mismos. De esta forma, el algoritmo combinado presentado en la sección anterior es paralelizado en un sistema distribuido. En efecto, la población de *individuos* puede descomponerse en subpoblaciones, asignándose cada una de ellas a diversos procesadores de una red de computadoras posiblemente heterogéneas, que periódicamente intercambian *individuos* entre sí de acuerdo a una política migratoria.

```
Leer_Datos;
Levantar_Procesos_Esclavos;
Enviar_Parámetros_a_cada_Esclavo;
Fin ⇐ FALSE;
DO WHILE ( Fin )
        Recibir_Mensaje_de_Terminación_de_Esclavos;
        IF (Todas las máquinas terminaron) THEN Fin ⇐ TRUE;
END DO
Enviar_Mensaje_de_Fin_a_Esclavos;
Eliminar_Process_Esclavos;
```

Pseudocódigo 5.2: proceso *Master*.

La implementación paralela diseñada está basada en el modelo *Master/Esclavo* [44]. De esta forma, el proceso *Master* se encarga de levantar los procesos *Esclavos*, y de enviar a cada uno de ellos los parámetros correspondientes del problema. En el Pseudocódigo 5.2, se presenta al algoritmo del proceso *Master*.

En cada proceso *Esclavo* (ver Pseudocódigo 5.3) se aplica, sobre la subpoblación asignado a él, el Algoritmo Genético combinado (AGC) descripto en la sección anterior. Cuando el criterio de terminación de cada proceso *Esclavo* se haya cumplido cada uno de ellos envía un mensaje de terminación al proceso *Master*.

Una vez que el proceso *Master* recibe todos los mensajes de terminación de los procesos *Esclavos*, se da por finalizado la ejecución del algoritmo eliminando todos los procesos paralelos.

```
Recibir Parámetros;
Iniciar Población;
Estadística de la Población;
Selección de Individuos;
DO WHILE ( TRUE )
    Reproducción;                /*Cruzamiento y Mutación*/
    Escoger Migrantes;
    Enviar Migrantes;            /*a otros procesos Esclavos*/
    Recibir Migrantes;           /*de otros procesos Esclavos*/
    Seleccionar Individuos;      /*Manteniendo tamaño Poblacional*/
    Estadística de la Población;
    IF ( t ≥ numMax ) THEN Optimizador_Local;
    IF ( criterio de fin ) THEN Mensaje Fin al Master;
END DO
```

Pseudocódigo 5.3: Proceso *Esclavo*.

La idea de combinar el Algoritmo Genético con métodos numéricos tradicionales e implementarlos en un ambiente paralelo de comunicación asíncrona proviene del trabajo realizado Mühlenbein [8].

En el trabajo de Mühlenbein el *A-Team* implementado combina un Algoritmo Genético Paralelo de *granularidad gruesa* con algoritmos de búsqueda aleatoria, utilizados como operadores del Algoritmo Genético para la optimización de funciones mucho más complejas que las funciones de *test* utilizadas por D´Jong para probar al Algoritmo Genético [26]. De esta forma, se obtuvo una combinación de algoritmos que interactuaban sinergéticamente reduciendo de esta manera los recursos computacionales que demandará la sola utilización de métodos numéricos tradicionales, reportándose muy buenas aceleraciones (*speedup*).

De acuerdo con el resultado obtenido por este importante trabajo, la combinación del Algoritmo Genético con tradicionales métodos numéricos y tras la paralelización del algoritmo híbrido resultante, se obtuvo no sólo mejoras en cuanto a los resultados numéricos obtenidos sino que también la convergencia a óptimos globales fue mucho más rápido. De esta forma, las ventajas en el desempeño de los *A-Teams* han traído consigo un creciente interés en la utilización de esta técnica para diversas aplicaciones [50-52].

CAPÍTULO 6: IMPLEMENTACIÓN DEL *A-TEAM* PARA LA MINIMIZACIÓN DEL CAUDAL TURBINADO.

El desempeño de un algoritmo se mide generalmente por la calidad de los resultados obtenidos y por la rapidez en la respuesta. En este aspecto, los *Team Algorithm* han demostrado ser muy eficientes superando muchas veces en desempeño a los métodos tradicionales [5, 7, 10, 13-18]. Debido a estas ventajas en combinar diferentes métodos , se realizó en el presente trabajo la combinación del Algoritmo Genético con un método numérico de optimización tradicional, como el método del Gradiente. Este algoritmo combinado fue aplicado al problema de minimización del caudal total turbinado en una represa hidroeléctrica.

Sin embargo, a medida que los problemas se vuelven más complejos las implementaciones secuenciales ya no resultan muy prácticas debido a que invierten mucho tiempo para llegar a buenos resultados. Como ya se ha visto, una alternativa para disminuir el tiempo de computación consiste en paralelizar el problema, descomponiéndolo en numerosos subproblemas. Al respecto, los A-Teams han surgido como una novedosa herramienta que no solo mejora los tiempos de respuesta del algoritmo sino que se ha comprobado que tiene un efecto sinergético [5, 7, 9, 15-18], esto es, el algoritmo no solo mejora la velocidad de procesamiento sino que también la calidad de los resultados, razón por la cual en el presente capítulo se describe el desarrollo de un A-Team formado por un AG y el método numérico utilizado (descripto en el Capítulo 3) similar al desarrollado por Mühlenbein [8], y su aplicación al problema de optimización de energía de una Central Hidráulica.

6.1 IMPLEMENTACIÓN DE UN ALGORITMO GENÉTICO COMBINADO

La aleatoriedad como parte del mecanismo de búsqueda del AG puede representar un obstáculo para el cumplimiento de la restricción del problema, representada por la ecuación **(3.2)**, en la que la sumatoria de las potencias (P_i) generada por cada turbina *i* debe ser igual a la potencia demandada (P_D).

De esta forma, en el presente trabajo se desarrolló un *Team Algorithm* formado por un Algoritmo Genético combinado con un método numérico de optimización (como el método numérico utilizado descripto en el Capítulo 3 del presente trabajo). La implementación de este *Team Algorithm* está basado en trabajos similares como los presentados en [6, 10-11]. De esta forma, en el *Team Algorithm* aquí desarrollado el Algoritmo Genético trabaja sólo con una cantidad determinada de turbinas aplicándose el método numérico en las unidades restantes, asegurando de esta manera el cumplimiento de la referida restricción. Por consiguiente, se puede decir que el AG es el algoritmo responsable de encontrar las *regiones prometedoras* que serán exploradas en detalle por el método numérico, reduciéndose de esta manera la dimensión del espacio de búsqueda del método numérico.

En este trabajo, la función objetivo que se busca optimizar está dada por la eficiencia total en la generación de energía. La eficiencia total η en la generación de energía, es representada por la siguiente fórmula [4]:

$$\eta = \frac{P_D}{\delta \times h \times \sum Q_i} \qquad \textbf{(6.1)}$$

donde:

- Q_i es el caudal turbinado en cada unidad generadora *i*,
- ΣQ_i indica el caudal total entregado a todas las turbinas en servicio,
- δ representa el peso específico del fluido turbinado.

Los valores máximo y mínimo de la función objetivo están muy próximos entre sí con la definición **(6.1)**, lo que produce un mal desempeño del operador de Selección, y por ende pierde su capacidad de búsqueda, estancándose en máximos locales. Una forma de corregir este defecto es "*escalar*" la función objetivo [24]. A esta función escalada se lo conoce en la literatura como función de adecuabilidad o *fitness* [21, 24]. El escalamiento utilizado en el presente trabajo corresponde a un escalamiento lineal, el cual es representado gráficamente en la **Figura 6.1**.

De esta forma el Algoritmo Genético trabaja con el *fitness* o función escalada *f*, el cual es representado matemáticamente como sigue:

$$f = a \times \eta + b \qquad \textbf{(6.2)}$$

donde a y b son constantes que se definen a partir de la **Figura 6.1**:

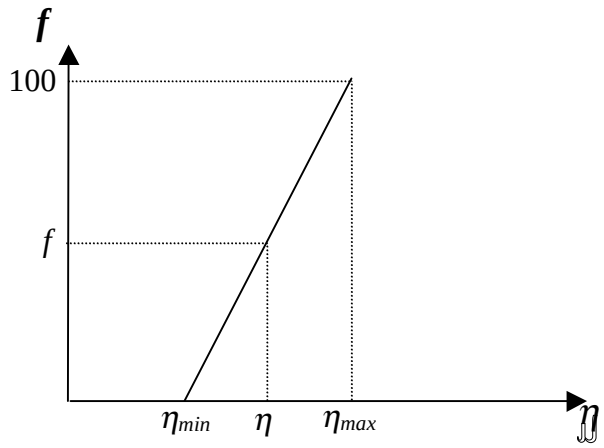


Figura 6.1: Escalamiento de la eficiencia η.

De esta forma, para el problema de minimización del caudal total turbinado, tenemos que en el Algoritmo Genético un *individuo* está formado por una estructura de datos compuesta por las siguientes variables:

- la codificación, en sistema binario, de las potencias (P_i) asignadas al Algoritmo Genético. La representación binaria de las variables independientes se la conoce como *cromosoma* [21, 24]. En el trabajo aquí presentado las variables independientes son las potencias P_i;

- los valores numéricos reales de cada potencia P_i, donde ($i \leq N_T$);
- los valores reales de caudal Q_i turbinado en cada unidad generadora *i*, donde ($i \leq N_T$). Dichos valores de caudal son obtenidos a partir de una lista de potencias en función de caudal, es decir, a partir de un "*index-test*";
- la eficiencia η en la generación de energía.

Para cada *individuo j* donde ($j \leq N$), siendo N la cantidad de *individuos* de la población, el Algoritmo Genético calculará la potencia de *k* unidades ($1 \leq k < N_T$) y el método numérico utilizado calculará la potencia de las ($N_T - k$) unidades restantes, sirviendo de complemento para el cálculo de la eficiencia η. De esta forma, se asegura el cumplimiento de la restricción **(3.2)** quedando conformado un *Team Algorithm* que trabaja de la siguiente manera:

a) En el Algoritmo Genético, para cada *individuo j* de la población inicial, se generan aleatoriamente sus potencias P_i ($1 \leq i \leq k < N_T$); siendo P_i la potencia generada por la turbina *i* de la central hidráulica.

b) A continuación dentro del Algoritmo Genético, para cada *individuo j* de la población inicial, se aplica el método numérico utilizado, de forma a calcular las potencias P_i ($k < i \leq N_T$). Dicho cálculo se realiza asignando a las ($N_T - k$) turbinas restantes una potencia deseada de generación de ($P_D - \Sigma P_i$).

c) Conocidas todas las potencias P_i, se obtienen los caudales Q_i correspondientes (usando los "*index-test*") para calcular la eficiencia η, utilizando la ecuación **(6.1)**, y luego su función de adecuabilidad *f* conforme **(6.2)**.

d) Mientras no se cumpla el criterio de parada del algoritmo, se aplican en cada generación los operadores probabilísticos del AG y se evalúan los nuevos *individuos*.

e) Una vez que se cumpla el criterio de parada, el *Team Algorithm* implementado (Algoritmo Genético combinado), proporciona el *individuo* con la mejor eficiencia η. Sin embargo, la solución así obtenida puede contener errores de redondeo debido a la representación binaria de las unidades generadoras. Entonces, se finaliza aplicando el método del Gradiente a la solución proporcionada por el AG combinado.

En el **Pseudocódigo 6.1** se presenta la forma en que se implementó el *Team Algorithm*, en donde también se describe como se aplican los operadores probabilísticos del AG y el método numérico utilizado.

```
t ← 0;
Iniciar Población(t);
Estadística de la Población(t);

DO WHILE (No se cumpla criterio de Parada)
   t ← t + 1;
   Reproducción;                              /*Cruzamiento y Mutación*/
   Evaluación fitness de Individuos;          /*Aplicando Método Numérico*/
                                              /*para cumplir restricción (3.2)*/
   Seleccionar Población(t) de Población(t−1);
   Estadística de la Población(t);
END DO

Aplicar Gradiente (Individuo solución);
```

Pseudocódigo 6.1: *Team Algorithm* implementado.

6.2 IMPLEMENTACIÓN DEL *A-TEAM*

Dado el buen desempeño de los algoritmos paralelos asíncronos se dispuso la paralelización, en un ambiente asíncrono, del *Team Algorithm* implementado formando un algoritmo que en la literatura es conocido como *A-Team* (Asynchronous Teams). El algoritmo así desarrollado está basado en el trabajo de Mühlenbein [8].

El *A-Team* desarrollado en el presente trabajo está basado en el modelo *Master*/*Esclavo* [44]. El proceso *Master* (ver **Pseudocódigo 6.2**) se encarga de administrar los recursos incluyendo el lanzamiento de diversos procesos *Esclavos*, y enviar a los mismos todos los datos que configuran el problema. En estos procesos *Esclavos* (ver **Pseudocódigo 6.3**) se aplica el Algoritmo Genético combinado ya descripto, de esta forma cada procesador *Esclavo* realiza el cálculo propiamente dicho. Por consiguiente:

```
Leer Datos;
Levantar Procesos Esclavos;
Enviar Parámetros a cada Esclavo;
Fin ← FALSE;
DO WHILE (Fin)
        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 6.2: Proceso *Master*.

La aplicación del *A-Team* aquí desarrollado implica la división de una población de N *individuos* en pequeñas subpoblaciones. Cada una de las subpoblaciones es asignada a cada procesador disponible de una red de computadoras.

En cada proceso *Esclavo* se aplica el *Team Algorithm* descripto sobre la subpoblación correspondiente, y llegado el momento de la transmisión de los resultados parciales se aplica el operador de Migración escogiendo una determinada cantidad de los mejores *individuos* de la subpoblación para que migren a los demás procesadores del sistema, obteniéndose de esta manera un buen comportamiento del *A-Team* en lo que se refiere a la rapidez de computación y la calidad en los resultados obtenidos.

```
Recibir Parámetros;
Iniciar Población;
Estadística de la Población;
Selección de Individuos;
DO WHILE ( TRUE )
        Reproducción;                       /*Cruzamiento y Mutación*/
        Evaluación fitness cada Individuo;  /*Aplicando Método Numérico*/
                                            /*para cumplir restricción (3.2)*/
        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 ( Criterio de Fin ) THEN
                Aplicar Gradiente ( Individuo solución );
                Mensaje al Master;
        END IF
END DO
```

Pseudocódigo 6.3: Proceso *Esclavo*.

Es necesario recordar que la transmisión de los *individuos* a los demás procesadores del sistema se realiza en forma asíncrona (la transmisión entre procesadores de la red no está sincronizada), permitiendo de esta manera que cada computadora trabaje de acuerdo a su capacidad de procesamiento, eliminando los *tiempos de sincronización*. De esta manera, como se verá en los resultados experimentales, se logra no solo aumentar la velocidad de procesamiento sino que también la calidad de los resultados obtenidos son comparativamente superiores a los obtenidos de las implementaciones debido a que el asincronismo introduce un efecto sinergético en el *A-Team*.

Una vez que se cumpla el criterio de parada del *Team Algorithm* de cada proceso *Esclavo*, se obtiene en cada uno de ellos el mejor valor correspondiente para la eficiencia η. Sin embargo, debido a la representación binaria de las potencias $P_i (i \leq N_T)$, se puede cometer errores de redondeo. Por esta razón, sobre los *individuos* que resulten de cada proceso *Esclavo* se aplica nuevamente el método del Gradiente con el fin de mejorar la precisión del resultado obtenido.

Finalmente, luego de aplicar el método del Gradiente, cada proceso *Esclavo* envía un mensaje de terminación al proceso *Master*. Cuando el proceso *Master* recibe todos los mensajes de terminación provenientes de todos lo procesos *Esclavos*, se da por finalizado el algoritmo.

CAPÍTULO 7: ESTUDIOS EXPERIMENTALES

En este capítulo se presenta el modelo de la represa hidroeléctrica con el cual se realizó la toma de datos experimentales, aplicando los métodos de optimización estudiados en los capítulos anteriores.

Los métodos de optimización desarrollados y utilizados para la obtención de los resultados experimentales son los siguientes:

- el método numérico de optimización, descripto en el Capítulo 3;
- el *Team Algorithm* formado por la combinación del Algoritmo Genético con el método numérico utilizado;
- el *A-Team* propuesto.

Los resultados numéricos obtenidos fueron comparados entre sí, con el fin de analizar el comportamiento característico de cada método desarrollado y establecer, de esta manera, las ventajas y desventajas de cada uno de las implementaciones estudiadas.

7.1 AMBIENTE COMPUTACIONAL

El sistema distribuido utilizado consistió en una red Ethernet de 10 Mbps constituida por 5 computadoras personales PREMIO, con procesadores Pentium de 75 MHz, 8 Mb de memoria RAM y operando con el Sistema Operativo LINUX.

Los algoritmos secuenciales utilizados en el presente trabajo fueron inicialmente codificados en Lenguaje ANSI C (standard). En cambio, el *A-Team* propuesto fue implementado en PVM (*Parallel Virtual Machine*) versión 4.02, el cual es una versión extendida del Lenguaje C, y permite el diseño de algoritmos paralelos síncronos y asíncronos.

7.2 PROBLEMA EJEMPLO

Para la toma de datos experimentales se considera como problema ejemplo una represa hidroeléctrica con 9 unidades generadoras, pudiendo generar cada una de ellas entre 300 MW y 710 MW. Para cada turbina se tienen las correspondientes curvas de eficiencia obtenidas experimentales a partir de un "*index-test*". Para todos los resultados se consideró que el parámetro correspondiente a la altura del embalse es de h = 116,5 metros.

Para todas las implementaciones realizadas, los parámetros de entrada son los siguientes: la potencia demandada o requerida (P_D), la altura disponible (h), las turbinas que están en servicio y el número de turbinas en servicio (N_T). Los algoritmos diseñados trabajan con las curvas de eficiencia interpoladas con respecto al parámetro h, dado que el valor de la variable h está parametrizada, tomando algunos valores discretos.

En el método numérico desarrollado, el dominio en el cual el método de búsqueda exhaustivo realizará el análisis global se divide en subespacios, formando una CUADRÍCULA. La dimensión del espacio de búsqueda, en este caso, es igual al número de turbinas asignadas al método numérico. En la CUADRÍCULA, los puntos son analizados sistemáticamente y se consideran sólo aquellos puntos que atienden a la restricción del problema, dado por la ecuación **(3.2)**.

En el Algoritmo Genético combinado con el Método Numérico utilizado (AG+MN), se utilizó una población de 200 *individuos*. El modelo del Operador de Selección que se aplicó fue el de la *Ruleta*, tal como se describe en [24]. La probabilidad de cruzamiento utilizada fue igual a 0,6 y la probabilidad de Mutación fue igual a 0,003. Estas probabilidades indican, de una forma estadística, cuantas veces los operadores correspondientes serán aplicados a los *individuos* de la población anterior para obtener los descendientes para la generación (iteración) siguiente.

Para el *A-Team* propuesto, la población global fue dividida en 4 subpoblaciones de 50 *individuos* cada una, asignándose a los 4 procesadores disponibles de la red de computadoras descripta. Los procesos *Esclavos* fueron aplicados sobre cada subpoblación, dejándose el quinto procesador para el proceso *Master*.

Tanto para el Algoritmo Genético combinado como para el *A-Team* propuesto se realizaron 20 corridas de cada implementación, promediándose luego los resultados. Las experimentaciones fueron realizadas para las potencias demandadas de 1500 MW, 2550 MW y 3500 MW. La siguiente nomenclatura es utilizada para describir los resultados obtenidos a partir de las experimentaciones realizadas con los algoritmos desarrollados en el presente trabajo:

η_{NM} ..... Eficiencia obtenida por el Método Numérico utilizado (MN).

t_{NM} ..... Tiempo total empleado por el MN para obtener η_{NM}.

η_{prom} ..... Promedio de la eficiencia para las 20 corridas realizadas.

t_{prom} ..... Promedio de los tiempos de ejecución para las 20 corridas.

η_{max_AG} ..... Eficiencia máxima obtenida por el AG+MN.

σ_{AG+MN} ..... Desviación típica de datos del AG+MN.

$\eta_{max\_A\text{-}Team}$ .. Eficiencia máxima obtenida por el *A-Team* propuesto.

$\sigma_{A\text{-}Team}$ ..... Desviación típica de datos del *A-Team* propuesto.

7.4 ANÁLISIS DE LOS RESULTADOS OBTENIDOS

Uno de los parámetros adicionales de entrada para el método numérico utilizado consiste en un número que indica la cantidad de subespacios que existen en la CUADRÍCULA por cada dimensión.

El término *Intensidad* indica el parámetro que establece el número de subespacios por unidad de dimensión. En la CUADRÍCULA mencionada, el método numérico de optimización realizará la búsqueda de los puntos más prometedores y sobre ellos se aplicará el método del Gradiente.

Para los resultados experimentales se utilizó una *Intensidad* igual a 9. Esto significa que la cantidad total de subespacios en la CUADRíCULA es igual a $Intensidad^{NT} = 9^{NT}$. De esta forma, la cantidad de subespacios en los que el método numérico debe realizar la búsqueda del punto óptimo crece exponencialmente con el número de turbinas en servicio.

En el método numérico utilizado, al aumentar la cantidad de subespacios en que se divide el dominio de definición de la función, el número de puntos a analizar aumenta y con ello mejora también los resultados obtenidos por el método numérico. Sin embargo, como la cantidad de puntos por analizar crece, aumentan los recursos computacionales requeridos. Esta es la razón por la cual en la práctica se utiliza una *Intensidad* de búsqueda menor. Cuanto menor es la cantidad de subespacios en que se divide el espacio de búsqueda, mayor es la velocidad de procesamiento. Pero, en contrapartida, disminuye la calidad de la solución, encontrándose casi siempre óptimos locales.

En el Algoritmo Genético combinado (AG+MN), la *Intensidad* de búsqueda para el método numérico de optimización utilizado fue igual a 3; y como se puede observar en las Tablas 7.1, 7.2 y 7.3, el AG+MN posee mayor velocidad de procesamiento que el método numérico utilizado cuando se aumenta el número de turbinas en servicio.

Se observa también una mejoría en la *calidad* de la solución (eficiencia obtenida), obteniéndose resultados ligeramente superiores a los obtenidos por el Método Numérico.

Por consiguiente, se logran mejorar tanto el tiempo de computación y la *calidad* de la solución, a medida que aumenta la complejidad del problema, al utilizar un *Team Algorithm* formado por el AG y el método numérico de optimización utilizado, incluso usando una *Intensidad* de búsqueda menor para el método numérico.

Algoritmo	**Variable**	**$N_T = 3$**	**$N_T = 4$**	**$N_T = 5$**	**$N_T = 6$**	**$N_T = 7$**	**$N_T = 8$**	**$N_T = 9$**
MN Intensidad 9	η_{MN}	85.878	85.878	85.878	85.878	85.878	85.878	–
	t_{MN} (seg.)	0.1	0.56	4.17	43.63	1040.92	56211.02	Muy Grande

Algoritmo	**Variable**	**$N_T = 3$**	**$N_T = 4$**	**$N_T = 5$**	**$N_T = 6$**	**$N_T = 7$**	**$N_T = 8$**	**$N_T = 9$**
AG+MN	$\eta_{max\text{-}AG}$	85.816	85.818	85.991	85.991	85.991	85.991	85.991
	η_{prom}	85.816	85.818	85.990	85.991	85.97	85.985	85.984
	σ_{AG}	4.6×10^{-5}	4.6×10^{-5}	4.2×10^{-5}	4.0×10^{-5}	0.00317	0.00338	0.0029
	t_{prom} (seg.)	13.61	38.256	128.256	566.71	2984.1	9018.02	9058.38
A-Team	$\eta_{max\text{-}A\text{-}Team}$	85.991	85.991	85.991	85.991	85.991	85.991	85.991
	η_{prom}	85.989	85.989	85.987	85.988	85.988	85.987	85.984
	$\sigma_{A\text{-}Team}$	0.00286	0.00351	0.00631	0.00526	0.00364	0.00557	0.00708
	t_{prom} (seg.)	8.663	16.502	75.982	171.46	223.488	1313.89	3500.18

Tabla 7.1: Resultados Experimentales para $P_D = 1500$ MW.

Algoritmo	Variable	$N_T = 3$	$N_T = 4$	$N_T = 5$	$N_T = 6$	$N_T = 7$	$N_T = 8$	$N_T = 9$
MN Intensidad 9	η_{MN}	–	87,710	87,710	87,710	87,710	87,710	–
	t_{MN} (seg.)	–	0,32	14,04	176,01	1186,54	22137,4	Muy Grande

Algoritmo	Variable	NT = 3	NT = 4	NT = 5	NT = 6	NT = 7	NT = 8	NT = 9
AG+MN	$\eta_{max\text{-}AG}$	–	87,882	87,910	87,910	87,910	87,910	87,910
	η_{prom}	–	87,881	87,893	87,893	87,906	87,904	87,909
	σ_{AG}	–	5.9×10^{-5}	0,00265	0,0378	0,0165	0,0124	0,0028
	t_{prom} (seg.)	–	18,231	33,654	63,138	149,628	854,17	3037,52
A-Team	$\eta_{max\text{-}A\text{-}Team}$	–	87,964	87,992	87,992	87,992	87,992	87,994
	η_{prom}	–	87,957	87,984	87,970	87,977	87,989	87,991
	$\sigma_{A\text{-}Team}$	–	0,0308	0,0963	0,1628	0,1164	0,0874	0,0041
	t_{prom} (seg.)	–	12,443	19,421	32,199	106,465	370,921	1759,3

Tabla 7.2: Resultados Experimentales para $P_D = 2550$ MW.

Se puede observar en las Figuras 7.1, 7.2 y 7.3, que el Método Numérico es mucho más rápido que las implementaciones propuestas para $N_T < 5$. Sin embargo, esto ocurre solo cuando la cantidad de turbinas en servicio es menor que la cantidad de turbinas que realmente están en funcionamiento en las grandes Centrales Hidroeléctricas de Generación.

Algoritmo	Variable	$N_T = 5$	$N_T = 6$	$N_T = 7$	$N_T = 8$	$N_T = 9$
MN Intensidad 9	η_{MN}	87,036	87,283	87,285	87,285	–
	t_{MN} (seg.)	3,52	49,05	1156,13	137976,74	Muy Grande

Algoritmo	Variable	$N_T = 5$	$N_T = 6$	$N_T = 7$	$N_T = 8$	$N_T = 9$
AG+MN	$\eta_{max\text{-}AG}$	87,048	87,388	87,388	87,389	87,389
	η_{prom}	87,048	87,386	87,381	87,382	87,389
	σ_{AG}	$1,02\times10^{-6}$	0,00301	0,01149	0,01211	0,00038
	T_{prom} (seg.)	26,83	45,37	331,70	1636,17	56301,45
A-Team	$\eta_{max\text{-}A\text{-}Team}$	87,048	87,388	87,389	87,389	87,389
	η_{prom}	87,048	87,385	87,379	87,383	87,389
	$\sigma_{A\text{-}Team}$	$1,02\times10^{-6}$	0,00289	0,01413	0,03114	0,0004
	T_{prom} (seg.)	23,51	39,76	136,54	474,93	3800,22

Tabla 7.3: Resultados Experimentales para $P_D = 3500$ MW.

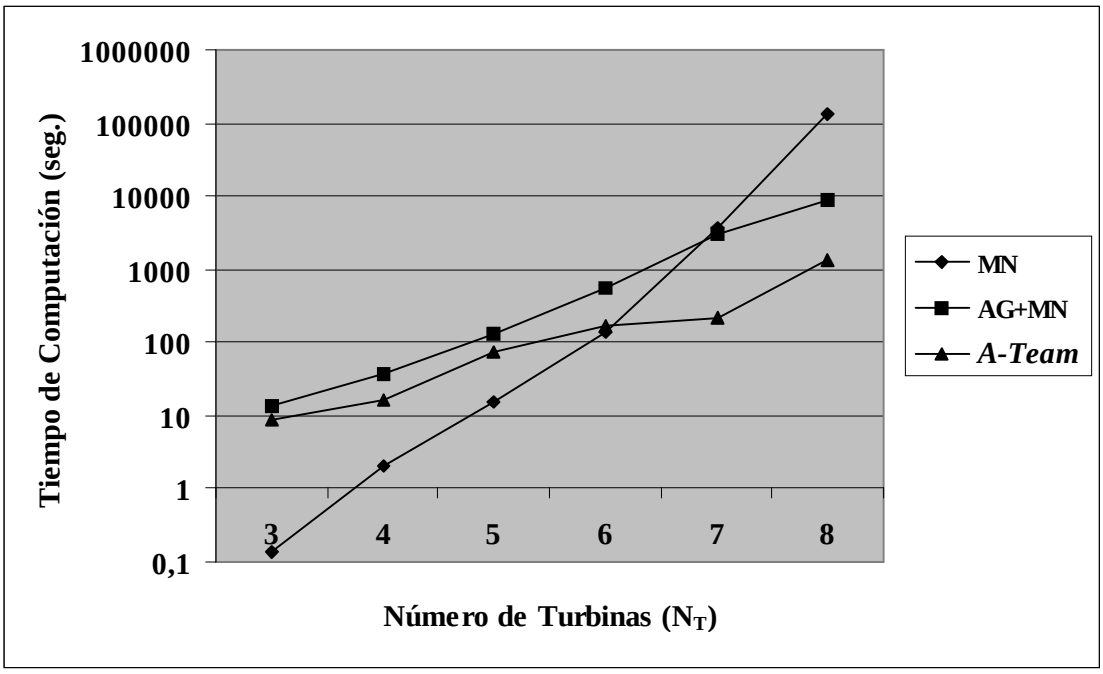


Figura 7.1: Tiempo vs. N° de Turbinas para 1500 MW.

Con todo, cuando la dimensión del espacio de búsqueda (N_T) aumenta, el problema se vuelve más complejo. En estas circunstancias, el desempeño de las implementaciones propuestas mejoran considerablemente, como puede observarse en las Figuras 7.1, 7.2 y 7.3.

De entre las implementaciones propuestas, la que mejor desempeño presenta corresponde al *A-Team* desarrollado, en donde la rapidez de procesamiento es mayor y la calidad de los resultados son mejores que las implementaciones secuenciales. En efecto, las Figuras 7.1, 7.2 y 7.3 muestran claramente que a partir de un número de turbinas $N_T > 5$, el *A-Team* es más rápido que el método numérico, y esta característica tiende a crecer a medida que el problema se vuelve más complejo.

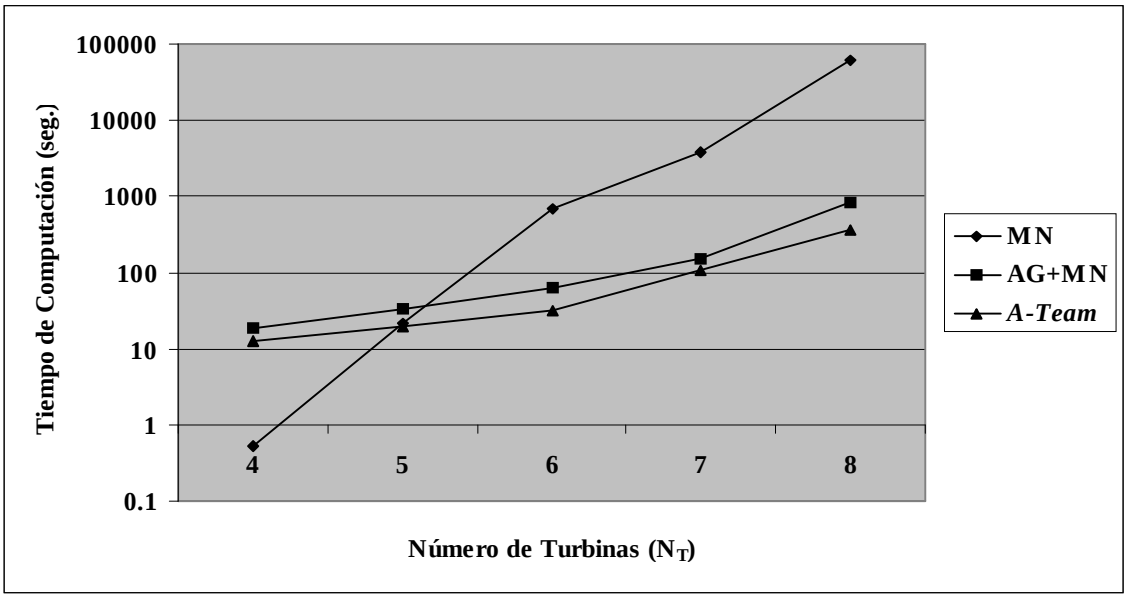


Figura 7.2: Tiempo vs. N° de Turbinas para 2550 MW.

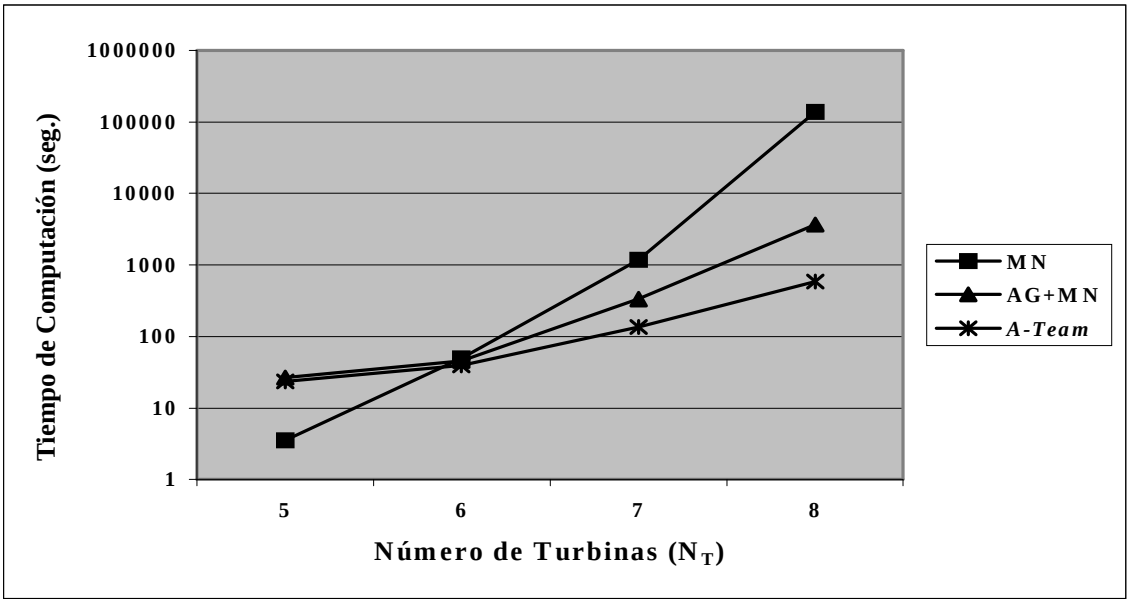


Figura 7.3: Tiempo vs. N° de Turbinas para 3500 MW.

Se observa también en las Tablas 7.1, 7.2 y 7.3, que en promedio la desviación típica del *A-Team* es mayor que la desviación típica del Algoritmo Genético combinado. Esto significa que los resultados obtenidos por el *A-Team* son más variados, y por lo tanto, más dispersos que los obtenidos por el Algoritmo Genético combinado. Al parecer, en el *A-Team* existe un acentuado comportamiento aleatorio. Sin embargo, lo que sucede es que se tiene una mayor diversidad en la búsqueda del punto óptimo debido al asincronismo, lo que favorece una mayor eficiencia en el proceso de optimización global. Esta es la razón principal del porque el *A-Team* presenta mejores desempeños que las demás implementaciones cuando el problema es más complejo.

Las Figuras 7.4, 7.6 y 7.8 se obtuvieron al promediar η_{max} (correspondiente al MN) y η_{prom} (correspondiente al AG+MN y al *A-Team*) con respecto al número de experimentaciones realizadas, para cada valor de P_D. A su vez, las Figuras 7.5, 7.7 y 7.9 fueron obtenidas promediando tiempos de ejecución con respecto al número de pruebas realizadas.

Como se puede observar claramente en las Figuras 7.4 a 7.9 el *A-Team* propuesto presenta mejores resultados (mayores valores de eficiencia) con menores tiempo de ejecución que las demás implementaciones realizadas.

Figura 7.4: Eficiencia promedio vs. implementaciones realizadas, para P_D = 1500 MW.

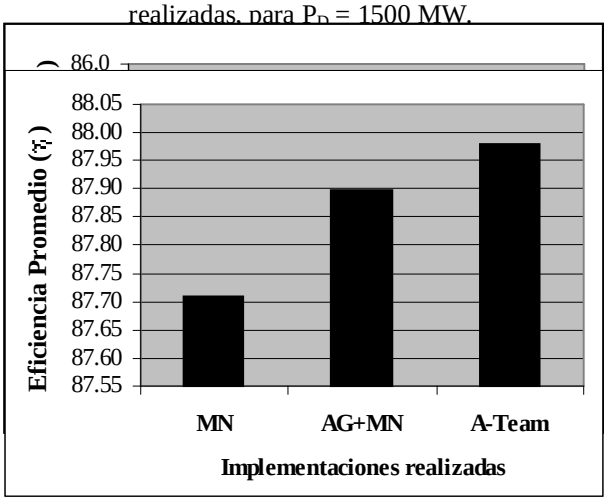


Figura 7.5: Tiempo promedio (seg) vs. implementaciones realizadas, para P_D = 1500 MW.

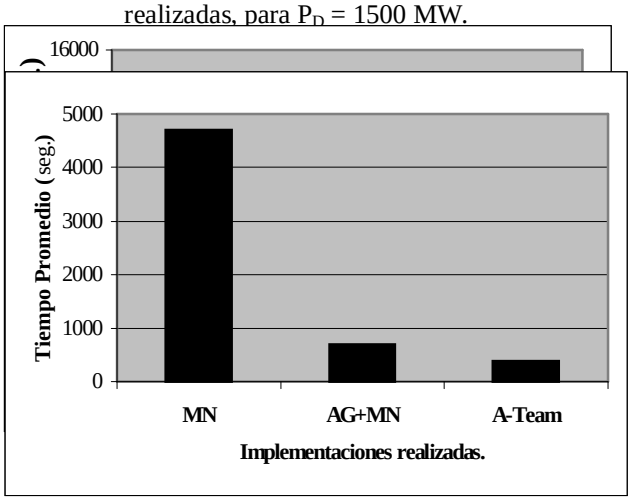
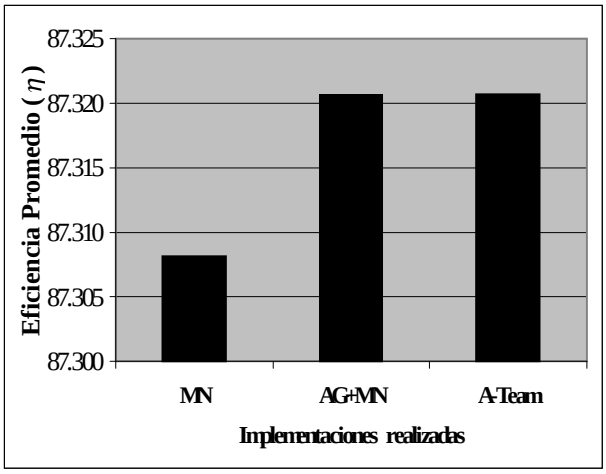


Figura 7.6: Eficiencia promedio vs. implementaciones realizadas, para P_D = 2550 MW.

Figura 7.7: Tiempo promedio (seg) vs. implementaciones realizadas, para P_D = 2550 MW.



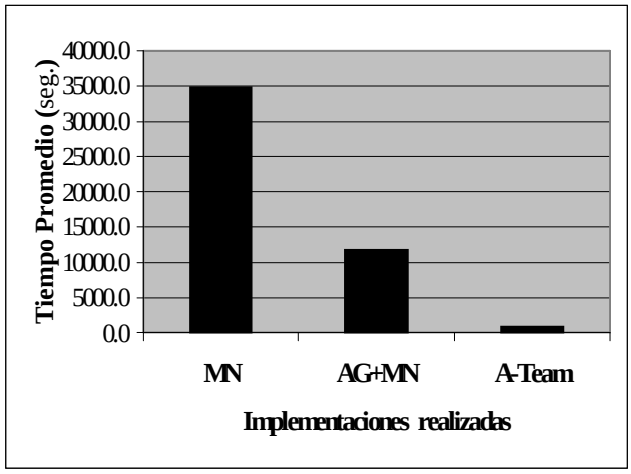


Figura 7.8: Eficiencia promedio vs. Implementaciones realizadas, para P_D = 3500 MW.

Figura 7.9: Tiempo (seg.) vs. Implementaciones realizadas, para P_D = 3500 MW.

Para el presente trabajo, la aceleración (*speedup*), que es igual al cociente entre el tiempo de ejecución secuencial del Método Numérico utilizado (T_{MN}) y el tiempo de ejecución del *A-Team* ($T_{A\text{-}Team}$) para soluciones equivalentes [9], está dada por la siguiente ecuación:

$$Speedup = \frac{T_{MN}}{T_{A\ -\ Team}} \qquad \textbf{(7.1)}$$

La aceleración (*speedup*) mide y compara el aumento en la velocidad de cálculo de una implementación con respecto a la otra.

Comparando la aceleración para diferentes valores de potencia demandada (P_D), se puede observar una vez más en la Figura 7.10 la superioridad del *A-Team* en la medida que crece la complejidad del problema. Así, la aceleración aumenta no solo con el tamaño del problema, sino también con el valor de P_D. Esto porque en la medida que se demande mayor generación de energía, más turbinas deben entrar a operar efectivamente, e inclusive algunas pueden necesitar operar en condiciones extremas de generación en alta carga, con alta alinealidad. Por consiguiente crece la complejidad del problema, lo que crea un escenario apropiado para la implementación de un *A-Team*.

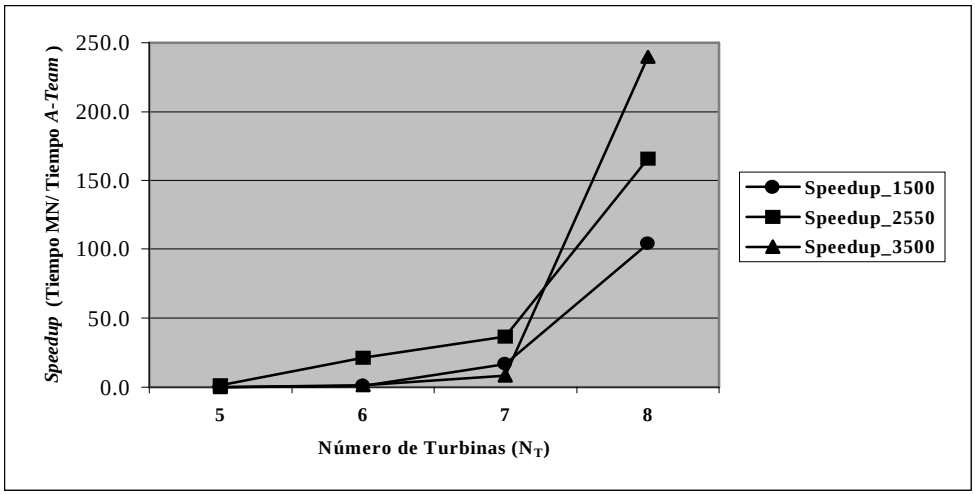


Figura 7.10: Aceleración (*Speedup*) vs. Número de turbinas.

CAPÍTULO 8: CONCLUSIONES

En el contexto actual, la creciente demanda de energía de los países debido al aumento del número de usuarios (debido a la políticas de electrificación) y la implementación de nuevas y grandes industrias, exigen la construcción de represas hidroeléctricas cada vez más grandes, conteniendo gran número de unidades generadoras, siendo éstas a su vez de mayor potencia, con el fin de abastecer satisfactoriamente a los centros de consumo. En este sentido, nuestro país posee dos grandes represas hidroeléctricas de gran porte: Itaipú y Yacyretá, gracias a las cuales el Paraguay se constituye en un gran exportador de energía eléctrica.

Por consiguiente, debido a las dificultades de implementación de una represa hidroeléctrica de tal magnitud (desde el punto de vista técnico y financiero), surge la necesidad minimizar la cantidad de caudal a ser turbinado en las unidades generadoras para aumentar de esta manera la disponibilidad de energía (cantidad de agua mantenida en el embalse) y disminuir los costos de generación.

En este contexto, la implementación del *A-Team* propuesto en este trabajo, de acuerdo a los resultados experimentales analizados en el capítulo anterior, presenta numerosas ventajas (en cuanto a la rapidez de respuesta y calidad de la solución obtenida) que superan a los métodos numéricos tradicionales. De ahí que, el *A-Team* propuesto promete ser una herramienta útil para el problema de minimización del Caudal Total turbinado de una Central Hidráulica de Generación con múltiples unidades generadoras.

8.1 PUNTO DE VISTA TÉCNICO

Entre las ventajas que resultan de la implementación del método propuesto (*A-Team*) se resaltan las siguientes:

- De acuerdo a las Figuras 7.1, 7.2 y 7.3 (Capítulo 7), el *A-Team* implementado disminuye considerablemente el tiempo de cálculo requerido para optimizar la eficiencia η en la generación de una Represa Hidroeléctrica. Se observa que esta mejora tiene lugar para un número de turbinas $N_T \geq 5$ como se muestra en las Figuras mencionadas. Sin embargo, de las Tablas 7.1, 7.2 y 7.3 se puede concluir también que el Método Numérico utilizado posee mejor desempeño que el *A-Team* cuando el número de turbinas es $N_T < 5$, número inferior a la cantidad de turbinas utilizadas en represas hidroeléctricas de gran envergadura.
- No se necesita invertir en costosas supercomputadoras (o *mainframes*), de mayor capacidad y velocidad de procesamiento para implementar métodos numéricos de optimización tradicionales. El *A-Team* se puede implementar en una red formada por computadoras personales (de mucho menor costo), lográndose menores tiempos de procesamientos y alta calidad en los resultados [6].

Las restricciones operativas en una represa hidroeléctrica de generación están principalmente determinadas por las posibles variaciones en la altura del embalse y la necesidad de parar determinadas unidades generadoras para mantenimiento y/o reparación. Además, las unidades generadoras pueden llegar a tener franjas prohibidas de generación debido a que en dicha franja el nivel de vibración en la turbina es demasiado alto y puede llegar a producir fallas o fatigas mecánicas [36].

El *A-Team* desarrollado está implementado de tal manera que solo necesita como parámetros de entrada, para el algoritmo, las curvas de eficiencia de las turbinas en servicio, el número de turbinas en servicio, la altura del embalse y la potencia demandada. No es necesario realizar una modificación completa del algoritmo por cada nuevo conjunto de parámetros de entrada. En otras palabras, el *A-Team* diseñado es un algoritmo adaptable.

De esta forma, el método propuesto mejora el proceso de minimización del Caudal total que debe ser turbinado, encontrándose muy buenos resultados en menores tiempos de ejecución. Incluso la implementación propuesta, debido al efecto sinergético, puede llegar a encontrar soluciones óptimas no encontradas por el Método Numérico. Esto se observa en la Tabla 7.2 en la que para $N_T = 9$ se llega a tener una eficiencia de 87.994 %.

Además, una ventaja muy clara del *A-Team* propuesto se presenta cuando se tiene en servicio un gran número N_T de unidades generadoras y se necesita generar una potencia requerida (P_D) muy alta. De acuerdo a las Figuras 7.1, 7.2 y 7.3, el método numérico (MN) invertiría un tiempo considerablemente superior en realizar el cálculo, debido a las características exponenciales de las curvas de respuesta del MN. Sin embargo, el *A-Team* tardaría mucho menos en realizar dichos cálculos, para la misma configuración. Además, los resultados serían superiores en calidad a los obtenidos por el método numérico como se observan en las Tablas 7.1, 7.2 y 7.3.

En síntesis, el presente trabajo presenta una alternativa de optimización robusta cuando el problema que se trate es de gran porte, como es el caso de una central hidroeléctrica con numerosas turbinas en funcionamiento y que desee generar energía eléctrica optimizando la utilización de recursos hídricos disminuyendo costos de generación.

8.2 PUNTO DE VISTA ECONÓMICO

La función de una central hidráulica de generación es la producción de energía para su posterior comercialización, suministrando energía a los consumidores a un costo atractivo y con un nivel de calidad aceptable. Para lograr que el costo sea atractivo, se pretende generar la mayor cantidad de energía con los mismos recursos existentes.

De esta manera el costo de producción de una central hidroeléctrica está determinada por los siguientes componentes: el valor monetario del agua (energía potencial del agua en el embalse), costo estimado de mantenimiento y/o reparación y la demanda periódica de energía [36]. De esta forma, el objetivo de la presente sección es la de realizar una estimación de las ventajas económicas que implicaría la implementación del algoritmo propuesto (*A-Teams*). La estimación económica consistirá en el costo de la energía adicional que podría generar la usina con los mismos recursos (unidades generadoras, agua, etc.) al utilizar mejores algoritmos de optimización para planear la generación de energía.

En vista del mejor desempeño (velocidad de computación y calidad de la solución obtenida) del *A-Team* desarrollado, con respecto a las otras implementaciones realizadas, el cálculo de estimación económica se realizará entre el *A-Team* y el método numérico (MN).

Para la estimación económica, se utilizará el valor monetario equivalente a la energía potencial del recurso hídrico mantenido en el embalse que fuera presentado en [36], debido a que la represa hidroeléctrica utilizada como referencia, en dicho artículo, presenta características similares al problema ejemplo utilizado en el presente trabajo. El valor monetario equivalente en el mencionado trabajo es igual a 14 U$S/MWh, equivalente a 10,08 U$S/KW mes (considerando un 1 mes = 30 días).

Utilizando las Tablas 7.1, 7.2 y 7.3 se obtienen las Tablas 8.1, 8.2 y 8.3 respectivamente. En estas Tablas tenemos los caudales totales mínimos correspondientes a las eficiencias máximas obtenidas por el método numérico utilizado (MN) y el *A-Team* implementado.

Por ejemplo, en la Tabla 8.1, el caudal 1390.8 m^3/seg. se obtiene utilizando la ecuación **6.1** en la que se tienen como datos**:** la eficiencia η (de la Tabla 7.1), la potencia demandada (P_D), el peso específico del líquido (δ) y la altura del embalse (h). De la misma forma se procede para el cálculo de los demás caudales. Una vez calculados todos los caudales, se obtiene la diferencia de caudales ΔQ [m^3/seg.] entre el caudal correspondiente al MN y el caudal correspondiente al *A-Team*.

Algoritmo	**N_T = 3**	**N_T = 4**	**N_T = 5**	**N_T = 6**	**N_T = 7**	**N_T = 8**	**N_T = 9**
Q_{MN} (m^3/seg.)	1390,8	1390,8	1390,8	1390,8	1390,8	1390,8	–
$Q_{A\text{-}Team}$ (m^3/seg.)	1388,972	1388,972	1388,972	1388,972	1388,972	1388,972	1388,972
ΔQ (m^3/seg.)	1,828	1,828	1,828	1,828	1,828	1,828	–
Potencia Equivalente[kW]	1973,73	1973,73	1973,73	1973,73	1973,73	1973,73	–
U$S/mes	19895,2	19895,2	19895,2	19895,2	19895,2	19895,2	–

Tabla 8.1: Cálculo del Ahorro en el Costo de Generación (U$S/mes) para P_D = 1500 MW.

Luego de calcular la diferencia de caudal ΔQ [m^3/seg.], se obtiene la potencia que se podría generar con dicho caudal. Dicha potencia representa la energía adicional que podría generarse utilizando los mismos recursos (cantidad de unidades generadoras, caudal de agua, altura del embalse, etc.) utilizando el *A-Team* para planificar la generación de energía, y representa un ahorro en el costo de generación al compararse con la utilización del método numérico para producir la misma potencia (P_D) utilizando la misma configuración.

Algoritmo	**N_T = 4**	**N_T = 5**	**N_T = 6**	**N_T = 7**	**N_T = 8**	**N_T = 9**
Q_{MN} (m^3/seg.)	2314,98	2314,98	2314,98	2314,98	2314,98	–
$Q_{A\text{-}Team}$ (m^3/seg.)	2308,29	2307,56	2307,56	2307,56	2307,56	2307,50
ΔQ (m^3/seg.)	6,685	7,419	7,419	7,419	7,419	–
Potencia Equivalente[kW]	7384,563	8198,609	8198,609	8198,609	8198,609	–
U$S/mes	74436,393	82641,979	82641,979	82641,979	82641,979	–

Tabla 8.2: Cálculo del Ahorro en el Costo de Generación (U$S/mes) para P_D = 2550 MW.

La Potencia Equivalente (kW), de las referidas Tablas, se obtiene aplicando la ecuación **6.1**, teniendo como parámetros: la eficiencia máxima del *A-Team* ($\eta_{max\text{-}A\text{-}Team}$, de las Tablas 7.1, 7.2 y 7.3) considerando que se está utilizando el *A-Team*, la diferencia de caudal ΔQ (m^3/seg.), el peso específico (δ) y la altura del embalse (h). Finalmente, se calcula el Ahorro en el costo de generación multiplicando la Potencia equivalente (kW) por el equivalente monetario de la energía potencial en el embalse dado en [36] (10,08 U$S/kW mes).

Algoritmo	**N_T = 5**	**N_T = 6**	**N_T = 7**	**N_T = 8**	**N_T = 9**
Q_{MN} (m^3/seg.)	3202,02	3192,96	3192,89	3192,88	–
$Q_{A\text{-}Team}$ (m^3/seg.)	3201,58	3189,11	3189,11	3189,08	3189,08
ΔQ (m^3/seg.)	0,441	3,848	3,782	3,795	–
Potencia Equivalente[kW]	482,559	4223,635	4150,839	4165,379	–
U$S/mes	4864,194	42574,237	41840,453	41987,024	–

Tabla 8.3: Cálculo del Ahorro en el Costo de Generación (U$S/mes) para P_D = 3500 MW.

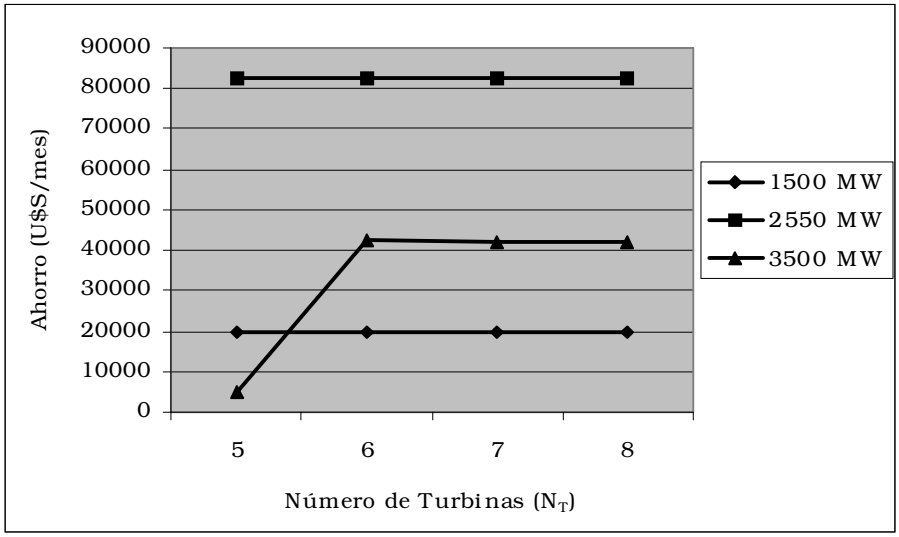


Figura 8.1: Ahorro (U$S/mes) vs. Número de Turbinas (N_T).

De lo expuesto, se observa que la ventaja de implementar el *A-Team* en lugar de métodos numéricos de optimización tradicionales, cuando el número de unidades generadoras crece ($N_T \geq 5$), no solo se manifiesta en la capacidad de encontrar mejores resultados en menores tiempos de computación, sino también en la capacidad de disminuir los costos de generación al utilizar mejor los recursos hídricos de una represa hidroeléctrica.

Como se observa en las Tablas 8.1, 8.2 y 8.3, la disminución en los costos de generación se debe a la reducción en ΔQ (m^3/seg.), mediante el cálculo realizado por el *A-Team*, del caudal turbinado. El valor ΔQ (m^3/seg.), resulta en una potencia adicional que es capaz de generar la usina, usando los mismos recursos disponibles en la represa, lo que corresponde a un Ahorro en los gastos de producción de energía (U$S/mes) por parte de la Central.

En la Figura 8.1, observamos que el Ahorro en los costos de generación (U$S/mes) se mantienen aproximadamente constantes dependiendo de la potencia eléctrica a generar (P_D). Por consiguiente, el *A-Team* desarrollado promete ser una muy buena herramienta de cálculo para la planificación de la generación de energía porque el punto de operación óptimo de la Central es computado en menor tiempo que las técnicas matemáticas tradicionales y, además, con su implementación podrían reducirse considerablemente los costos de generación.

REFERENCIAS

[1] Información disponible en la siguiente dirección URL: `http://www.itaipu.gov.br/dt/dispen.htm.`

[2] Información disponible en la siguiente dirección URL: `http://www.infonet.com.py/ande/gen/gen.htm.`

[3] Simão F. J., Henning J., "Una introducción al MONDIG". *III Seminario del Sector Eléctrico Paraguayo* (*III SESEP*). Asunción – Paraguay, 1998.

[4] Wood A. J., Wollenberg B. F., *Power Generation, Operation & Control*, John Wiley & Sons, 1983.

[5] Barán B., *Estudio de Algoritmos Combinados Paralelos Asíncronos*. Tesis Doctoral. Universidad Federal de Río de Janeiro – COPPE/UFRJ, Río de Janeiro - Brasil. Octubre de 1993.

[6] Barán B., "Métodos combinados para Sistemas Distribuidos", *Artículo publicado en la Revista de la Sociedad Científica del Paraguay*, AÑO I, TERCERA ÉPOCA, N° 2, junio 1997.

[7] Souza P. S., Talukdar S. N., "Genetic Algorithms in Asynchronous Teams". *Proceedings of the Fourth International Conference on Genetic Algorithms* (*ICGA – 91*), pp. 392 – 397, San Diego – California, 1991.

[8] Mühlenbein H., Schomisch M., Born J., "The Parallel Genetic Algorithm as Function Optimizer". *Proceedings of the Fourth International Conference on Genetic Algorithm* (*ICGA – 91*), pp. 271 – 278, San Diego – California, 1991.

[9] Barán B., Kaszkurewicz E., Bhaya A., "Parallel Asynchronous Team Algorithms: Convergence and Performance Analysis". *IEEE Transactions on Parallel and Distribuited Systems*, Vol. 7, N° 7, 1996.

[10] Barán B., Cáceres N., Chaparro E., "Reducción del Tiempo de Búsqueda utilizando una Combinación de Algoritmos Genéticos y Métodos Numéricos". *III Encuentro Chileno de Computación*, Arica – Chile, 1995.

[11] Barán B., Chaparro E., "Algoritmos Asíncronos Combinados en un Ambiente Heterogéneo de Red". *XXIII Conferencia Latinoamericana de Informática* (*CLEI – PANEL´98*), pp. 367 – 376, Valparaíso – Chile, 1997.

[12] Press W. H., Flannery B. P., Teukolsky S. A., Vetterling W. T., *Numerical Recipes in C*: *The art of Scientific computing*, Cambridge University Press, 1988.

[13] Dusonchet Y. P., Talukdar S. N., Sinnot H. E., "Load Flows using a combination of Point Jacobi and Newton´s Method". *IEEE Transactions on Power Apparatus and Systems*, PAS-90, pp. 941 – 949, 1971.

[14] Marschak J., Radner R., *Economic Theory of Teams*. New Haven, CT: Yale University Press, 1972.

[15] Talukdar S. N., Pyo S. S., Giras T. C., "Asynchronous Procedures for Parallel Processing". Technical Report DRC – 18-54-82, Carnegie-Mellon University, Pittsburgh – Pennsylvania, 1982.

[16] Talukdar S. N., Pyo S. S., Mehrotra R., "Designing Algorithms and Assigments for Distribuited Processing". *Electric Power Research Institute*. *Final Report*. Carnegie-Mellon University, Pittsburgh – Pennsylvania, 1983.

[17] Talukdar S. N., Ramesh V. C., Nixon J. C., "A Distribuited System of Control Specialist for Real-Time Operations". *Proceedings of the Third Symposium on Expert Systems Applications to Power Systems*. Japan, 1991.

[18] Fox M. S., "An Organizational View of Distribuited Systems". *IEEE Transactions on Systems, Man and Cybernetics*, Vol. SMC-11, N° 1, pp. 70 – 80, 1981.

[19] Ramesh V. C., Quadrel R., de Souza P., Talukdar S. N., "Asynchronous Teams: An Organizational Model for Distribuited Problem Solving".*1991 Summer National Meeting, American Institute of Chemical Engineers*, Pittsburgh, 1991.

[20] David W. Rolston, *Principios de Inteligencia Artificial y Sistemas Expertos*. Editorial McGRAW HILL LATINOAMERICANA S. A., 1990.

[21] Tanomaru J., "Motivação, Fundamentos e Aplicações de Algoritmos Genéticos". *II Congreso Brasileiro de Redes Neurais, III Escola de Redes Neurais*. Curitiba – Brazil, 1995.

[22] Freeman J. A., Skapura D. M., *Redes Neuronales*: *Algoritmos*, *Aplicaciones y Técnicas de Programación*. Editorial Addison – Wesley Iberoamericana S. A., Wilmington, Delaware – EEUU, 1993.

[23] Bäck T., Hammel U., Schwefel H. P., "Evolutionary Computation: Comments on the History and Current State". *IEEE Transactions on Evolutionary Computation*, Vol. 1, N° 1, April 1997.

[24] Goldberg D. E., *Genetic Algorithm in Search, Optimization & Machine Learning*. Addison Wesley, 1989.

[25] Holland J. H., *Adaptation in Natural and Artificial Systems*, University of Michigan Press, 1975.

[26] DeJong K. A., *An Analysis of the Behavior of a class of Genetic Adaptive Systems*, Doctoral Thesis, University of Michigan, 1975.

[27] Fogel L. J., Owens A. J., Walsh M. J., *Artificial Intelligent Through Simulated Evolution*, John Wiley, 1966.

[28] Rechenberg I., "Evolution Strategy". In J. M. Zurada, R. J. Marks II, and C. J. Robinson (eds.), *Computational Inelligence – Imitating Life*, IEEE Press, pp. 147 – 159, 1994.

[29] Koza J. R., *Genetic Programming*: *On the Programming of Computers by means of Natural Selection*, MIT Press, 1992.

[30] Belew R., Forrest S., "Learning and Programming in Classifier Systems", *Machine Learning*, Vol. 3, pp. 193 – 223, 1988.

[31] Debernardi E., "Experiencia del Paraguay en Proyectos Electroenergéticos Binacionales". *Anales del Foro Regional de Energía del Cono Sur*, pp. 119 – 125, Asunción – Paraguay, 1995.

[32] Falcão D. M., A. Do Bomfim L. B., Dornellas C. R. R., Taranto G. N., "Genetic Algorithms in Power Systems Optimization". *V Symposium of Specialist in Electric Operational and Expansion Planning* (*SEPOPE*), Recife – Brazil, May 1996.

[33] Ota H., Uesugi M., Kashiwakura M., "Integrated Computer Systems for Electric Power Management". *Information and Control Technology of Power Systems*, HITACHI REVIEW Vol. 41, N° 3, August 1992, pp. 119 – 124.

[34] Yuji Ito, Toshihide Aramaki, Tomio Suzuki, "Total Control System Supporting Combined Operation of Hidraulic Power and Dam Systems". *Information and Control Technology of Power Systems*, HITACHI REVIEW Vol. 41, N° 3, August 1992, pp. 145 – 152.

[35] Koji Ando, Toshiyuki Oguro, Akiji Hasegawa, Yuji Nakata, "Tele-control System with intelligent Functions". *Information and Control Technology of Power Systems*, HITACHI REVIEW Vol. 41, N° 3, August 1992, pp. 125 – 152.

[36] Nilsson O., Sjelvgen D., "Variable Splitting Applied to Modelling of Start-Up Costs in Short Term Hydro Generation Schedulling". *IEEE Transactions on Power Systems*, pp. 1 – 6, June 1996.

[37] Wong K. P., Wong Y. W., "Combined Genetic Algorithm/Simulated Annealing/Fuzzy Set Approach to Short – Term Generation Scheduling with Take-or-Pay Fuel Contract". *IEEE Transactions on Power Systems*, Vol. 11, N° 1, February 1996.

[38] Wong K. P., Wong Y. W., "Short-term Hydrothermal Scheduling Part I: Simulated Annealing Approach". To appear in *IEE Proceedings Part C*, 1994.

[39] Darwin C. R., *On the Origins of Species by Means of Natural Selection*, Penguin Classics, 1985.

[40] Holland J. H., *Adaptation in Natural and Artificial Systems*, University of Michigan Press, 1975.

[41] Fletcher R., *Practical Methods of Optimization*. John Wiley & Sons, 2nd Edition, 1990.

[42] Ginsberg M., *Essentials of Artificial Intelligence*. Morgan Kauffmann Publishers. San Francisco – California, 1992.

[43] Cantú – Paz E., "A Summary of Research on Parallel Genetic Algorithms". *IlliGAL Technical Report No. 95007*, July 1995.

[44] Mejía M., Cantú E., "DGENESIS: Software para la Ejecución de Algoritmos Genéticos Distribuidos". *XX Conferencia Latinoamericana de Informática* (*CLEI – PANEL´94*), Mexico 1994.

[45] Tanese R., “Parallel Genetic Algorithm for a Hypercube”. In John Grefenstette Editor. *Proceedgins of the Second International Conference on Genetic Algorithms*, Lawrence Erlbaum Associates Publishers, 1987.

[46] Grosso P., *Computer Simulations of Genetic Adaptation*: *Parallel Subcomponent Interaction in a Multilocus Model*. Phd Thesis, University of Michigan, 1985.

[47] Tanese R., “Distribuited Genetic Algorithms”. In D. Schaffer Editor. *Proceedings of the Third International Conference on Genetic Algorithms*. Morgan Kauffmann Publishers, 1989.

[48] Cantú-Paz E., Mejía Olvera M., “Experimental Results in Distribuited Genetic Algorithms”. *International Symposium on Applied Corporate Computing*, pp. 99 – 108, Monterrey – Mexico, 1994.

[49] Anciano J. L., Arráiz E, Savino N., “Análisis de Paralelización en un Algoritmo Genético para Diseño de Antenas”. *XXII Conferencia Latinoamericana de Informática* (*CLEI PANEL´*96). Santa Fé de Bogotá – Colombia, 1996.

[50] Información disponible en la siguiente dirección URL:
`http://www.cs.cmu.edu/afs/cs/project/edrc-22/project/ateams/WWW/home_page.html.`

[51] Componogara E., Souza P. S., “*A-Teams* for an Oil Transportation Problem through Pipelines”. *International Conference on Information Systems Analysis and Synthesis* (*ISAS´96*), Orlando – EEUU, 1996.

[52] Souza P. S., Favilla J. R., “Asynchronous Teams for Steel Industry: Mini Mills”. *International Conference on Information Systems Analysis and Synthesis* (*ISAS´96*), Orlando – EEUU, 1996.