UNIVERSIDAD AUTÓNOMA DEL ESTADO DE MORELOS

FACULTAD DE CIENCIAS QUÍMICAS E INGENIERÍA

CENTRO DE INVESTIGACIÓN EN INGENIERÍA Y CIENCIAS APLICADAS

ALGORITMO GENÉTICO DISTRIBUIDO, PARA OPTIMIZAR LA RED HIDRÁULICA DEL FRACCIONAMIENTO REAL MONTECASINO DEL MUNICIPIO DE HUITZILAC

TESIS PROFESIONAL
PARA OBTENER EL GRADO DE:

DOCTOR EN INGENIERÍA Y CIENCIAS APLICADAS CON OPCIÓN TERMINAL EN TECNOLOGÍA ELÉCTRICA

P R E S E N T A:

M.I.C.A. BEATRIZ MARTÍNEZ BAHENA

ASESOR: DR. MARCO ANTONIO CRUZ CHÁVEZ
COASESOR: DR. MARTÍN HERIBERTO CRUZ ROSALES

CUERNAVACA, MORELOS MARZO 2016

Resumen

En este trabajo de investigación se presenta un algoritmo genético que ofrece una solución al problema de la deficiencia en la distribución de agua potable de la red hidráulica actual del Fraccionamiento Real Montecasino (FRM), ubicada en el municipio de Huitzilac, Morelos. Se propone una metodología de solución la cual implica agregar nuevos elementos a la red del FRM. Estos elementos son tanques de almacenamiento, nuevas tuberías y válvulas reguladoras de presión. Para evaluar las restricciones del modelo que representa la red del FRM se utiliza el simulador Epanet, el cual ha sido utilizado en infinidad de trabajos a nivel mundial. El algoritmo genético evalúa el modelo de optimización que minimiza el costo de agregar nuevos elementos al sistema. La factibilidad de la solución aportada por el algoritmo genético es validada por Epanet. Se utiliza una metodología de sintonización, la cual permite realizar el análisis de sensibilidad de los parámetros de control del algoritmo genético, esto con la finalidad de que el algoritmo mejore el desempeño tanto en eficiencia como en eficacia. El algoritmo genético se desarrolla en dos versiones, secuencial y distribuida. Para la versión distribuida se implementa el modelo Maestro-Esclavo utilizando comunicación colectiva mediante la biblioteca MPI.

El análisis de los resultados obtenidos por el algoritmo genético para la red del FRM muestra que se obtuvieron en promedio presiones adecuadas y balanceadas, mediante pequeñas modificaciones en la red a un costo mínimo. También se observa que se logra un buen suministro de agua dadas las presiones obtenidas en cada nodo de la red. Las pruebas experimentales del algoritmo fueron realizadas en el equipo del Laboratorio de Optimización, ubicado en el Centro de Investigación en Ingeniería y Ciencias Aplicadas de la UAEM.

Abstract

This research work presents a genetic algorithm that provides a solution for the problem of deficient distribution of drinking water to the current hydraulic network in the Fraccionamiento Real Montecasino (FRM), at Huitzilac, Morelos. For solution, a methodology that implicates adding new elements to FRM network is proposed. These elements are storage tanks, new pipelines and pressure reducing valves. To evaluate the FRM model restrictions Epanet simulator is used, this has been employed in a great number of works around the world. The genetic algorithm evaluates the optimization model that minimizes the cost when new elements are added to the system. The feasibility of solution obtained by the genetic algorithm is validated by Epanet. It employs an attunement methodology, which allows carry out the sensibility analysis of control parameters of the genetic algorithm to increase its efficiency and effectiveness. The genetic algorithm is developed in two versions, sequential and distributed. For distributed version a master-slave model is implemented, using collective communication by MPI library.

The analysis of the results obtained by the genetic algorithm for FRM network shows that in average adequate and balanced pressures were obtained by means of small modifications on the network at a reduced cost. It is also showed that a good supply of water is achieved, given the pressures obtained in each network node. Experimental tests of the algorithm were performed in laboratory equipment Optimization, located at the Center for Research in Engineering and Applied Sciences of UAEM.

Agradecimientos

A Dios porque me ha permitido despertar cada mañana. Gracias a Dios por darme la oportunidad de cumplir con mis objetivos y metas planteadas en mi vida.

A CONACYT por brindarme el apoyo económico para ser posible la realización de mis estudios de doctorado.

A mi asesor Dr. Marco A. Cruz Chávez por el tiempo dedicado, por su paciencia y apoyo incondicional para dirigir este trabajo de investigación.

A mis compañeros y amigos Jazmín, Yessica, Jorge y Christian con los que he compartido grandes momentos, por su gran apoyo incondicional.

A mis compañeros y amigos del cuerpo académico de Optimización y Software (caos).

Dedicatoria

A mis padres Víctor Martínez Romero y Francisca Bahena Fuentes porque ellos fueron la parte esencial. Gracias a su gran esfuerzo, apoyo y sacrificio para lograr que se cumplieran mis metas.

A mis sobrinos Yatziry Lizzet, Leslie Evelyn y Alex Jhovany porque gracias a sus lindas sonrisas, pequeños abrazos y dulces besos me impulsaron a seguir adelante para que algún día pueda ser un gran apoyo en su vida.

.

A mis hermanos Héctor, Elizabeth, Marcial, Gustavo y Christian por su apoyo incondicional y por todos esos bellos momentos que hemos pasado junto.

A toda mi familia porque gracias a ellos pude lograr mi formación profesional y personal, con su sacrificio, apoyo y dedicación pude concluir una etapa más en mi vida.

A Luis Alberto Rubio Saavedra por su apoyo incondicional, por estar conmigo en las buenas y en las malas, por sus acertados consejos para seguir adelante con este trabajo, por siempre darme ánimos y sacarme una sonrisa en cualquier momento y por todos esos increíbles momentos que hemos pasado juntos.

Nomenclatura

Nomenclatura general

WDNs	Siglas en inglés para redes de distribución de agua (Water Distribution Networks)
PRV	Siglas en inglés para válvulas reguladoras de presión (Pressure Reducing Valves)
SA	Siglas en inglés para recocido simulado (simulated annealing)
TS	Siglas en inglés para búsqueda tabú (tabu search)
ILS	Siglas en inglés para búsqueda local iterada (iterated local search)
VNS	Siglas en inglés para búsqueda local variable (variable local search)
EAs	Siglas en inglés para algoritmos evolutivos (evolutionary algorithms)
ACO	Siglas en inglés para optimización de colonia de hormigas (ant colony optimization)
PSO	Siglas en inglés para optimización por enjambre de partículas(particle swarm optimization)
SS	Siglas en inglés para búsqueda dispersa(scatter search)
DE	Siglas en inglés para evolución diferencia (differential evolution)
ES	Siglas en inglés para estrategias evolutivas(evolutionary strategies)
MPI	Siglas en inglés para Interface de paso de mensajes (Message Passing Interface)
AGS-FRM	Siglas para Algoritmo genético secuencial para la red del Fraccionamiento Real Montecasino

AGD-FRM Siglas para Algoritmo genético distribuido para la red del Fraccionamiento Real Montecasino

Problema red de distribución de agua

nd Número de diámetros de la lista de diámetros comerciales

C_i Costo de la tubería con diámetro i, tomado de una lista de diámetros comerciales

L_{ij} Longitud de cada tubería j con diámetro i

nT Número de tanques de un conjunto de tanques con diferentes características

C_T Costo del tanque T agregado a la red

nV Número de válvulas

C_V Costo de la válvula V utilizada en cada tubería

$d_{\min}$ Diámetro mínimo

$d_{\max}$ Diámetro máximo

d_{ij} Diámetro de la tubería j

Q_{in} Caudal que entra en un nodo

Q_{out} Caudal que sale de un nodo

Q_e Caudal de demanda

h_f Pérdidas de energía por fricción

E_p Energía de potencia suministrada por una bomba

H_i Presión en un nodo i

H_{min} Presión mínima

H_{max} Presión máxima

nP	Total de tuberías
p	Tubería correspondiente a cada tanque

Algoritmo genético

$\mathrm{P_s}$	Porcentaje de selección
$\mathrm{P_m}$	Porcentaje de mutación
$\mathrm{T_{pob}}$	Tamaño de población
$\mathrm{P_c}$	Probabilidad de cruzamiento
$\mathrm{P_{ini}}$	Población inicial
$\mathrm{P_{mut}}$	Probabilidad de mutación
$\mathrm{N_g}$	Número de generaciones
N_{inds}	Número de individuos

TABLA DE CONTENIDO

Índice de Figuras

Índice de Tablas

Índice de Algoritmos

Capítulo 1

Introducción

Capítulo 1 Introducción

1.1 Problema de distribución de agua

Las redes de distribución de agua (WDNs) son uno de los problemas más importantes para la sociedad. Sin agua el ser humano no podría sobrevivir, por lo cual es de gran interés contar con redes de distribución de agua que permitan satisfacer las necesidades de los usuarios y no ocasionen pérdidas económicas. Una red de distribución de agua juega un papel importante en mejorar el nivel de vida en una comunidad, comercios públicos e industrias.

El problema consiste en una red que contiene elementos hidráulicos como son tuberías, bombas, válvulas, fuentes de abastecimiento (depósitos o tanques), los cuales permiten una adecuada distribución de agua. Las tuberías permiten llevar el agua desde las fuentes de abastecimiento hasta a los puntos de consumo (viviendas, comercios, industrias, hidrantes de riego e incendio). Las fuentes de abastecimiento son fuentes externas o sumideros para un sistema, estas pueden ser pozos, ríos, arroyos o conexiones a otros sistemas. Las válvulas ayudan a tener un control adecuado de la presión en un sistema. Las bombas son elementos que permiten elevar el agua desde fuentes superficiales o subterráneas a plantas de tratamiento, almacenamientos, o directamente al sistema de distribución (Rossman, 2000; CNA, 2007).

El problema puede abordarse desde diferentes etapas: Diseño, Operación, Rehabilitación y Mantenimiento (Cruz-Chávez et al., 2009).

El diseño de una red tiene como objetivo suministrar agua a todas las áreas que satisfaciendo las demandas y la presión. Encontrando la mejor configuración de diámetros de las tuberías en el menor costo posible.

La operación en una red de distribución de agua consiste en suministrar agua a los usuarios de una manera eficiente, cumpliendo con las restricciones de las presiones mínimas requeridas por los usuarios, velocidades en los fluidos de las tuberías de la red (Cruz-Chávez et al., 2009), esto para tener un mejor control del abastecimiento y su distribución.

La rehabilitación en una red existente se presenta cuando las presiones en ésta adquieren valores inadecuados, ocasionados por el incremento de la demanda o mayores caudales en los tramos que generan mayores pérdidas de carga. Las soluciones posibles se enfocan en reducir las pérdidas de carga, elevar la presión general en la red por medio de tanques más altos, modificar una parte de la red para mejorar su funcionamiento hidráulico e incrementar la capacidad de la red de distribución (CNA, 2007).

El mantenimiento de una red existente, se considera cuando se requiere cambiar algunos elementos hidráulicos, como tuberías, válvulas, bombas u otro elemento que estén en mal estado.

Existen diferentes tipos de topologías de las redes: ramificadas, malladas y mixtas (Bhave, 1991). Una red ramificada consiste en tener un solo punto de alimentación. Es decir, el agua sólo tiene un camino para llegar de un punto a otro. Por lo tanto se pueden calcular los caudales circulantes para las tuberías solo con la ecuación de continuidad (Iglesias et al., 2003).

Las redes malladas, se caracterizan por la existencia de mallas (ciclos) y cualquier par de puntos de la red puede ser unido por al menos dos caminos distintos. Por lo que, los caudales circulantes por las conducciones no solo quedan definidos aplicando únicamente la ecuación de continuidad, sino que también es necesario utilizar la ecuación de equilibrio de malla.

Las redes mixtas, es una combinación de ramificaciones y mallas por lo que se pueden calcular de forma inmediata los caudales circulantes mediante ambas ecuaciones de continuidad y equilibrio. Estas ecuaciones se presentan en el capítulo 3, sección 3.5.

De acuerdo a la teoría de la complejidad, el problema se clasifica como un problema NP-Completo (Gupta et al., 1993), debido a que es un problema complejo, ya que el esfuerzo computacional crece de forma exponencial conforme se incrementa el tamaño de la instancia, por lo que es necesario utilizar métodos de optimización de acuerdo a la clasificación de la complejidad de los problemas (Papadimitriou y Steiglitz, 1998) como metaheurísticas. Las metaheurísticas se definen como una estrategia de alto nivel aplicada a problemas combinatorios que mejore óptimos locales guiando el proceso de búsqueda, lo que permite encontrar buenas soluciones. Estos métodos aproximados están diseñados para problemas de optimización combinatoria en tiempo polinomial. Las metaheurísticas proporcionan un marco general para crear nuevos algoritmos híbridos, combinando diferentes conceptos derivados de la inteligencia artificial, la evolución biológica y los mecanismos estadísticos (Osman y Kelly, 1996).

Desde hace más de tres décadas, se han implementado métodos y modelos de optimización para encontrar el diseño óptimo de una red de distribución de agua, permitiendo reducir el consumo energético, mejorar la economía y los beneficios sociales (Alperovits y Shamir, 1977). El problema de diseño de redes de distribución de agua ha sido estudiado por varios investigadores (Shamir, 1974; Savic y Walters, 1997; Montesinos et al., 1999; Reca et al., 2008; Van Dijk et al., 2008;Liu et al., 2008; Hui Zhang et al., 2009; Shu y Zhang, 2010; Weickgenannt et al., 2010; Vasan y Simonovic, 2010; Lei et al., 2011; Artina et al., 2012; Chang et al., 2013; Kurek y Ostfeld, 2013; Ostfeld et al., 2014; D'Ambrosio et al., 2015), quienes han planteado diferentes métodos de optimización de forma teórica. Otros investigadores han implementado modelos reales para mejorar redes existentes, donde el objetivo es encontrar la mejor localización de válvulas de control para obtener las presiones adecuadas, evitando pérdidas de fugas de agua en sistemas de distribución de agua (Reis et al., 1997; Araujo et al., 2006; Cattafi et al., 2011; Dai y Li, 2014; Ali, 2015; Pecci et al., 2015).

Este trabajo de investigación trata de resolver una red de distribución de agua existente en el Fraccionamiento Real Montecasino, de Huitzilac, Morelos, mediante un algoritmo genético para mejorar su distribución en todos los puntos de consumo de la red.

1.2 Objetivo General

Desarrollo de un algoritmo genético que permita una distribución de agua eficiente de una comunidad, minimizando los costos por los cambios realizados en los elementos de la red.

1.3 Objetivos Específicos

1. Plantear un modelo de optimización para representar la instancia propuesta del problema de redes de distribución de agua.
2. Solucionar el modelo de la instancia propuesta mediante la elaboración de un algoritmo genético distribuido en clúster.
3. Realizar estudios experimentales para probar que el método de solución propuesto es eficaz para la red de distribución de agua del Fraccionamiento Real Montecasino.
4. Evaluar la eficiencia del algoritmo genético distribuido en clúster.

1.4 Alcance de la Investigación

1. El modelo de optimización será evaluado mediante un algoritmo genético y el modelo de satisfacción de restricciones con el simulador Epanet.

2. Las pruebas serán ejecutadas en clúster para evaluar la eficiencia del algoritmo.
3. Sintonización de los parámetros de control del algoritmo genético secuencial, para mejorar el tiempo de convergencia y la eficacia de las soluciones de dicho algoritmo.
4. La instancia de prueba utilizada para el algoritmo genético, es la red de distribución de agua que existe en el Fraccionamiento Real Montecasino de Huitzilac, Morelos.

1.5 Contribuciones de la Tesis

Las contribuciones principales de este trabajo de investigación son las siguientes:

1. Modelo de optimización para poderlo utilizar en la solución de la red de distribución de agua del Fraccionamiento Real Montecasino del municipio de Huitzilac.
2. Diseño y desarrollo de un algoritmo genético distribuido para el problema de la red de distribución de agua del Fraccionamiento Real Montecasino, Huitzilac.
3. Modificaciones al código libre de Epanet con la finalidad de configurar las características adecuadas de nuevos tanques de almacenamiento para poderlo utilizar en la solución de la red del Fraccionamiento Real Montecasino, Huitzilac.
4. Una solución a la red del Fraccionamiento Real Montecasino, la cual permite distribuir agua a todo el fraccionamiento, lo que actualmente no sucede.

1.6 Organización de la Tesis

Capítulo 1. Introducción. En este capítulo se da una introducción general del problema de distribución de agua. Se presenta el objetivo general, así como los objetivos específicos, alcances, contribuciones y por último la organización general de la tesis. Capítulo 2. Marco teórico. En este capítulo se da a conocer los métodos exactos, heurísticos, metaheurísticos y paralelos implementados por otros autores para resolver el problema de redes de distribución de agua en general. Capítulo 3. Problema de distribución de agua de la red del FRM. En este capítulo se expone la descripción de la red hidráulica del Fraccionamiento Real Montecasino de la comunidad de Huitzilac, la representación simbólica del problema, se define la problemática de la red, por último se da una propuesta del modelo de optimización, así como se presenta el modelo de satisfacción de restricciones. Capítulo 4. Metodología de solución. En este capítulo se describe el algoritmo genético secuencial propuesto y sus componentes, así como el algoritmo genético distribuido. Capítulo 5. Resultados experimentales. En este capítulo se presentan los resultados del algoritmo genético secuencial y algoritmo genético distribuido. La metodología de sintonización de los parámetros de control del algoritmo secuencial, el análisis de la complejidad del algoritmo. Un análisis de eficacia y eficiencia de ambos algoritmos. Capítulo 6. Conclusiones y Trabajo futuro. En este capítulo se presentan las conclusiones obtenidas de este trabajo de investigación, así como el trabajo futuro.

Capítulo 2

Marco Teórico

Capítulo 2 Marco Teórico

2.1 Introducción

La optimización combinatoria es una rama de la optimización en matemáticas aplicadas y en ciencias computacionales, relacionada con la investigación de operaciones, teoría de algoritmos y teoría de la complejidad computacional (Cook et al., 1997). La optimización combinatoria consiste en la solución algorítmica de problemas donde se busca maximizar o minimizar el valor de la función objetivo. Los algoritmos de optimización combinatoria resuelven instancias de problemas que son difíciles en general, explorando el espacio de soluciones. Algunos tipos de problemas de optimización pueden ser resueltos en tiempo polinomial mediante métodos exactos, como el problema de árbol de expansión mínima, problemas de la ruta más corta y la red de flujo máximo entre otros.

Los métodos de optimización se clasifican en métodos exactos y métodos aproximados (o heurísticos). Los métodos exactos son aquellos que proporcionan una solución óptima de cualquier problema, estos métodos no son muy utilizados para encontrar soluciones a problemas NP-duros, debido a que el tiempo crece exponencialmente con el tamaño del problema.

Los métodos heurísticos encuentran resultados óptimos en tiempo razonables. Una *heurística* es un método bien estructurado, desarrollado con base a la experiencia que permite obtener soluciones aproximadas de un problema sin garantizar la optimalidad (Wetzel, 1983). Dentro de los algoritmos heurísticos se encuentran las metaheurísticas. Las metaheurísticas son algoritmos de propósito general que pueden ser aplicadas para resolver cualquier tipo de problema de optimización. Los métodos metaheurísticos se pueden definir como metodologías generales de

nivel superior que se pueden utilizar como estrategias de guía en el diseño de la heurística para resolver problemas de optimización (Talbi, 2009).

2.2 Métodos aplicados al problema de distribución de agua

Desde hace más de tres décadas se han implementado métodos exactos y de aproximación de forma teórica y real, para encontrar el diseño óptimo de una red de distribución de agua permitiendo obtener el mínimo costo del diseño de la red. Otros investigadores han implementado modelos de optimización para mejorar redes existentes, donde el objetivo es encontrar la mejor localización de válvulas de control para obtener las presiones adecuadas, evitando pérdidas de fugas de agua en sistemas de distribución de agua.

2.2.1 Métodos Exactos

Los métodos exactos son aquellos que proporcionan una solución óptima de cualquier problema, estos métodos no son muy utilizados para encontrar soluciones a problemas NP-duros, debido a que el tiempo crece exponencialmente con el tamaño del problema (Talbi, 2009). Algunos de los métodos exactos más utilizados son: programación lineal entera mixta programación dinámica y algoritmos de Branch & Bound (B&B)

Programación lineal entera mixta (MILP) es un problema de programación lineal en el que algunas de las variables son enteras y otras variables binarias. En la ingeniería los problemas más frecuentes son los problemas de programación lineal entera mixta, estos problemas proporcionan un marco de modelado flexible y eficiente para formular y resolver muchos problemas de ingeniería (Juárez, 2013).

Programación dinámica se basa en la división recursiva de un problema en subproblemas más simples y se va construyendo la solución con las soluciones de esos subproblemas. Este método de optimización por etapas es el resultado de una secuencia de decisiones parciales. El procedimiento evita una enumeración total del espacio de búsqueda mediante la poda de secuencias de decisiones parciales que no puede conducir a la solución óptima (Talbi, 2009).

Algoritmo de Branch & Bound es posiblemente el más empleado en la resolución exacta de problemas de optimización combinatoria. El método se basa en una enumeración implícita de todas las soluciones del problema de optimización a tratar. El espacio de búsqueda es explorado dinámicamente por la estructura de un árbol. El nodo raíz representa el problema a resolver. Los nodos hoja son las posibles soluciones y los nodos internos son subproblemas del espacio total de la solución. La poda de árbol de búsqueda se basa en una función de delimitación que poda subárboles que no contienen ninguna solución óptima (Juárez, 2013).

Programación con restricciones es un lenguaje construido en torno a los conceptos de búsqueda de árbol y las consecuencias lógicas. Los problemas de optimización de programación con restricciones son modelados por medio de un conjunto de variables ligadas por un conjunto de restricciones (Talbi, 2009). Las variables toman sus valores en un dominio finito de números enteros. Las restricciones pueden tener formas matemáticas o simbólicas.

Algunos de los métodos exactos que se han utilizado para resolver el problema de redes de distribución de agua en general se presentan a continuación:

En el trabajo de (Kessler y Shamir, 1989) presentan un análisis teórico de la programación lineal y el método del gradiente para el diseño óptimo de redes de distribución de agua. El método propuesto por Alperovitz y Shamir (1977) consiste en dos etapas: (1) un problema de programación lineal se

resuelve para una distribución y el flujo factible dado (2) una búsqueda se lleva a cabo en el espacio de variables de flujo, basado en el gradiente de la función objetivo. Presentan la formulación de matriz para ambas etapas utilizando matrices de la teoría de grafos. El procedimiento de búsqueda se mejora usando el método del gradiente.

Sherali y Smith propusieron un algoritmo de Branch and Bound (B&B) para resolver el problema de diseño de la red de distribución de agua. Abordaron un enfoque de optimización global (Sherali y Smith, 1997). El algoritmo fue aplicado a las dos variantes del problema clásico de Alperovitz y Shamir (1977), obteniendo en ambos casos buenos resultados.

En el trabajo de (Cattafi et al., 2011) presentaron una nueva aplicación de la programación lógica a un problema de la vida real en ingeniería hidráulica. Tomaron la formulación del problema de (Giustolisi et al., 2008). Implementaron un algoritmo de programación lineal para la localización óptima de válvulas en una red de distribución de agua. Obtuvieron mejores soluciones que la mejor solución conocida en la literatura para la red de distribución de agua de Apulia.

En el trabajo de (Dai y Li, 2014) presentaron un enfoque de reformulación para resolver el problema MILP (Programación Lineal Entera Mixta) para identificar las ubicaciones óptimas de válvulas reguladoras de presión en un sistema de distribución de agua. El problema MILP se reformula como MPCC (Programación Matemática con restricciones complementarias) que puede resolverse de manera eficiente por los algoritmos de NLP (Programación No Lineal). El problema MPCC se regulariza a un NLP utilizando un método de penalización que se resuelve por una secuencia de problemas de NLP.

2.2.2 Métodos Heurísticos

Los métodos heurísticos encuentran resultados óptimos en tiempo razonables. Una *heurística* es un método bien estructurado, desarrollado con base a la experiencia que permite obtener soluciones aproximadas de un problema sin garantizar la optimalidad (Wetzel, 1983). Dentro de los heurísticos se encuentran las *metaheurísticas* que son algoritmos de propósito general que pueden ser aplicadas para resolver cualquier problema de optimización. Estos algoritmos pueden ser vistos como metodologías generales de nivel superior que se pueden utilizar como una estrategia de guía en el diseño de heurísticas para resolver problemas de optimización específicos (Talbi, 2009).

Las metaheurísticas se dividen en dos clases: *basadas en trayectoria*, las cuales requieren una solución inicial simple y en cada paso de la búsqueda la solución actual es reemplazada por una mejor solución encontrada en su vecindad. Estos métodos permiten encontrar rápidamente una solución local óptima. Los métodos basados en una solución son: recocido simulado (SA), búsqueda tabú (TS), búsqueda local iterada (ILS), búsqueda local variable (VNS). Las metaheurísticas *basadas en población*, son aquellas que hacen uso de una población de soluciones, la cual mediante un proceso iterativo se va mejorando. En cada generación del proceso, la población se sustituye por nuevos individuos que fueron creados mediante el uso de ciertos operadores. Estos métodos son: algoritmos evolutivos (EAs), optimización de colonia de hormigas (ACO), optimización por enjambre de partículas (PSO), búsqueda dispersa (SS), evolución diferencial (DE), estrategias evolutivas (ES) (Gendreau & Potvin, 2005;Talbi, 2009;Alba et al., 2013).

Recocido simulado es una metaheurística de búsqueda aleatoria utilizada en la solución de problemas de optimización combinatoria, la cual fue propuesta por (Kirkpatrick et al., 1983) para el diseño de circuitos VLSI, y

(Cerny, 1985) para el problema del agente viajero (TSP), ellos mostraron que este proceso podría tener una asociación de los conceptos del proceso original de simulación, con elementos de problemas de optimización combinatoria y por lo mismo podría ser aplicado a estos problemas. Este algoritmo se basa en la analogía entre el proceso de recocido de sólidos y los problemas de optimización combinatoria.

Búsqueda tabú fue propuesta por Glover (1989,1990). La TS es básicamente una estrategia de búsqueda local determinista, donde en cada iteración, la mejor solución en la vecindad de la solución actual es seleccionada como la nueva solución actual, a pesar de que se obtenga un aumento del costo de la solución, esto para evitar quedar atrapado en un óptimo local. Utiliza una memoria a corto plazo, conocida como la lista tabú, donde se guardan las soluciones recientes (o atributos de soluciones visitadas recientemente) para evitar ciclismo a corto plazo (Gendreau, 2003; Glover et al., 2007).

Búsqueda local iterada es una heurística que aplica de forma iterativa un método de búsqueda local (Lourenço et al., 2001; Den et al., 2001). Hoos y Stützle afirman que ILS es una de las metodologías más sencillas y eficaces para evitar quedar atrapados en óptimos locales (Hoos y Stützle, 2004). El proceso inicia con una solución cualquiera s y el conjunto de soluciones en el vecindario $N(s)$, del cual se elige una solución s' a través de un movimiento σ. Este movimiento σ se realiza a través de un proceso estocástico, el cual mejora la función objetivo, es decir, si se requiere minimizar, entonces se debe de cumplir que $f(s') <= f(s)$, si esto se cumple se reemplaza la solución s por la solución s' que la mejore, esto se repite hasta alcanzar el criterio de paro de la búsqueda local. Se realizan nuevas búsquedas locales de manera iterativa. Cada vez que termina una búsqueda local se evalúa $f(s') <= f(S)$, si esto se cumple se reemplaza la solucion local s' por la mejor solución S que se tiene hasta el momento. Este procedimiento continúa hasta que la solución no siga mejorando. El criterio

de paro de ILS es un máximo número de ejecuciones de búsquedas locales (Martínez-Bahena et al., 2012).

Búsqueda local variable se basa en la idea de cambiar sistemáticamente la estructura de vecindad dentro de un heurístico de búsqueda local, en lugar de utilizar una sola estructura de vecindad (Mladenović y Hansen, 1997). Dado un conjunto de estructuras de vecindad, una solución se genera aleatoriamente en la primera zona de la solución actual, a partir del cual se realiza un descenso local. Si el óptimo local obtenido no es mejor que el titular, entonces el procedimiento se repite con la siguiente zona Hansen y Mladenović (2001, 2003, 2005).

Búsqueda dispersa fue originalmente propuesta por (Glover, 1996). La SS es una técnica evolutiva, combina un conjunto de soluciones para crear nuevas soluciones. En cada iteración se dispone de un conjunto de soluciones del problema y a partir de ellas se realizan operaciones para obtener un nuevo conjunto de soluciones que sean de alta calidad y que al mismo tiempo sean suficientemente diferentes para lograr una búsqueda que abarque.

Algoritmos evolutivos representan una amplia clase de metodologías de resolución de problemas, con algoritmos genéticos es uno de los más conocidos (Holland, 1975). Estos algoritmos están basados por la forma en que las especies evolucionan y se adaptan a su medio ambiente para la supervivencia, basado en el principio de la selección natural por (Darwin, 1859).

Desde hace más de tres décadas se han implementado diferentes métodos de optimización para resolver el problema de redes de distribución de agua en general.

En el trabajo de (Araujo et al., 2006) implementaron un algoritmo genético y un módulo de Epanet para el análisis hidráulico de la red de Hanoi. Desarrollaron dos modelos basados en el método de optimización

con un algoritmo genético para el control de la presión y la reducción constante de fuga.

En la investigación de (Reca et al., 2008) evaluaron el desempeño de varias metaheurísticas como: algoritmo genético, recocido simulado, búsqueda tabú y búsqueda local iterada, para resolver el problema de diseño de redes de distribución de agua, considerando obtener el costo mínimo y la mejor configuración de diámetros de las tuberías. Las pruebas las realizaron para una red mediana (*Alperovits*) y una red grande (*Balerma*).

En el trabajo de (Baños et al., 2010) aplicaron un algoritmo memético para el diseño de redes de distribución de agua, considerando obtener el menor costo y la mejor configuración de los diámetros de las tuberías. Realizaron pruebas para tres redes muy conocidas en la literatura (*Alperovits, Hanoi y Balerma*).

Vasan y Simonovic desarrollaron un modelo de computación llamado DENET, el cual involucra la técnica de optimización de evolución diferencial y el simulador hidráulico Epanet (Vasan y Simonovic, 2010). Formularon un modelo para minimizar el costo del diseño del sistema de abastecimiento de agua de Nueva York y la red de Hanoi. La aplicación DENET demostró buenos resultados para ambos problemas.

Cisty propuso una hibridación del algoritmo genético con el método de programación lineal (LP), llamado GALP para resolver el problema del diseño de sistemas de distribución de agua (Cisty, 2010). El método está elaborado para su uso en el diseño y rehabilitación de redes de agua potable y en sistemas de riego a presión. Las pruebas se realizaron a la red de Hanoi, pero además se realizaron para la red de Hanoi doble y triple.

En el trabajo de (Lei et al., 2011) propusieron un método de optimización de colonia de hormigas para resolver el problema de la red de distribución de agua. Las pruebas se realizaron a la red Hanoi. Obtuvieron el menor costo de 2.2 millones para la red de distribución de agua.

Geem desarrollo un modelo de optimización multiobjetivo para las redes de distribución de agua, usando la teoría difusa *(fuzzy)* y una búsqueda armónica (Geem, 2015). Las pruebas se realizaron a dos instancias, la red de Hanoi y la una red real. Los resultados del modelo fueron exitosos, permitiendo minimizar el costo y maximizar la fiabilidad.

Otros autores también han implementado métodos de optimización para el problema de encontrar la ubicación óptima de válvulas, con la finalidad de obtener un control adecuado de la presión en las redes de distribución de agua.

En el trabajo de Reis et al. (1997) abordaron el problema de encontrar la ubicación óptima de las válvulas de control y sus valores adecuados en una red de tuberías de suministro de agua. Para ello utilizaron un algoritmo genético para obtener la máxima reducción de las fugas de un determinado nodal de demandas y los niveles de los embalses.

En el trabajo de (Nicolini y Zovatto, 2009) presentaron el problema de la gestión de la presión óptima en sistemas de distribución de agua a través de válvulas reductoras de presión (PRVs) y así controlar las fugas de agua. Formularon un problema de optimización de dos criterios. El primer criterio es minimizar el número de válvulas y el segundo es minimizar la fuga total en el sistema. Dicha modelo lo resolvieron mediante un algoritmo genético multiobjetivo.

Liberatore y Sechi utilizaron un método de referencia de presión (PRM) para seleccionar el número y la ubicación de las válvulas. Aplicaron una búsqueda de dispersión para encontrar la ubicación óptima y calibración de las válvulas en las redes de distribución de agua (Liberatore y Sechi, 2009). Formularon una función multiobjetivo, considerando obtener el menor costo de la inserción de las válvulas y una penalización cuando la presión supera el valor máximo permitido. Las pruebas fueron realizadas a una red teórica y a una red real de la cuidad de Burcei en Cerdeña, Italia.

En la tesis de Hurtado-Guzmán, (2009) se presentó un algoritmo genético multiobjetivo para encontrar la localización óptima de las válvulas reductoras de presión. Utilizo la codificación de un cromosoma mixto (binario y real). El método fue aplicado a algunos sectores hidráulicos del poniente del Distrito Federal, México.

Kang y Lansey utilizaron un algoritmo genético para minimizar la masa de inyección de cloro en fuentes o para reducir las concentraciones de cloro y las presiones mínimas en todo el sistema (Kang y Lansey, 2010). Aplicaron el método a un sistema de tamaño mediano, el cual muestro que el funcionamiento óptimo de las válvulas existentes combinados con la desinfección puede mejorar la calidad del agua.

En el trabajo de (Saldarriaga y Salcedo, 2015) desarrollaron un modelo de optimización multiobjetivo para la determinación de la localización óptima y la configuración de válvulas reductoras de presión en las redes de distribución de agua, con la finalidad de minimizar las pérdidas de agua. Utilizaron el algoritmo NSGA-II y realizaron las pruebas la red de la subsección 8-04 de Bogotá, Colombia.

Ali presentó un modelo de optimización para minimizar las fugas en los sistemas de distribución de agua. El modelo fue resuelto mediante un algoritmo genético, incorporando mejoras para reducir el espacio de búsqueda y mejorar la eficiencia en la búsqueda de las soluciones óptimas. El modelo también permite encontrar la ubicación óptima de las válvulas y sus ajustes adecuados (Ali, 2015).

En el trabajo de (Wright et al., 2015) propusieron un método de optimización para el control de las válvulas reductoras de presión en las redes de distribución de agua con topología dinámica. Utilizaron el simulador Epanet para realizar el análisis. El uso de un estudio experimental en una red operativa grande, mostró la optimización de la configuración de PRV con una topología dinámico para dar lugar a las reducciones de la presión de 3,7% en comparación con PRV optimizadas en una estructura cerrada.

En el trabajo de (Darvini & Soldini, 2015) se presentan el análisis de un sistema de distribución de agua de la ciudad de Chiaravalle, Italia. El modelo hidráulico de la red ha sido implementado con Epanet y calibrado para mejorar la gestión del sistema. Las simulaciones numéricas les permitieron analizar el comportamiento de la red bajo diferentes condiciones de operación y verificar la posibilidad de reducir la presión de servicio mediante la inserción de algunas PRV.

Algunos trabajos que se desarrollaron bajo en ambiente Grid para mejor la eficiente del algoritmo.

En el trabajo de (Ewald et al., 2008) presenta un algoritmo genético multiobjetivo distribuido en implementación Grid. La versión del algoritmo distribuido se basa en el modelo de islas. El algoritmo se aplicó a la asignación de las estaciones de bombeo en un sistema de distribución de agua potable. El algoritmo basado en Grid se aplicó para un caso de estudio. Los resultados se compararon con una versión no distribuida del algoritmo.

En la tesis de (Ávila-Melgar, 2015) se presenta un algoritmo genético en ambiente Grid para el problema de diseño de redes de distribución de agua. Propuso un modelo Maestro-Alumno para la distribución de dicho algoritmo. Las pruebas se realizaron para la red de *Alperovits, Hanoi* y *Balerma*, para las tres instancias se alcanzó la cota del costo de acuerdo al conocido en la literatura.

Capítulo 3

Problema de Distribución de Agua de la Red del FRM

Capítulo 3 Problema de Distribución de Agua de la Red del FRM

En este capítulo se presenta la descripción de la red hidráulica del Fraccionamiento Real Montecasino del municipio de Huitzilac. Muestra la representación simbólica del problema mediante la teoría de grafos. Con la cual se define la problemática que se tiene en la red. Se detalla el modelo de optimización propuesto para darle solución al problema. También se describe el modelo de satisfacción de restricciones utilizado para cumplir con las leyes hidráulicas de un sistema de agua.

3.1 Descripción de la red del FRM

La red de distribución de agua del Fraccionamiento Real Montecasino (FRM) está ubicada en el municipio de Huitzilac, Morelos, México. La red tiene una población de 366 habitantes y está a 2250 metros de altitud sobre el nivel del mar.

La Figura 3-1 muestra la red hidráulica del FRM. La red está compuesta por cinco secciones (Ensueño, Piamonte, Montecasino, Sección 1 y 2), las secciones están indicadas con un color, por ejemplo la sección de Piamonte esta de color rojo, la sección 1 y 2 de color verde y la sección Ensueño y Montecasino de color negro. Las líneas de color azul, corresponde a las tuberías. Los puntos de color blanco representan cada una de la toma domiciliaria (casas, industrias, comercios, etc.). Los cuadros de color azul, indican los tanques de almacenamiento de cada sección.

Los tanques generales (pozo, lago o rio), es de donde se abastecen los tanques de almacenamiento. Esta red hidráulica está instalada con diferentes niveles de altitud, la parte más alta es la de sección 2 y sección Ensueño, las secciones más bajas son Piamonte y sección 1. Actualmente la red del FRM no cuenta con bombas debido a que el suministro de agua es por gravedad. Tampoco se tienen válvulas para el control adecuado de la presión en la red del FRM. La red a tratar en este trabajo es considerada como una red mixta debido a que contiene mallas y ramificaciones, por lo que es necesario utilizar la ecuación de continuidad y energía (ver sección 3.5). La Figura 3-2 muestra las calles la red hidráulica del Fraccionamiento Real Montecasino, Huitzilac.

Figura 3-1 Red hidráulica del Fraccionamiento Real Montecasino, Huitzilac

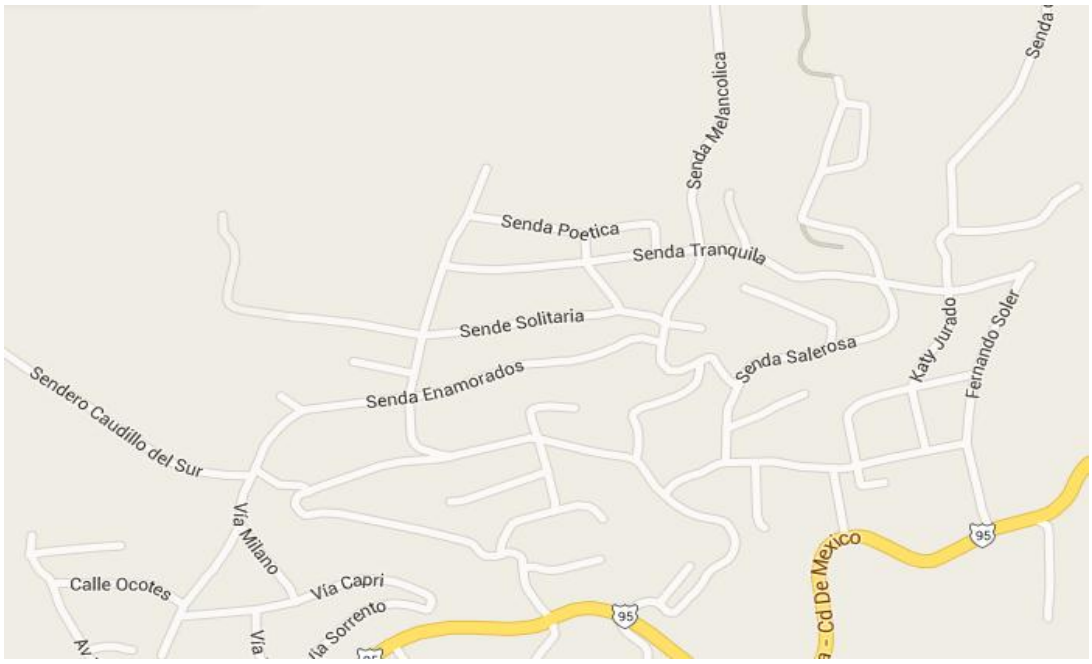

Figura 3-2 Calles que se encuentran en el Fraccionamiento Real Montecasino

De la Figura 3-3 a la Figura 3-8 se muestran las fotos de los tanques de la red hidráulica del Fraccionamiento Real Montecasino, cada uno tiene diferente capacidad de almacenamiento. El tanque "Sección 1" tiene un volumen de 108.00 m^3. El tanque "Ensueño" tiene una capacidad de 237.79 m^3. El tanque de agua "1" tiene un volumen de 110.60 m^3. El tanque de agua "2" con un volumen de 222.20 m^3. El tanque "Sección 1 y 2" tiene una capacidad de 324.00 m^3 y el tanque "Montecasino" tiene un volumen de 70.00 m^3. Estos datos fueron tomados de los registros del sistema de agua potable del municipio de Huitzilac.

Figura 3-3 Tanque “Sección 1”

Figura 3-4 Tanque “Ensueño”

Figura 3-5 Tanque de agua “1”

Figura 3-6 Tanque de agua “2”

Figura 3-7 Tanque “Sección 1 y 2”

Figura 3-8 Tanque “Montecasino”

3.2 Representación simbólica del problema

El problema de distribución de agua se puede representar mediante la teoría de grafos. En general, un grafo es un conjunto de puntos y líneas donde cada línea une a un par de puntos. Entonces un grafo G se denota como $G = (V, A)$, donde V es un conjunto de vértices o nodos y A es un conjunto de aristas o arcos. Existen diferentes tipos de grafos, los grafos no dirigidos son aquellos que tienen la dirección en ambos sentidos y los grafos dirigidos solo tienen una dirección (Korte y Vygen, 2008).

La Figura 3-9 muestra el grafo de la red del FRM, donde los nodos representan a las fuentes de abastecimiento (embalses, tanques) o puntos de consumo (viviendas, comercios, industrias, etc.), si en un nodo se encuentra una fuente de abastecimiento o un punto de consumo este se representa mediante un símbolo diferente (Tabla 3-1). Las aristas representan a los elementos de conexión como son tuberías, válvulas, bombas, entre otros. Cada arista tiene un costo asociado que puede representar el costo del diámetro de la tubería, la velocidad del flujo en las tuberías, la longitud de una tubería, el costo de la válvula, el costo de la bomba u otro parámetro.

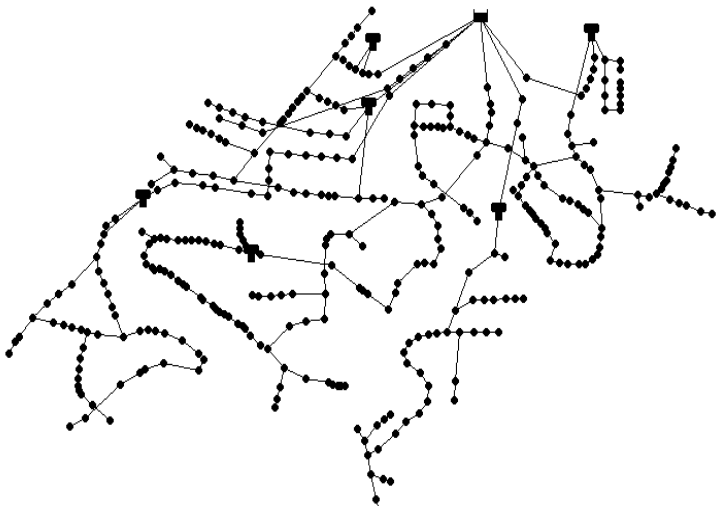

Figura 3-9 Representación de la red hidráulica en un grafo

La Tabla 3-1 muestra la representación simbólica de los elementos en una red. Algunos símbolos se presentan en la red original del FRM (Figura 3-9), como son tanques, embalses, tomas domiciliarias y tuberías. Los elementos de bombas y válvulas no se presentan porque actualmente no existen en la red del FRM.

Tabla 3-1 Representación de los símbolos

Símbolo	Representación
	Tanque de almacenamiento
	Embalse (rio, lago, pozo, etc.)
	Toma domiciliaria (vivienda, comercios, industrias, etc.)
	Válvula
	Bomba
	Tubería

La Tabla 3-2 resume los elementos principales de la red del FRM. Existen 364 tuberías con un diámetro de 22.7 a 101.6 mm, en el Apéndice A se presenta la descripción de cada una de las tuberías con sus características. Para mostrar los valores de los nodos se utiliza una demanda estándar de 300 litros por día en cada nodo. La presión mínima requerida es de 10 mca y presión máxima requerida es de 60 mca.

Tabla 3-2 Resumen de la red del FRM

Tuberías	Nodos	Tanques	Embalses	Demanda Estándar (L/S)	Presión mínima (mca[1])	Presión máxima (mca)	Diámetros tuberías (mm)
364	350	6	1	0.003	10	60	22.7 -101.6

La Tabla 3-3 y Tabla 3-4 muestran los parámetros para las tuberías y depósitos de la red hidráulica del FRM. Para las tuberías se necesita la longitud y diámetro. En el caso de los tanques es necesario conocer la altura, la cual es la cota topográfica de la unión del depósito con el resto de la red (metros sobre el nivel del mar). El nivel inicial, es el agua en el depósito antes de iniciar la simulación. El nivel mínimo a partir del cual se cierran los consumos, el nivel máximo a partir del cual se cierran los aportes y diámetro de los tanques. Estos datos son los necesarios para definir la problemática de la red del FRM.

Tabla 3-3 Datos de las tuberías

Tubería	Longitud(m)	Diámetro(mm)
1	238	38.1
2	991	19.05
3	299	38.1
4	991	19.05
..	..	..
364	572.1	50.8

[1]mca: metros columna de agua

Tabla 3-4 Características de los depósitos

Tanque	Altura (m)	Nivel Inicial (m)	Nivel Mínimo (m)	Nivel Máximo (m)	Diámetro (m)
1	2308	2	1	4	8.67
2	2334	2	0	4	5.86
3	2337	2	0	4	4.72
4	2364	2	0	4	5.93
5	2373	2	1	4	8.41
6	2398	2	1	4	10.16

3.3 Definición del problema

Actualmente la red del FRM no hace una distribución satisfactoria, esto es probablemente debido al incremento de la demanda en base a nuevos asentamientos de la población, por lo que no satisface las necesidades de todos los usuarios. Es importante mencionar que para que un sistema de distribución de agua opere correctamente debe cumplir con un conjunto de condiciones. La condición principal es disponer de las presiones mínimas necesarias para garantizar a los consumidores un servicio adecuado, evitando fugas de agua por rupturas de tuberías debido al exceso de presión y evitando así posibles pérdidas económicas.

Para el análisis de la red del FRM se realizó una simulación hidráulica utilizando el simulador Epanet. Epanet es un software libre, permite el análisis del comportamiento hidráulico en los sistemas de distribución de agua potable. Fue desarrollado por la Agencia de Protección Ambiental de Estados Unidos (Rossman, 2000). Es un software que se puede utilizar tanto en ambiente Windows como en Linux. Ha sido utilizado por varios investigadores (Araujo et al., 2006; Nicolini y Zovatto, 2009; Cattafi et al.,

2011; Arunkumar y Nethaji, 2011; Ali, 2015; Dai y Li, 2014), quienes han trabajado con problemas de redes de distribución de agua. Epanet necesita los datos adecuados para la simulación. Un ejemplo de los datos se muestra de la Tabla 3-2 a la Tabla 3-4. Epanet calcula las pérdidas de carga por fricción en las tuberías mediante las fórmulas introducidas por Darcy-Weisbach, Manning y Hazen-Williams (Rossman, 2000). Epanet permite simular sistemas de cualquier tamaño. Para la simulación de la red del FRM se utiliza la fórmula de Hazen-Williams:

$$10{,}674\mathrm{C}^{-1{,}852}\ \mathrm{d}^{-4{,}871}\mathrm{L} \qquad (3\text{-}1)$$

Donde C es el coeficiente de rugosidad, d es el diámetro (mm) de la tubería y L es la longitud (m) de la tubería. En este trabajo el valor del coeficiente de rugosidad es de 100, debido a que el material de la tubería es hierro galvanizado. La fórmula de Hazen-Williams es la más utilizada para flujo de agua. Epanet arroja el caudal que fluye por cada tubería, la presión en cada nodo, el nivel de agua en los tanques durante un determinado periodo de tiempo.

La Figura 3-10 muestra la representación de la red después de la simulación en ambiente Windows. Epanet arroja una escala de las presiones representada por colores. Los puntos señalados mediante un círculo de color azul indican que la presión en los nodos es negativa. La presencia de las presiones negativas es debida a la baja altura de las fuentes (embalses o depósitos), por pérdidas elevadas en alguna conducción o deficiencia en el suministro de la demanda requerida. Los nodos de color rojo indican que la presión es mayor a 60 mca y los demás colores representan las presiones que están en el rango.

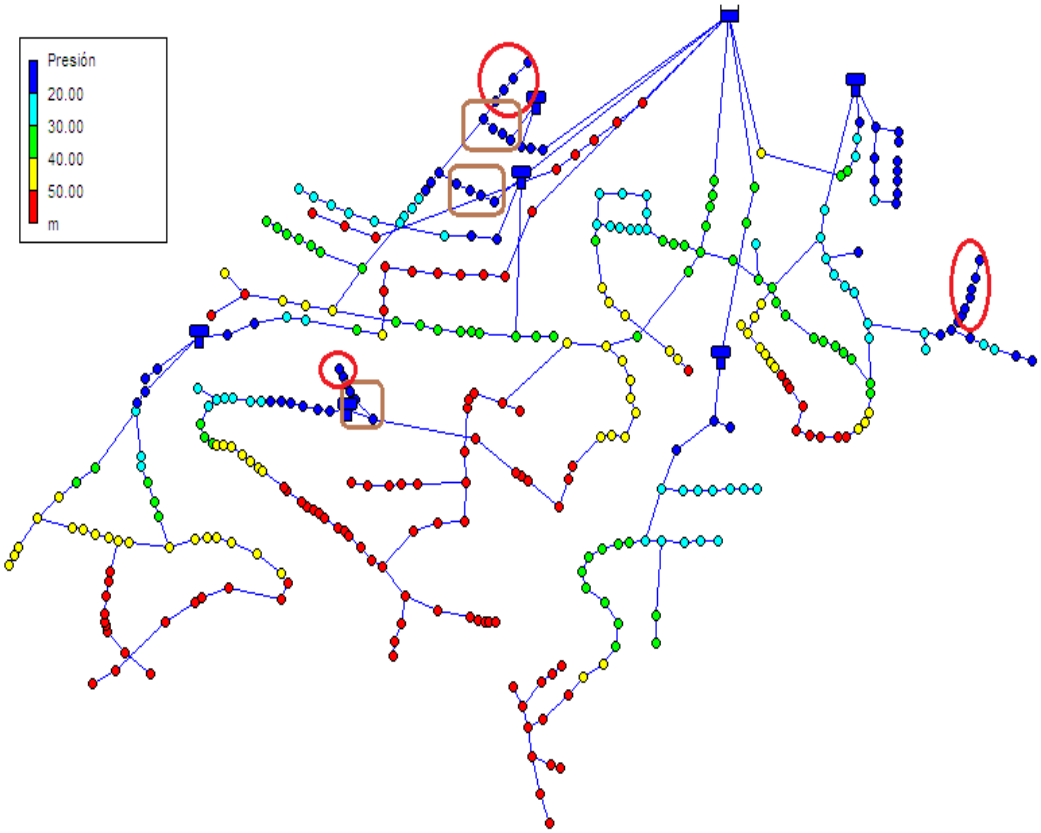

Figura 3-10 Simulación de la red del FRM

La Figura 3-11 presenta las presiones obtenidas en cada nodo. Los resultados muestran que hay varios nodos con presiones que no cumplen con la presión mínima y máxima establecida. Los nodos del rango 217 hasta 241 tienen presión negativa, en un total de 24 nodos. La carga de presión negativa es de aproximadamente -10 a -25 mca; esto indica que el agua no llega en esos nodos. Por otro lado, el exceso de presión podría ocasionar problemas de rupturas de tuberías, por lo tanto conlleva a fugas de agua y pérdidas económicas (Martínez-Bahena et al., 2014).

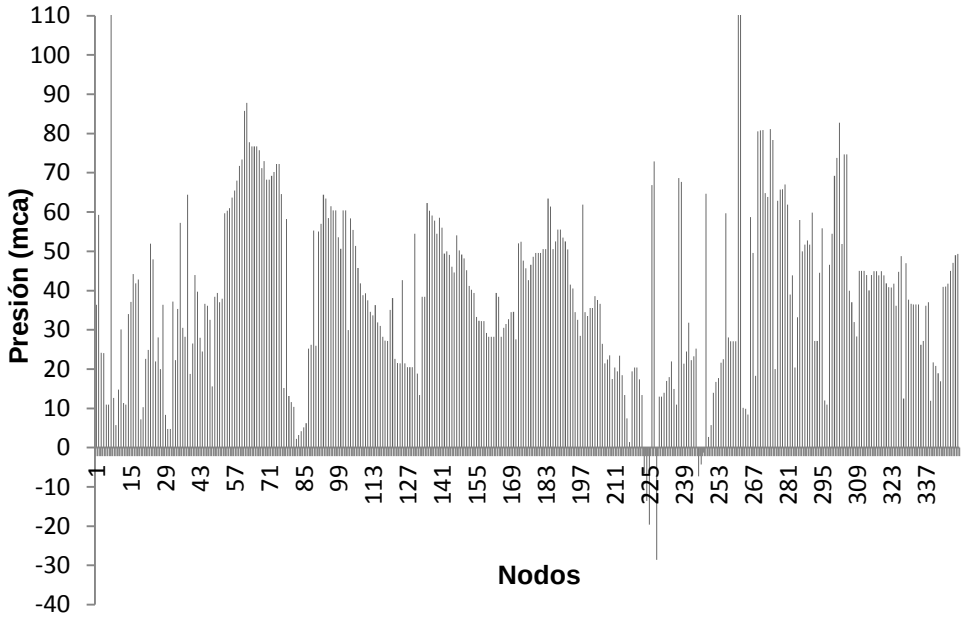

Figura 3-11 Presiones en los nodos de la red del FRM

De acuerdo al análisis realizado a la red del FRM, se observa que debido al mal diseño y ubicación topográfica de la zona, ya que cuenta con muchos desniveles por lo que en algunas tomas de suministro las presiones son demasiadas bajas e incluso negativas, por lo tanto el suministro de agua no es suficiente para muchos usuarios y en otros casos es muy elevada lo cual podría provocar fugas constantes, por lo que no se tendría un servicio adecuado para toda la comunidad.

Por lo tanto la problemática que se tiene que resolver es mejorar la distribución del agua, abasteciendo a todos los usuarios mediante un rediseño de la red hidráulica existente. Una forma es agregar nuevos tanques de almacenamiento, considerando también válvulas reguladoras de presión, tuberías o bombas. La opción de cambiar diámetros en las tuberías no es factible debido a lo caro que resultaría reconstruir la red hidráulica existente, por lo que realizar ese cambio implicaría cambiar las tuberías que están enterradas, lo cual requiere hacer excavaciones en toda la red e implicaría costo económico y tiempo.

Para agregar los elementos inicialmente enunciados a la red hidráulica, se debe obtener una configuración óptima de tal manera que genere el menor costo posible. De acuerdo a la representación del problema y análisis de la red, se propone un modelo de optimización, con el objetivo de encontrar el mínimo número de tanques, válvulas y tuberías necesarias a incorporar para que la red hidráulica pueda tener un funcionamiento eficaz, además de minimizar el costo en las modificaciones que fueron necesarias.

3.4 Modelo de optimización

El modelo de optimización propuesto en este trabajo se presenta a continuación. Está compuesto por una función objetivo y un conjunto de restricciones, las cuales permiten el adecuado funcionamiento en un sistema de distribución de agua. La función objetivo consiste en obtener el costo mínimo que implica agregar nuevos elementos al sistema, como son tanques de almacenamiento, válvulas reductoras de presión y tuberías. La función está sujeta a un conjunto de restricciones que permiten validar las soluciones.

$$\min \; C(P,T,V) = \sum_{T=1}^{nT}\sum_{i=1}^{nd}\sum_{j=1}^{p} C_i\, L_{ij} X_{ijT} + \sum_{T=1}^{nT} C_T\, Y_T + \sum_{V=1}^{nV}\sum_{J=1}^{nP} C_V\, K_{VJ} \tag{3-2}$$

Sujeto a:

$$\sum_{i=1}^{nd}\sum_{j=1}^{p} X_{ijT} = 1 \qquad \forall \; T \tag{3-3}$$

$$d_{min} \leq d_{ij} \leq d_{max} \tag{3-4}$$

$$X_{ijT} \in \{0,1\} \tag{3-5}$$

$$Y_T \in \{0, 1\} \quad (3\text{-}6)$$

$$K_{VJ} \in \{0, 1\} \quad (3\text{-}7)$$

La ecuación (3-2) representa la función objetivo. Esta consiste en obtener el costo mínimo de los cambios realizados en la red, el costo de las nuevas tuberías, de los nuevos tanques y válvulas. Para este trabajo se utilizaron costos de los componentes con base a un promedio de cotizaciones hechas en 2012 (véase Apéndice B).

La función objetivo está sujeta a un conjunto de restricciones. La restricción en (3-3) indica que diámetros de tubería se utilizan tomados de la lista de diámetros comerciales. Para cada segmento de tubería solo se utiliza un diámetro.

El conjunto de restricciones en (3-4) asegura que el diámetro elegido sea exclusivo de la lista de diámetros comerciales disponibles.

El conjunto de restricciones en (3-5) indica si un diámetro está siendo utilizado o no. Por ejemplo, si X_{ijT} =1 indica que está utilizándose, si no, entonces X_{ijT}=0. Cabe mencionar que la red puede tener diámetros repetidos.

El conjunto de restricciones en (3-6) indica si un tanque esta seleccionado, entonces Y_T=1, de lo contrario Y_T=0. Los tanques pueden tener diferentes características como: diámetro, nivel inicial, nivel máximo, nivel mínimo.

El conjunto de restricciones en (3-7) indica si una válvula V es insertada en una tubería J por lo tanto K_{VJ}*=1,* de lo contrario K_{VJ} *=0*. Las válvulas pueden tener diferentes características como: presión de tarado (*presión deseada aguas abajo*) y diámetro.

3.5 Modelo de satisfacción de restricciones

A continuación se presenta el modelo de satisfacción de restricciones, donde un conjunto de restricciones hidráulicas se deben cumplir para obtener el

funcionamiento adecuado del sistema y cumplir con las necesidades de los usuarios.

$$\sum_{i} Q_{in} - \sum_{i} Q_{out} = Q_e \tag{3-8}$$

$$\sum_{\mathrm{c}} \mathrm{h_f} = \sum_{\mathrm{c}} \mathrm{E_p} \tag{3-9}$$

$$\mathrm{H_{min}} \leq \mathrm{H_i} \leq \mathrm{H_{max}} \tag{3-10}$$

La ecuación (3-8) representa la ley de la conservación de la masa, donde la suma de flujos que entran y salen de un nodo es igual a cero. Entonces esta restricción indica que el caudal que entra en un nodo Q_{in} menos el caudal que sale del nodo Q_{out} es equivalente a la demanda Q_e.

La ecuación (3-9) representa la ley de la conservación de la energía. Esta restricción indica que la suma de las pérdidas de energía por fricción h_f en cualquier circuito c debe ser igual a cero o a la energía de potencia E_p considerada cuando se suministra por una bomba.

La restricción en (3-10) asegura las presiones necesarias en un sistema de distribución de agua. Esta restricción indica que la presión en un nodo H_i debe ser mayor que la presión mínima H_{min} requerida y menor que la presión máxima H_{max} requerida. Las presiones se definen de acuerdo a las características del problema, para la red del FRM las presiones mínimas son valores de 10 mca y los valores de las presiones máximas son 60 mca. Las presiones necesarias ayudan a tener un control adecuado de la distribución de agua, cumpliendo con el abastecimiento a los usuarios sin tener desperdicios y evitando rupturas en tuberías.

Capítulo 4

Metodología de Solución

Capítulo 4 Metodología de Solución

En esta sección describimos la metodología de solución propuesta para resolver el problema de distribución de agua de la red del FRM, esto con la finalidad de encontrar buenas soluciones y obtener el mínimo costo que implica rediseñar la red. Se consideran los siguientes pasos para la metodología de solución.

1. **Modelo de optimización.** Se propone una función objetivo y un conjunto de restricciones para solucionar la red del FRM (*ver capítulo 3 sección 3.4*).
2. **Estrategia de solución.** De acuerdo al análisis realizado a la red del FRM, se plantea una estrategia para solucionar el problema. Como primer punto, se agregan nuevos tanques de almacenamiento con las características necesarias. En segundo punto, se agregan válvulas reductoras de presión (PRVs), para tener un control adecuado de la presión en los sistemas. Otra de las estrategias que se podría implementar es redireccionamiento de diámetros de las tuberías, pero esta estrategia no se emplea, debido a que las tuberías ya existen, por lo tanto al realizar el cambio implicaría quitar las tuberías y sustituirlas por nuevas, lo cual genera un costo mayor.
3. **Algoritmo genético secuencial (AGS-FRM).** Se diseña y desarrolla un algoritmo genético que resuelva la red hidráulica del FRM. Considerando, para mejorar esta red, el incrementar tanques de almacenamiento y control del flujo de agua con base a válvulas reguladoras de presión. La configuración resultante es obtenida por el algoritmo genético.
4. **Algoritmo genético distribuido (AGD-FRM).** Se diseña y desarrolla un algoritmo genético distribuido, implementando el modelo Maestro-Esclavo

con la finalidad de mejorar la eficiencia computacional del algoritmo AGS-FRM.

5. **Sintonización de parámetros.** Se realiza el análisis de sensibilidad de los parámetros de control del algoritmo AGS-FRM, con el objetivo de encontrar los mejores valores para cada parámetro del algoritmo que permitan mejorar su desempeño. La sintonización de los parámetros se muestra en el capítulo 5.
6. **Pruebas Experimentales.** Se realizan las pruebas experimentales utilizando los parámetros de control sintonizados. Por cada prueba se realizan 30 ejecuciones para obtener un buen análisis estadístico del desempeño del algoritmo genético. Las pruebas experimentales se presentan en el capítulo 5.

4.1 Estrategia de solución

Después del análisis realizado a la red del FRM (Martínez-Bahena et al., 2014), se muestran las presiones en los nodos que no cumplen con el conjunto de restricciones del modelo 3.5, por lo que la solución no es factible debido a que no se realiza la distribución de agua de forma balanceada (ver capítulo 3 sección 3.3). Por lo tanto, se determinó la siguiente estrategia de solución:

Agregar nuevos tanques de almacenamiento con las características adecuadas en los puntos de la red donde no se cumpla con la presión mínima requerida. El beneficio de implementar tanques de almacenamiento con distribución del agua por gravedad, es reducir o eliminar el número de bombas y por lo tanto reducir el costo de la energía. El beneficio principal y de gran interés para este trabajo es incrementar la presión en el sistema de distribución de agua. Para ello, se determinan las coordenadas de los

posibles tanques de acuerdo a la ubicación topográfica de la red, esto con la finalidad de no agregar un tanque en aquellos puntos donde exista una construcción y no sea posible instalar una tubería o en una zona considerada de riesgo. Para el funcionamiento adecuado del sistema, es necesario establecer las características necesarias para los nuevos tanques de almacenamiento como es el diámetro (en metros), el diámetro es calculado mediante la fórmula (4-1), volumen (m^3), nivel máximo (rango de 5-15 m), nivel mínimo (rango de 0-10 m) y nivel inicial (rango de 2-10 m). La Figura 4-1 muestra los resultados obtenidos de la simulación realizada con Epanet. En esta simulación se agregaron 3 tanques de almacenamiento con diferentes características. La presión aumentó en aquellos nodos que no cumplían con la mínima presión requerida (*ver capítulo 3 sección 3.3*). Pero para que sea una solución factible debe cumplir con la presión máxima requerida. Se observa en la gráfica que existen presiones altas en varios puntos de la red.

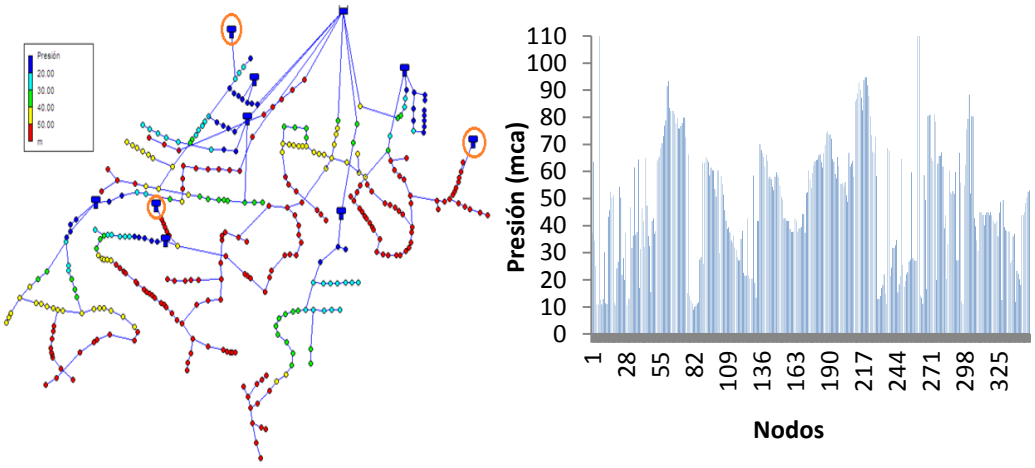

Figura 4-1 Tanques nuevos agregados y presiones en cada nodo

Agregar válvulas reguladoras de presión (PRV), las cuales permiten disminuir la presión en aquellos puntos donde es muy elevada, esto con la finalidad de garantizar un servicio balanceado a la comunidad y evitar

rupturas constantes de tuberías. Es importante mencionar que el problema de encontrar la localización óptima de válvulas en un sistema ha sido estudiado durante varios años por diferentes autores (Reis et al., 1997; Araujo et al., 2006; Nicolini y Zovatto, 2009; Cattafi et al., 2011; Dai y Li, 2014; Ali, 2015; Wright et al., 2015), con el objetivo de tener una mejor distribución de agua, además de que disminuye el costo de operación. Cuando se agrega una válvula se genera un costo económico, por lo tanto entre más válvulas se agreguen al sistema el costo se incrementa. En este trabajo se toma la idea de (Liberatore y Sechi, 2009), de solo agregar válvulas en las tuberías que conectan a nodos con presiones mayores a la presión máxima requerida. Cabe mencionar que cada tubería es candidata a insertar una válvula, pero debido a que al colocar una válvula en alguna tubería que conecte a un nodo donde no existe una presión alta, esta disminuye incluso hasta ser negativa, por lo que ya no cumpliría con la presión mínima requerida y hace de esta opción inviable.

Para la inserción de las válvulas se toman algunos puntos de (Hurtado-Guzmán, 2009), donde el sentido de la válvula es el sentido del flujo original del agua en la tubería seleccionada, el diámetro de la válvula es el mismo que el diámetro (22.7 -101.6) de la tubería elegida, la presión de tarado[2] es determinada en el rango (10-60 mca) esto con la finalidad de realizar la simulación lo más real posible.

La Figura 4-2 muestra los resultados de agregar válvulas reguladoras de presión. Para esta simulación se agregaron 20 válvulas y como resultado los nodos cumplen con la presión máxima requerida, por lo tanto la solución es factible ya que satisface las restricciones del modelo presentado en capítulo 3 sección 3.5. Después de agregar tanques de almacenamiento y válvulas reguladoras de presión se observa que las presiones en la red del FRM tienen una mejor distribución.

[2] Presión de tarado es la presión deseada aguas abajo

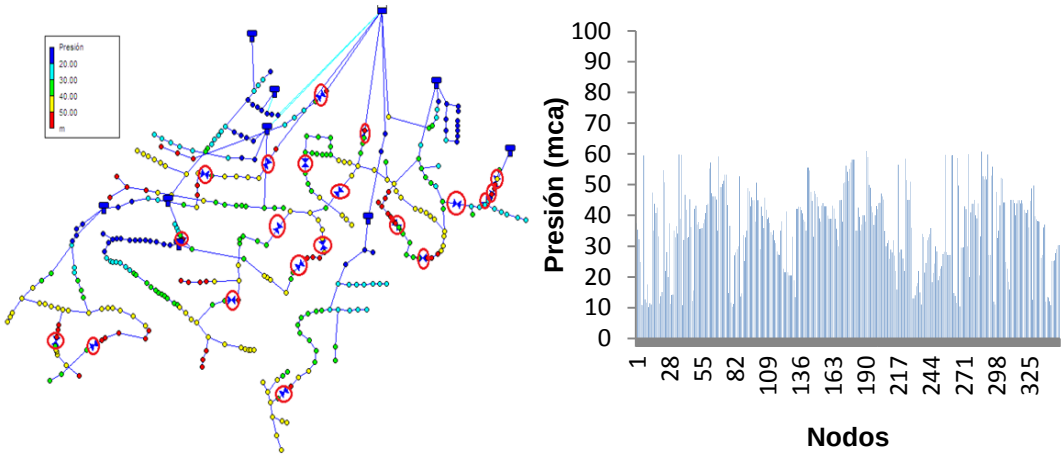

Figura 4-2 Solución factible de la red del FRM, cuando se agregan válvulas reguladoras de presión

La estrategia de solución propuesta obtiene soluciones factibles de acuerdo al modelo de satisfacción de restricciones (*ver capítulo 3 sección 3.5*), pero también se debe cumplir con el modelo de optimización (*ver capítulo 3 sección 3.4*). Para este problema la función objetivo es minimizar el costo que implica agregar tanques, tuberías y válvulas en un sistema existente. Para ello, es necesario utilizar metaheurísticas computacionales que obtengan buenas soluciones en un tiempo razonable. En este trabajo se implementa un algoritmo genético, debido a que se han obtenidos buenos resultados en otros trabajos de investigación sobre distribución de agua (Araujo et al., 2006; Cruz-Chávez et al., 2014; Ali, 2015), se presentan más investigaciones en el capítulo 2 sección 2.2.2.

4.2 Algoritmo genético

Esta sección describe el algoritmo genético propuesto en este trabajo. Los algoritmos genéticos, fueron inventados por (Holland, 1975) e introducidos por (Goldberg, 1989) como algoritmos de tipo evolutivo. Los AGs son

métodos adaptativos que se han utilizado ampliamente para resolver problemas de búsqueda y optimización. Estos algoritmos se basan en el proceso genético de los organismos vivos, donde a través de las generaciones las poblaciones evolucionan de acuerdo con los principios de la selección natural y la supervivencia del más apto propuesto por (Darwin, 1859).

Los algoritmos genéticos tienen varias ventajas en comparación con otros métodos. Estos algoritmos se pueden aplicar a una área amplia de problemas, obtienen buenos resultados en problemas del mundo real, tienen un gran potencial para combinarse con otros métodos de búsqueda y optimización, pueden adecuarse fácilmente al ambiente paralelo y distribuido, además sus operadores pueden adaptarse para cierto tipo de problemas (Hurtado-Guzmán, 2009; Reeves y Rowe, 2002).

4.3 Componentes del algoritmo genético

Los algoritmos genéticos se componen de varios elementos como población inicial, evaluación de población, operador de selección, cruzamiento y mutación, estos componentes hacen que el algoritmo tenga un buen funcionamiento, además de que se pueden adaptar a diferentes problemas de optimización combinatoria (Amor, 2008).

4.3.1 Representación de los individuos

Una solución para la red del FRM es representada como un individuo compuesto varios cromosomas. Un cromosoma es un conjunto de genes. Un gen representa un valor (*binario, entero o real*) de la solución. La
Figura 4-3 muestra la representación gráfica de un individuo, el individuo tiene tres cromosomas (*válvulas, tuberías y depósitos*) y el valor de aptitud (ver *Ec.*(3-2)). Para el cromosoma de válvulas, el valor del gen es la presión de tarado. En el cromosoma de tuberías, el valor del gen es el diámetro de la

tubería y el cromosoma de depósitos, está conformado de varios genes (*nivel inicial, nivel final, nivel máximo y diámetro*).

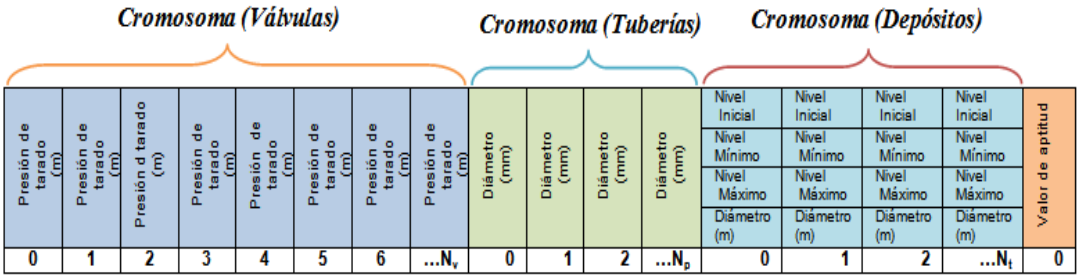

Figura 4-3 Representación de un individuo

4.3.2 Generación de la población inicial

La población inicial se puede generar mediante alguna técnica heurística o aleatoriamente, en este trabajo se genera de forma aleatoria. Los individuos de la población son evaluados para verificar su factibilidad. ¿Cómo saber si una solución es factible?, si cumple con todas las restricciones del modelo presentado en el *capítulo 3 sección 3.5*, ¿Por qué trabajar con una población de individuos factibles?, porque se ha comprobado que cuando se trabaja con soluciones factibles para este tipo de problemas el comportamiento del algoritmo mejora considerablemente (Cruz-Chávez et al., 2014).

La Figura 4-4 muestra el algoritmo que genera la población inicial de soluciones factibles.

1.- Primeramente se carga la instancia (datos) del problema, se inicializan las variables P_{ini} y N_{inds}.

2.- Posteriormente se genera un individuo de forma aleatoria, donde las características de los tanques son generadas de forma aleatoria, de las nuevas tuberías y también la activación y características de las válvulas en las tuberías candidatas.

3.- Para que el algoritmo genético utilice Epanet se crea un archivo con extensión ".inp", el cual contiene todos los datos de la red del FRM. En el Apéndice C se presenta un ejemplo de un archivo en formato ".inp".

4.- Después el archivo con extensión ".inp" es leído por Epanet para cargar los datos y así poder realizar la simulación hidráulica. Al terminar la simulación, Epanet arroja la presión obtenida en cada nodo, el caudal en cada tubería, el nivel de agua en cada tanque, entre otros datos. Posteriormente se procede a verificar si la solución cumple con el conjunto de restricciones (*ver capítulo 3 sección 3.5*), si es así, se considera una solución factible y se agrega a la población inicial P_{ini}. De lo contrario se realiza la factibilidad del individuo, mediante algunas modificaciones en las características de los tanques y configuración de las válvulas. Si después de un cierto número de iteraciones no se alcanza la factibilidad del individuo, entonces se procede a generar un nuevo individuo de forma aleatoria.

5.- Posteriormente se incrementa $\mathrm{N_{inds}}$ que es un contador de individuos factibles. El mismo procedimiento se genera hasta que la población inicial P_{ini} sea completada.

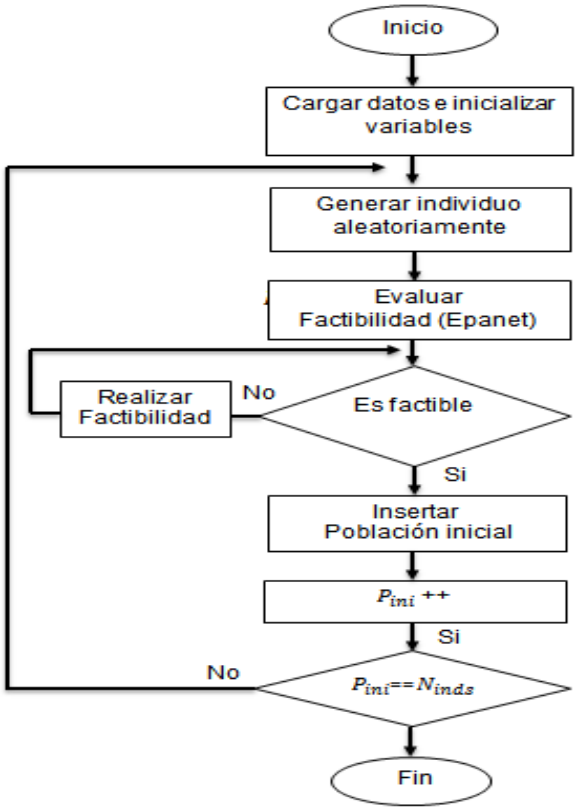

Figura 4-4 Algoritmo para generar la población inicial

4.3.3 Evaluación de la factibilidad de la población

La evaluación de la población es muy importante porque determina si un individuo cumple con las características adecuadas para formar parte de la población. La factibilidad de cada individuo es con respecto al modelo de satisfacción de restricciones presentado en el *capítulo 3 sección 3.5*. El modelo es evaluado mediante el simulador Epanet para verificar que cumpla con la ley de la conservación de masa en cada nodo, la ley de conservación de energía en cada circuito y las presiones necesarias en cada nodo para que un sistema de distribución de agua funcione correctamente. Para la red del FRM se considera una solución factible, si cada nodo cumple con una presión mínima de 10 mca y una presión máxima de 60 mca.

Para incorporar el algoritmo genético con Epanet, se utiliza el Kit de herramientas del programador Epanet (*Epanet Programmer's Toolkit*), el cual es una colección de funciones que permiten simplificar la programación para el análisis de la red de distribución de agua. Las funciones del kit de herramientas (Toolkit) se pueden implementar en diferentes entornos de desarrollo como C/C++, Visual Basic, Delphi, Pascal, bajo un ambiente Windows y Linux (Rossman, 1999). El código fuente de Epanet, escrito en Lenguaje C se puede descargar desde el sitio oficial (EPANET, 2016).

En esta investigación se implementa el simulador Epanet en ambiente Linux, para ello se realizaron los siguientes pasos(Ávila-Melgar et al., 2016):

1. Descargar el código fuente de Epanet del sitio web (EPANET, 2016). Se elige el archivo de Epanet de acuerdo a las características del equipo de cómputo, ya sea para sistemas de 32 bit o 64 bit.
2. Descomprimir la carpeta descargada utilizando la instrucción: *tar -xzvf filename-epanet.tar.gz*

3. Incorporar el kit de herramientas de Epanet, llamado “toolkit.h” para que se ejecute en ambiente Linux. Para ello es necesario crear una librería estática. Se realizan los siguientes puntos:
   I. Compilar los archivos que se encuentran dentro de la descarga del código fuente de Epanet (archivos con extensión .c) para obtener los archivos objeto (archivos con extensión .obj) mediante la siguiente instrucción:
      gcc -c -o objeto.o hola.c
   II. Empacar los archivos objeto mediante la instrucción: ***ar –rcs libstaticname.a filename1.o, filename2.o, filenameN.o*** donde “*libstaticname*”, es el nombre que se le quiera dar a la librería estática. La instrucción se debe ejecutar dentro de la carpeta que contiene los archivos (epanet.o hash.o hydraul.o inpfile.o input1.o input2.o input3.o mempool.o output.o quality.o report.o rules.o smatrix.o). Ejemplo:
      ar –rcs libepanet.a epanet.o hash.o hydraul.o inpfile.o …
   III. Copiar la librería estática creada en el mismo directorio donde se encuentra el algoritmo genético, para evitar la configuración de variables de entorno.
   IV. Incluir la librería “toolkit.h”, dentro del código fuente del algoritmo genético.
   V. Compilar el algoritmo genético secuencial con Epanet en ambiente Linux con la siguiente instrucción:
      gcc –o GAS GAS.c epanetsl.a -lm
      El algoritmo genético distribuido se compila utilizando mpicc con la siguiente instrucción:
      mpicc –o GAD GAD.c epanetsl.a -lm

El Algoritmo 4-1 lee los datos de la red del FRM por medio de Epanet, creando un archivo con la extensión ".inp", línea 2 al 3. Después en la línea 4 se lee el archivo mediante la función *ENopen (),* se verifica si el archivo existe (Línea 5 y 6), si es así, se procede a cargar los datos en el simulador Epanet con la función *ENopenH ()* línea 8, después se obtiene el número total de enlaces y nodos de la red (línea 9 y 10).

Algoritmo 4-1 Leer datos de la red

```
1: Entrada: Configuración
2  Crear_archivo_INP( Configuración) //El archivo es en formato .inp
3: sprintf(ArchINP, "IND-%d-%d.inp", ind, rank);
4: Error <- ENopen(ArchINP," ","");//Leer archivo .inp
5: Si (Error>0) Entonces
6:     ENclose()
7: Sino
8:     ENopenH()
9:     ENgetcount(EN_LINKCOUNT, &NtotalLinks);// número de enlaces
10:   ENgetcount(EN_NODECOUNT, &NtotalNodes);//número de nodos
11:Fin Si
```

En el Algoritmo 4-2 se actualizan las características de los nuevos tanques y de las nuevas tuberías. Primeramente se inicializan todas las variables de entrada (línea 1 a la 3). Después se actualizan los datos de cada uno de los nuevos tanques de la red. Estos datos son altura, nivel inicial, nivel mínimo, nivel máximo y el diámetro (línea 5 a la línea 9). También se modifican los datos de las nuevas tuberías, que se agregaron de acuerdo al número de tanques. Estos datos son diámetro y nodo destino (línea 11 y 12), se incrementa el número de tanque, número enlace y número iteraciones (línea 13 a la 15). Este procedimiento se repite hasta el número total de los nuevos tanques (*NtotalNodes).*

Algoritmo 4-2 Actualizar datos de los tanques

```
1: Entrada: POB, NtotalNodes, NtotalLinks, ind, NVERTICES,NDEPS,NEMBS, NTUBERIAS
2: tank := NVERTICES+NDEPS+NEMBS+1
3: link := NTUBERIAS+1
4: Mientras tank< NtotalNodes Hacer
5:        ENsetnodevalue(tank,EN_ELEVATION, POB[ind].DEPS[i].altura)
6:        ENsetnodevalue(tank, EN_MINLEVEL,POB[ind].DEPS[i].NivelMin)
7:        ENsetnodevalue(tank, EN_MAXLEVEL,POB[ind].DEPS[i].NivelMax)
8:        ENsetnodevalue(tank, EN_TANKLEVEL,POB[ind].DEPS[i].NivelIni)
9:        ENsetnodevalue(tank, EN_TANKDIAM,POB[ind].DEPS[i].diametro)
10:       /*Datos para la nueva tubería*/
11:       ENsetlinkvalue(link, EN_DIAMETER, POB[ind].TUBS[i].diametro)
12:       ENsetlinkvalue(link, EN_NODEEND, POB[ind].TUBS[i].Vdestino)
13:       link := link +1
14:       tank:= tank + 1
15:       i:= i +1
16: Fin Mientras
```

La función ENsetnodevalue (tank, EN_TANKDIAM, POB [ind].DEPS[i].diametro), permite modificar el diámetro de los nuevos tanques, para ello se realiza una modificación dentro del código libre de Epanet (escrito en lenguaje C). Epanet permite la simulación de tanques circulares, pero para la solución de la red del FRM, se implementan tanques rectangulares, por lo que es necesario realizar la simulación con Epanet mediante tanques rectangulares. Para los depósitos cuadrados o rectangulares, el diámetro equivalente es 1.128 veces la raíz cuadrática de la sección transversal.

El volumen de un depósito circulante es:

$$V_{dep.circular} = \frac{\pi.d^2}{4}.h, \text{ donde } d = diámetro\ (m) \text{ y } h = altura(m) \qquad (4\text{-}2)$$

El volumen de un depósito rectangular es:

$$\boldsymbol{V_{dep.rect} = a.b.h}, \text{ donde } \boldsymbol{a = largo(m)} \text{ y } \boldsymbol{b = ancho(m)} \qquad (4\text{-}3)$$

Para trabajar con Epanet, el volumen del depósito circular debe ser igual al del depósito rectangular, por lo tanto

$$\boldsymbol{V_{dep.circular} = V_{dep.rect}} \qquad (4\text{-}4)$$

(4-5)

Sustituyendo $\frac{\pi.d^2}{4}.\boldsymbol{h} = \boldsymbol{a}.\boldsymbol{b}$.

Despejando para obtener un diámetro equivalente

$$\boldsymbol{d_{eq}} = \sqrt{\frac{4.a.d}{\pi}} = \mathbf{1.128}\sqrt{\boldsymbol{a}.\boldsymbol{b}} \quad (4\text{-}6)$$

Al verificar el módulo “EN_TANKDIAM” se observa que de acuerdo a la fórmula del diámetro equivalente de depósitos rectangulares, la ecuación dentro del código no es la adecuada, ya que el factor constante $4/\pi$ aparecía elevado a la primera potencia en lugar del valor de su raíz cuadrada, por lo que se modificó al valor 1.128 correspondiente (véase Tabla 4-1).

Tabla 4-1 Comparación del código original y código modificado del módulo “EN_TANKDIAM”

Código original	Código modificado

case EN_TANKDIAM:
v = 0.0;
if (index > Njuncs){
v=4.0/PI*sqrt(Tank[index-Njuncs].A)*Ucf [ELEV];
} **break**;
	case EN_TANKDIAM:
	v = 0.0;
	if (index > Njuncs){
	v=sqrt (4.0/PI)*sqrt (Tank [index-Njuncs].A)*Ucf [ELEV];
	} **break**;

La función ENsetlinkvalue (link, EN_NODEEND, POB [ind].TUBS[i].Vdestino), permite modificar las características de las tuberías, válvulas y bombas. Dentro del código fuente del simulador Epanet se agrega el módulo “EN_NODEEND” para la configuración correcta de los tanques de almacenamiento y así poder obtener soluciones a la red del FRM. El módulo agregado permite modificar la conexión de un tanque con cualquier otro nodo de la red (véase Algoritmo 4-3).

Algoritmo 4-3 Módulo EN_NODEEND

```
case EN_NODEEND:
    if (Link [index].Type <= PIPE) {
        Link [index].N2 =value;
        Reactflag = 1;
    }
break;
```

Después de las modificaciones realizadas al código fuente del simulador Epanet, es necesario volver a compilar los archivos con extensión ".c", para crear los archivos ".o" y así poder crear la librería estática del simulador Epanet, para ello se siguen los mismos pasos inicialmente aquí detallados (Ávila-Melgar et al., 2016).

En el Algoritmo 4-4 se actualizan las características de las válvulas y el estado de las tuberías mediante un conjunto de funciones. De la línea 3 hasta la línea 10 se actualiza el estado de todas las tuberías en la red, donde estado=1 es cuando la tubería está abierta y estado=0 indica que está cerrada. De la línea 11 hasta la 21 permite activar o cerrar las válvulas. Donde función ENsetlinkvalue (index, EN_INITSETTING, POB[ind].VALS[v]), activa una válvula, modificando el módulo EN_INITSETTING mediante la presión de tarado[3].

Algoritmo 4-4 Actualizar datos de las válvulas

```
1:  Entrada:  NTUBERIAS, POB, NTUBSNUEVAS, NtotalLinks
2:  tub<-0, v<-0
3:  Mientras tub < NTUBERIAS  Hacer
4:          Si (POB[ind].TUBSTEMP[tub]==1) Entonces
5:                  ENsetlinkvalue(tub,EN_INITSTATUS,1)
6:          Sino
7:                  ENsetlinkvalue(tub,EN_INITSTATUS,0)
8:          Fin Si
9:          tub <- tub + 1
10: Fin Mientras
11: index<-NTUBERIAS+NTUBSNUEVAS+1
12: Mientras index < NtotalLinks  Hacer
13:       Si (POB[ind].VALS[v]!=0) Entonces
14:               ENsetlinkvalue(index, EN_INITSETTING,POB[ind].VALS[v])
15:         Sino
16:               ENsetlinkvalue(index,EN_INITSETTING,0)
```

[3] Presión de tarado es la presión deseada aguas abajo

```
17:             ENsetlinkvalue(index,EN_INITSTATUS,0)
18:         Fin Si
19:         v<-v +1
20:         index <- index +1
21: Fin Mientras
```

El Algoritmo 4-5 permite obtener las presiones arrojadas en cada nodo después de la simulación hidráulica de la red mediante la función *ENsolveH*(). Al realizar la simulación se obtienen diferentes parámetros como caudales, velocidades y presiones. La función ENgetnodevalue (nodo, EN_PRESSURE, &pressure), permite obtener la presión alcanzada en cada nodo de la red (línea 8). Se verifica que cada nodo cumpla con la presión mínima requerida (línea 9) y así determinar cuántos no cumplen con la condición (línea 10). También se contabiliza los nodos que no cumplan con la presión máxima requerida (línea 12 hasta línea 15) para poder agregar válvulas a esas tuberías y así regular la presión en el sistema. Después se incrementa la variable *nodo* hasta el número total de nodos (línea 16).

Algoritmo 4-5 Obtener presiones de la simulación

```
1: Entrada:  NVERTICES, PRESIONMIN, PRESIONMAX, PresionNodos
2: presMin<-0,presMax<-0, pressure<-0, node<-0,nodo<-0
3: ENsolveH()
4: Mientras nodo < NVERTICES Hacer
5:      pressure<-0
6:      ENgetnodeid(nodo, id)
7:      node<- atoi (id)//convertir de char a int
8:      ENgetnodevalue(nodo, EN_PRESSURE, &pressure)
9:      Si (pressure<PRESIONMIN) Entonces
10:         presMin<- presMin + 1
11:     Fin Si
12:     Si (pressure>PRESIONMAX) Entonces
13:           PresionNodos[presMax]<-node
14:           presMax <- presMax + 1
15:     Fin Si
16:     nodo <- nodo +1
17:Fin Mientras
```

4.3.4 Evaluación de población

El valor de aptitud de cada individuo corresponde al costo que implica realizar los cambios necesarios para obtener buenas soluciones (*véase el capítulo 3 sección 3.5*). Donde la función objetivo es minimizar el costo que se genera al agregar nuevos componentes como son tanques, válvulas y tuberías al sistema de distribución. La función objetivo está sujeta a un conjunto de restricciones que se deben cumplir al contemplar cualquiera de los componentes.

Los algoritmos genéticos trabajan con tres operadores básicos como: selección, cruzamiento y mutación. La implementación de cada uno de ellos para este trabajo se describe a continuación.

4.3.5 Selección

El método de selección utilizado en este algoritmo es por torneo, ya que en diferentes problemas de optimización ha dado buenos resultados (Amor, 2008; Ávila-Melgar, 2015).

El método de selección por torneo consiste en elegir un subconjunto de n individuos de una población, donde n puede tomar valores de dos o más individuos (Baños, 2006). Es un método eficiente y fácil de implementar, con complejidad computacional del algoritmo de $O(n)$. Esta técnica presenta como ventaja respecto a otras técnicas de selección su eficiencia computacional ya que no requiere escalamiento de la función de aptitud (usa comparaciones directas) por lo cual no requiere la ordenación de los individuos para poder seleccionarlos con base en su aptitud. Además, este método puede implementarse en paralelo (Ávila-Melgar, 2015).

Este método selecciona un conjunto de n individuos de forma aleatoria, posteriormente se comparan de acuerdo a su valor de aptitud, el más apto

es elegido como un individuo padre. Se realiza el mismo procedimiento para seleccionar al individuo madre. Comúnmente, la muestra de selección por torneo es $n = 2$ (Amor, 2008).

El Algoritmo 4-6 describe el proceso del operador de selección por torneo. Se generan dos individuos de forma aleatoria (línea 4 y 5). Si los individuos son iguales, se selecciona un nuevo individuo (línea 6 a la 8), así hasta que sean diferentes. Después se comparan si el ind1 es menor o igual que el ind2, entonces el ind1 se guarda para posteriormente ser cruzado, de lo contrario se guarda el ind2 (línea 9 a la línea 12). Se incrementa el número de iteraciones (línea 14). Este procedimiento se repite hasta completar el número de porcentaje de selección $(\mathrm{P_m})$

Algoritmo 4-6 Operador de selección por torneo

```
1: Entrada: Pm, NINDS, FITNESS
2: iter<-0, ind1<-0, ind2<-0
3: Mientras iter <Pm Hacer
4:        ind1=rand()%NINDS
5:        ind2=rand()%NINDS
6:        Mientras  ind1==ind2 Hacer
7:              ind2=rand()%NINDS
8:        Fin Mientras
9:        Si(FITNESS[ind1]<=FITNESS[ind2] ) Entonces
10:             individuo_seleccionado=ind1
11:       Sino
12:             individuo_seleccionado=ind2
13:       Fin Si
14:      iter:=iter+1
15:Fin Mientras
```

4.3.6 Cruzamiento

El operador de cruzamiento consiste en intercambiar características de dos padres para generar dos descendientes. En este trabajo se propuso un operador de cruzamiento llamado intercambio genético entre individuos (*IGEI*), debido a que los individuos son de diferente tamaño y contienen varios cromosomas, por lo tanto no era factible usar los operadores básicos (de un punto, dos puntos o múltiples puntos).

En la Figura 4-5 se presenta un ejemplo del comportamiento del operador *IGEI*. Para el cromosoma de válvulas, se elige de forma aleatoria una posición correspondiente a una válvula, para el primer individuo las posiciones son 1, 4 y 6, de igual manera para el segundo individuo las posiciones son 2, 4 y 5, después se hace el intercambio de gen (presión de tarado) entre {1,2, 4,4 y 6,5} así hasta una cierta cantidad de intercambios, la cual es determinada de forma aleatoria en función del total de válvulas. Para el cromosoma de tubería, se elige de forma aleatoria la posición de la tubería por ejemplo posición 0 y para el segundo padre, se selecciona la posición 1, se intercambia el valor de diámetro, así hasta una cierta cantidad de intercambios, la cual es determinada de forma aleatoria en función del total de tuberías. De igual manera para el cromosoma de tanques, se elige de forma aleatoria una posición correspondiente a un tanque para el primer individuo, por ejemplo posición 0 y de igual manera para el segundo se elige la posición 1, se intercambia los genes (Nivel inicial, final, máximo y diámetro), así hasta una cierto cantidad de intercambios, la cual es determinada de forma aleatoria en función del total de tanques.

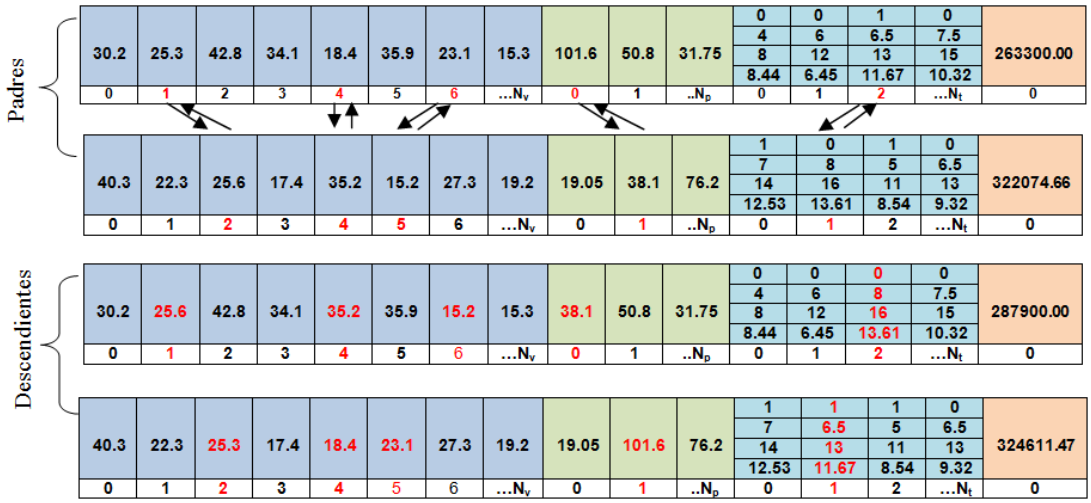

Figura 4-5 Procedimiento del operador de cruzamiento (IGEI)

En el Algoritmo 4-7 se muestra el proceso del operador cruzamiento. Primeramente se inicializan todas las variables de entrada al algoritmo (línea 1 y 2). Después se eligen dos individuos de forma aleatoria (línea 4 y 5), mientras sean iguales se selecciona un nuevo individuo (línea 6 al 8). Después se genera un número de acuerdo a la probabilidad de cruzamiento P_c (línea 9), se compara para verificar si se realiza el cruzamiento o no (línea 10). De la línea 1 a la 13 se realiza el cruzamiento entre los tanques y tuberías. Posteriormente se realiza el cruzamiento de las válvulas, donde se selecciona de forma aleatoria el número de genes y las posiciones de las válvulas a cruzar (línea 15 a la 33). Después de realizar el cruzamiento se obtienen dos individuos descendientes y se guardan en la población (línea 34 y 35). Si no se realiza el cruzamiento, los individuos padres seleccionados pasan directamente a la población (línea 37 y 38). Se incrementa el número de individuos (línea 36), el mismo procedimiento se realiza hasta completar la población de individuos.

Algoritmo 4-7. Pseudocódigo del operador de cruzamiento IGEI

```
1: Entrada: PCRUZ,POB,POBLACION, PSEL, NINDS
2: Ind<-0,ind1<-0, ind2<-0, NT<-0, Nvals1<-0, Nvals2<-0, Nvals<-0, posic<-0, posicVal1<-0, posicVal2<-0
3: Mientras ind <NINDS Hacer
4:          ind1<- rand()%PSEL
5:          ind2<- rand()%PSEL
6:          Mientras  ind1==ind2 Hacer
7:                    ind2 <- rand()%PSEL
8:          Fin Mientras
9:          pc <- (float)rand()/RAND_MAX
10:         Si(pc<PCRUZ) Entonces
11:                NT<- rand()%3+1
12:                Para tank<-0 Hasta NT Con Paso 1 Hacer
13:                          intercambioINFTanques(POB,ind1,ind2,tank)
14:                Fin Para
15:                Nvals1 <- NumeroValvulas(ind1,POB)
16:                Nvals2 <- NumeroValvulas(ind2,POB)
17:                Si (Nvals1<=Nvals2) Entonces
18:                      Nvals <- Nvals1
19:                Sino
20:                      Nvals <- Nvals2
21:                Fin Si
22:                posic <- rand()%Nvals
23:                Para i <- 0 Hasta posic Con Paso 1  Hacer
24:                          Repetir
25:                               posicVal1=rand()%NTotalVals
26:                          Hasta Que (POB[ind1].VALS[posicVal1]==0)
```

```
27:                     Repetir
28:                         posicVal2=rand()%NTotalVals
29:                     Hasta Que (POB[ind2].VALS[posicVal2]==0)
30:                     csgna=POB[ind1].VALS[posicVal1]
31:                     POB[ind1].VALS[posicVal1]=POB[ind2].VALS[posicVal2]
32:                     POB[ind2].VALS[posicVal2]=csgna
33:             Fin Para
34:             AgregarIndividuoPoblacion(POBLACION,POB,ind,ind1)
35:             AgregarIndividuoPoblacion(POBLACION,POB,ind+1,ind2)
36:         Sino
37:             AgregarIndividuosPoblacion(POBLACION, POB,ind,ind1)
38:             AgregarIndividuosPoblacion(POBLACION, POB,ind+1,ind2)
39:         Fin Si
40              ind<-ind +2
41: Fin Mientras
```

4.3.7 Mutación

El operador de mutación propuesto consiste en hacer una pequeña alteración aleatoria de los genes a un porcentaje de individuos P_m. La Figura 4-6 muestra un ejemplo del procedimiento del operador de mutación, en primer lugar, elige de forma aleatoria a un individuo de la población, después en el cromosoma de válvulas, se selecciona de forma aleatoria una posición, la cual corresponde a una válvula por ejemplo la posición 1,4 y 6, las cuales cambian el valor de 1 a 0, esto significa que esa válvula es eliminada de la tubería i. Para el cromosoma de tuberías, se refiere a las nuevas tuberías que permite conectar a los nuevos tanques con cualquier elemento de la red, de igual manera se selecciona de forma aleatoria una posición, por ejemplo la tubería 1 y se elige de forma aleatoria un diámetro de la lista de diámetros comerciales, en este caso el diámetro 50.8 fue cambiado por 38.1. Para el cromosoma de tanques, se elige de forma aleatoria un tanque, por ejemplo el tanque 0 y se modifica de forma aleatoria sus características (*nivel inicial, nivel mínimo, nivel máximo y diámetro*).

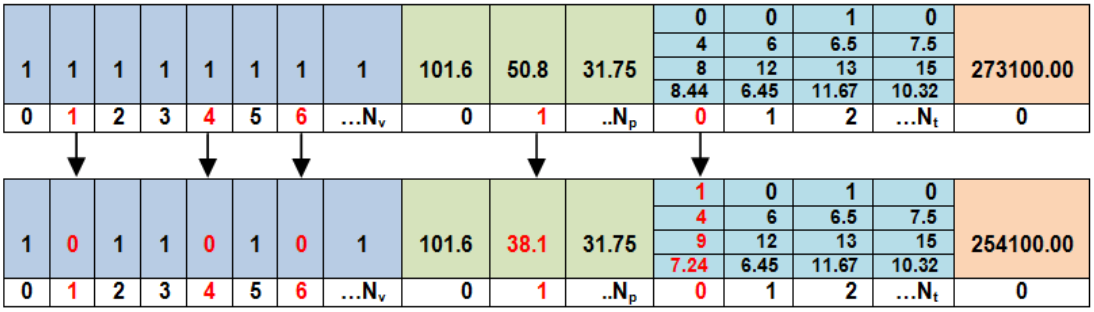

Figura 4-6 Proceso de mutación

En el Algoritmo 4-8 se muestra el proceso del operador de mutación. Primeramente se inicializan todas las variables de entrada al algoritmo (línea 1, 2 y 3). Después se elige un individuo de forma aleatoria (línea 5), se selecciona de forma aleatoria el número de genes a mutar (línea 6), pero de acuerdo a su probabilidad de mutación P_{mut} se determina si se muta el gen o no. Después se realiza una pequeña alteración a los genes del individuo, en el cromosoma válvulas (línea 7 a la línea 13). Posteriormente se realiza un cambio aleatorio en el cromosoma de tanques y tuberías, donde se selecciona de forma aleatoria el número de genes a mutar (línea 14 a la línea 16), de acuerdo a la cantidad total de genes que conforma el cromosoma. Una vez realizada la mutación se actualizan los campos modificados del individuo descendiente en Epanet (línea 18) para realizar la simulación hidráulica y así obtener la presión en cada nodo (línea 19). Si el individuo no cumple con las restricciones del modelo presentado en el capítulo 3 sección 3.5, entonces se considera un individuo no factible. Por lo que, se realiza la factibilidad del individuo (línea 20 a la 24) y se procede a guardar el individuo en la población (línea 23). Si el individuo después de la mutación es factible, entonces pasa directamente a formar parte de la población (línea 26). Este mismo procedimiento se realiza hasta completar el porcentaje de selección P_m, donde un contador lleva el control de los

individuos (línea 28). El resto del porcentaje de los individuos pasan directamente a la siguiente generación (línea 30 a la 33).

Algoritmo 4-8 Pseudocódigo del operador de mutación

```
1:Entrada: P_m, NINDST, POBTEMP, POB, NTotalVals
2:Iter<-0,Factible<-0,ind<-0,ind1<-0,posic<-0,posic1=0,gen<-0,t,genes<-0,
3:Vals1[NTotalVals],Nvals<-0,id_val<-0,val1<-0,posic_val<-0,NT<-0
4:   Para iter<- 0 Hasta P_m Con Paso 1 Hacer
5:        ind<- rand()%(NINDST/2)
6:        val1=NumeroValvulas (ind)
7:        genes<- rand()%val1
8:        Para gen<- 0 Hasta genes Con Paso 1 Hacer
9:             Repetir
10:                 posic_val=rand()%val1;
11:            Hasta Que (Vals1[posic_val]==-1)
12:            id_val<- Vals1[posic_val]
13:            MutacionIndividuo (posic, gen, POBTEMP, id_val)
14:       Fin Para
15:       NT<- rand()%3+1
16:       Para t <- 0 Hasta NT Con Paso 1 Hacer
17:            MutTanques(posic, POBTEMP, t)
18:       Fin Para
19:       CambiosDepositos(posic, POBTEMP)
20:       ObtenerPresiones(0)
21:       Si (presMin>=1 || presMax>=1) Entonces
22:            Factible<- FactibilidadInds(posic,POBTEMP,0)
23:            Si (Factible==1)Hacer
24:                 AgregarIndividuosPoblacion(POB,POBTEMP,posic1,posic)
25:            Fin Si
26:       Sino
27:        AgregarIndividuosPoblacion(POB,POBTEMP,posic1,posic)
28:       Fin Si
29:            posic1<- posic1 +1
30:  Fin Para
31:       Para ind <- posic1 Hasta NINDST/2 Con Paso 1 Hacer
32:            ind1<- rand()%(NINDST/2)
33:            AgregarIndividuosPoblacion(POB,POBLACION,ind,ind1)
34:       Fin Para
```

4.4 Algoritmo genético secuencial

El Algoritmo 4-9 muestra el pseudocódigo del algoritmo genético secuencial. En primer lugar, el algoritmo inicializa los parámetros de entrada, tales como probabilidad de cruce (P_c), probabilidad de mutación (P_{mut}), tamaño de la población (T_{pob}), número de generaciones (N_g), porcentaje de selección (P_s) y porcentaje de mutación (P_m), carga los datos de la instancia y crea la

configuración inicial. Los valores que toma esta configuración inicial dependen del análisis de sensibilidad que se realiza para los parámetros de entrada. Después genera una población inicial aleatoria de individuos (línea 5).Los individuos de la población son evaluados para verificar su factibilidad (línea 6), utilizando funciones de Epanet (*ver sección 4.3.3*). Posteriormente se evalúa el valor de aptitud (*ver Ec.*(3-2)) de cada individuo (línea 7). Los individuos evolucionan durante generaciones a través de tres operadores como: selección, cruzamiento y mutación. El método de selección que se utiliza es por torneo (línea 12), donde selecciona a los individuos más aptos. Después los padres seleccionados son recombinados para formar individuos descendientes que heredan las características de sus padres utilizando el operador de cruce *IGEI* (línea 13), debido a un factor de probabilidad (P_c), el cual determina si el cruce se lleva a cabo o no. Después del cruzamiento se evalúa la factibilidad de los individuos mediante Epanet (*véase sección 4.3.3*). Más tarde, se realiza la mutación a un porcentaje (P_m) de individuos de la población total (línea 15). Después de la mutación, nuevamente se evalúa la factibilidad de los individuos. Se agregan a la nueva población los individuos factibles (línea 17) hasta completar el tamaño de la población. Se evalúa el valor de aptitud de la nueva población (línea 20). Se reemplaza la nueva población e incrementa el número de generación. La Figura 4-7 presenta el diagrama de flujo del algoritmo genético secuencial, en el cual se sigue el flujo de ejecución de acuerdo a los pasos presentados en el Algoritmo 4-9.

Algoritmo 4-9. Pseudocódigo del algoritmo genético

1: Parámetros (P_c, P_{mut}, P_{tam}, N_g, P_s, P_m);*//Inicializar los parámetros de entrada*
2: CargarDatos *(); //Carga los datos de la instancia*
3: ConfiguracionInicial (); *//Genera configuración inicial*
4: *G*=0;*// Número de pasos*
5: Generar (*P*);*//Población inicial*
6: EvaluacionFactibilidad (*P*); *//Evaluación con Epanet*
7: EvaluarFitness*(P); //Evaluación de población (aptitud= (P,T,V))*
8: **Mientras** *G* <N_g **Hacer**
9: *P'*=0;*//Inicializar población P'*
10: MejorIndividuo*(*$MEJOR_{IND}$*, P); //Guarda el mejor individuo*
11: **Mientras** N_{inds}<P_{tam} **Hacer**
12: Selección (*P*, p_1, p_2);*//Seleccionar padres p1, p2 de la población P*
13: {h_1, h_2}=Cruzamiento (p_1, p_2, P_c);*//Cruzamiento de los padres* p_1, p_2
14: EvaluacionFactibilidad (h_1, h_2); *//Evaluación con Epanet*
15: Mutación (h_1, h_2, P_{mut});*//Mutación de los descendientes* h_1, h_2
16: EvaluacionFactibilidad (h_1, h_2); *//Evaluación con Epanet*
17: Agregar (*P'*, h_1, h_2);*//Insertar los descendientes* h_1, h_2 *a la población P'*
18: N_{inds}+=2;
19: **Fin Mientras**
20: EvaluarFitness*(P); //Evaluación de población (aptitud= (P,T,V))*
21: ReemplazoPoblacion (*P, P'*);
22: *G*+=1;*//Incremento de generación*
23: **Fin Mientras**

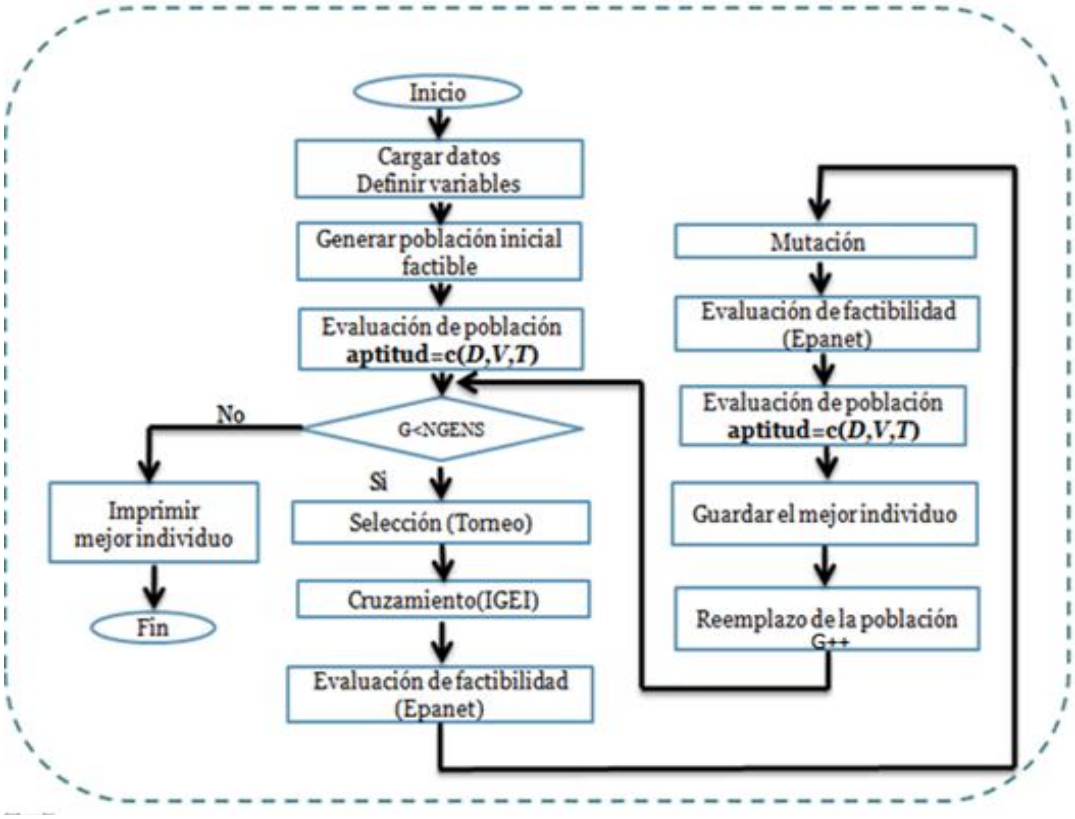

Figura 4-7 Algoritmo genético secuencial

4.5 Algoritmo genético distribuido

Para la elaboración del algoritmo genético distribuido se utilizó el modelo Maestro-Esclavo mediante comunicación colectiva con la biblioteca MPI.

4.5.1 Comunicación Colectiva

Message Passing Interface (por sus siglas en inglés, MPI), es una biblioteca fácil de usar y portable que se utiliza para la comunicación entre procesos (Barney, 2015). La estructura básica de un programa con MPI es la siguiente (Gropp et al., 1999):

1. Inicializar la comunicación
   - **MPI_Init ()** inicializa el ambiente MPI
   - **MPI_Comm_size ()** retorna el número de procesos
   - **MPI_Comm_rank ()** retorna el número de cada procesador
2. Comunicar para compartir datos entre procesos
   - **MPI_Send ()** envía un mensaje
   - **MPI_Recv ()** recibe un mensaje
3. Finalizar el ambiente paralelo
   - **MPI_Finalize ()** finaliza el ambiente paralelo que provee MPI.

Existen diferentes tipos de comunicación entre procesos mediante la biblioteca MPI. La comunicación punto a punto se presenta cuando existe un procesador que recibe y otro que envía mensajes. Se requiere de dos funciones que permitan la comunicación entre un grupo de procesadores, función de envío y recepción *(send/recv)*. Otro tipo de comunicación es la colectiva que permite la distribución de un dato a muchos procesadores que pertenecen al comunicador indicado. Esta comunicación puede o no sincronizar las operaciones incluidas. Un envío de información de un proceso a otro representa un arreglo de cualquier tipo de dato. Las

funciones usadas para una comunicación colectiva, contienen 2 parámetros para declarar el tipo de dato que se distribuirá en los procesadores, ambos parámetros deben estar declarados con el mismo tipo de dato. La comunicación colectiva utiliza las siguientes funciones para la distribución de los datos entre los procesos.

- Difusión de datos *(Broadcast)*
- Dispersión de datos *(Scatter)*
- Reunión de datos *(Gather)*
- Reunión en todos *(Allgather)*
- Todo a todos *(all to all)*
- Reducción *(Reduce* y *Allreduce)*

Para el diseño y desarrollo del algoritmo AGD-FRM se implementa la comunicación colectiva utilizando la función de *Scatter* y *Gather*. Para la sincronización de la comunicación entre los procesos se utiliza la función *MPI_Barrier (MPI_COMM_WORLD)* la cual hace una pausa mientras todos los procesadores del comunicador *com* terminan sus actividades. La ventaja de utilizar comunicación colectiva es porque el envío es más sencillo debido a que se genera un solo paquete de datos para la distribución en cada proceso, por lo que permite reducir la latencia generada en el envío.

En la función de dispersión de datos (**Scatter**) un procesador raíz distribuye los elementos de un arreglo hacia todos los procesos. Cada elemento del arreglo es dividido en n procesos, por lo que el número de procesadores debe ser igual a n números de elementos del arreglo.

La Figura 4-8 muestra el procedimiento de la función Scatter, como se observa el proceso P0 tiene un arreglo de 4 elementos, este proceso se considera Maestro, el cual distribuye un elemento a los procesos Esclavo, es decir proceso P1, P2 y P3. Para el problema de la red del FRM, el proceso P0 tiene un arreglo de subpoblaciones, cada subpoblación tiene n número

de individuos. El proceso P0 envía una subpoblación a los demás procesos, es decir P1, P2 y P3 reciben una subpoblación de n individuos, el número de subpoblaciones es igual al número de procesos.

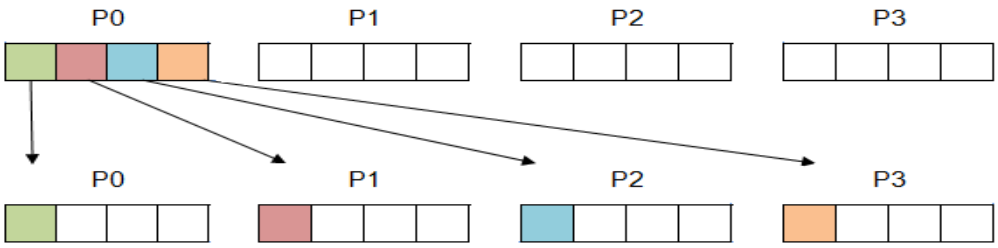

Figura 4-8 Procedimiento de la función Scatter

Sintaxis de la función Scatter:

MPI_Scatter (void **buffer_env***,* **int** envio_cont**, MPI_Datatype** tipodato_envío**, void** **buffer_rcv*, **int** recepcion_cont, **MPI_Datatype** tipodato_rcv, **int** *raiz*, **MPI_Comm** com**)**

La Tabla 4-2 presenta una explicación de los parámetros de entrada a la función Scatter, estos parámetros de acuerdo al problema de la red del FRM son: buffer_env (*Poblacion*), envío_cont (*NINDS*), tipodato_envío (*Individuo*), buffer_rcv (*SubPoblacion*), recepcion_cont(*NINDS*), tipodato_rcv (Individuo), raiz(*0*), com (*MPI_COMM_WORLD*).

Tabla 4-2 Variables de entrada a la función Scatter

Variable	**Descripción**
buffer_env	La dirección donde inicia el dato
envío_cont	El número de elementos que se enviará a cada procesador
tipodato_envío	El tipo de dato de cada elemento dentro del buffer
buffer_rcv	La dirección del buffer que se recibe
recepcion_cont	El número de elementos en el buffer de recepción

tipodato_rcv	El tipo de dato dentro del buffer de recepción
Raiz	El rango del procesador que envía
Com	El comunicador que contiene a los procesadores

En la función reunión de datos (**Gather**) todos los procesadores del comunicador envían datos al procesador raíz, es decir cada procesador envía sus datos al mismo procesador, es lo contrario de la función Scatter. La Figura 4-9 muestra el comportamiento de la función, donde el proceso *P0* se encargar de recopilar los elementos que le envían cada uno de los procesos Esclavo *P1, P2* y *P3*. Para el problema de la red del FRM, el proceso P0 recibe en un arreglo de subpoblaciones cada una de las subpoblaciones enviadas por los demás procesos, es decir P1, P2 y P3 envían una subpoblación de n individuos. La descripción de cada variable de la función Gather se presenta en la Tabla 4-2.

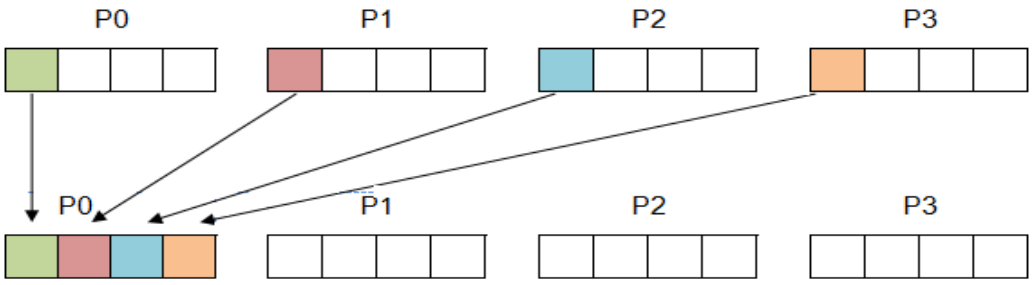

Figura 4-9 Comportamiento de la función Gather

Sintaxis de la función Gather:

MPI_Gather (**void** **buffer_env*, **int** envio_cont, **MPI_Datatype** tipodato_envío, **void** **buffer_rcv*, **int** recepcion_cont, **MPI_Datatype** tipodato_rcv, **int** *raiz*, **MPI_Comm** com);

4.5.2 Modelo Maestro Esclavo

El modelo **Maestro-Esclavo**, también conocido como "**paralelización global**" (Alba, 2013), permite mantener la secuencialidad del algoritmo original. Un procesador Maestro centraliza la población y gestiona la selección y los reemplazos de los individuos, este también se encarga del envío de subpoblaciones a los esclavos, los cuales ejecutan tareas de evaluación y aplicación de algunos operadores. Utilizado cuando el costo de generar y evaluar nuevas soluciones es elevado (Alba, 2013).

Se utilizó este modelo debido a que es recomendable cuando el algoritmo presenta tareas con alto costo computacional como por ejemplo la generación y la evaluación de nuevos individuos. La desventaja principal del modelo maestro/esclavos que debe considerarse es que al tener un solo maestro que se encarga de la coordinación y administración de los procesos, los esclavos se comunican con un solo nodo ocasionando problemas de latencias por las comunicaciones frecuentes. Así mismo un proceso maestro puede permanecer ocioso durante un periodo de tiempo y muy saturado de actividades en otro periodo de tiempo (Ávila-Melgar 2015).

La Figura 4-10 presenta el modelo Maestro-Esclavo del algoritmo genético para resolver la red hidráulica del FRM. El nodo maestro se encarga de cargar los datos de la instancia, genera la población inicial factible, evalúa el costo de la población, aplica el operador de selección y cruzamiento. Posteriormente divide la población total en subpoblaciones de acuerdo al número de procesos. El nodo maestro se encarga de enviar una subpoblación por proceso. Los nodos esclavo reciben la subpoblación para realizar la evaluación de factibilidad mediante el software Epanet, se genera la mutación y nuevamente se evalúa la factibilidad, así como la evaluación de la función de fitness. Después cada proceso regresa su subpoblación al nodo maestro, se guarda el mejor individuo y se reemplaza la población, se realiza el mismo procedimiento hasta un cierto número de generaciones. Se

determinó este proceso de distribución debido a que la evaluación de la factibilidad de los individuos consume más recursos computacionales. Este modelo permite mantener la secuencia del algoritmo original.

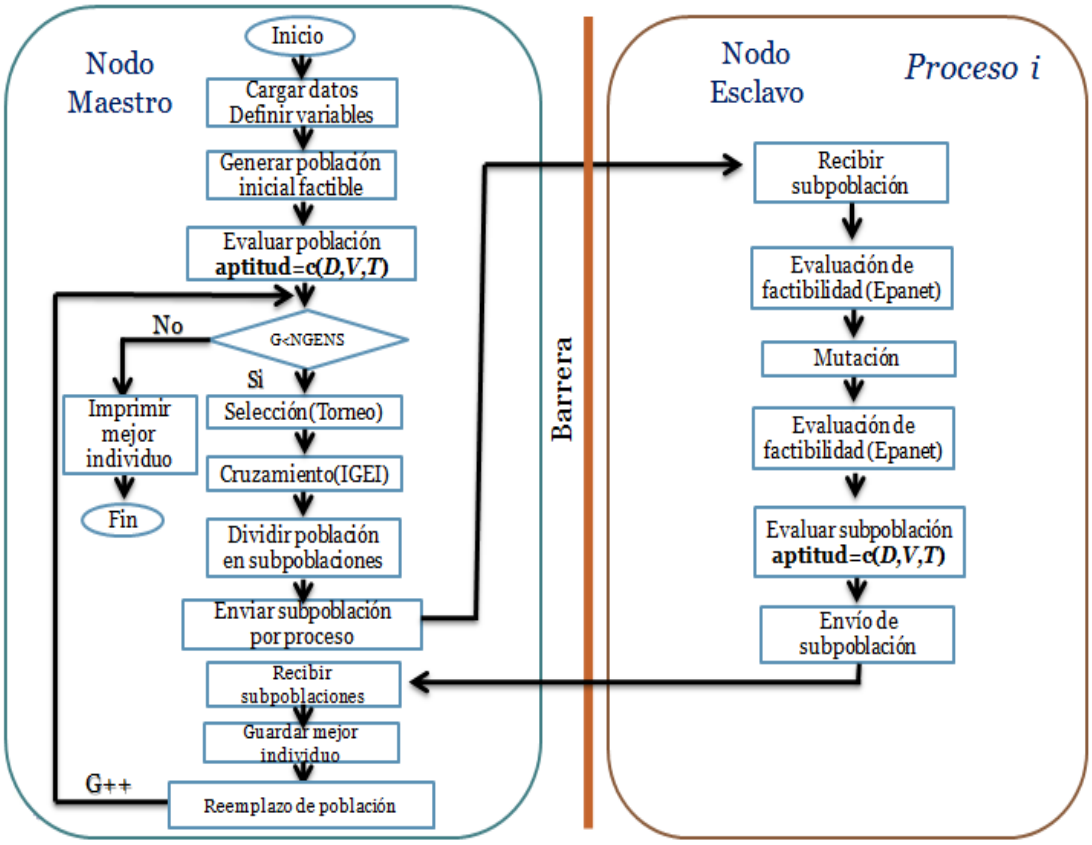

Figura 4-10 Modelo Maestro-Esclavo del algoritmo genético

El Algoritmo 4-10 muestra el pseudocódigo del algoritmo genético distribuido. En primer lugar, el algoritmo inicializa los parámetros de entrada, en la (línea 2 hasta línea 5) se inicializa el ambiente paralelo. Después se genera los tipos de datos de MPI para tanques, válvulas y tuberías (línea 6), se explica en el Apéndice D. Una vez obtenidos los datos de la instancia se carga y se genera la configuración inicial (línea 7 y 8). Por último el nodo Maestro genera la población inicial factible y evalúa la función de fitness (línea 9 hasta línea 12). Después se realizan una cierta cantidad de generaciones, donde el nodo Maestro se encarga de guardar el mejor individuo encontrado en cada generación, selecciona los mejores individuos

para posteriormente ser cruzados y obtener nuevos descendientes hasta a completar la población. El cruzamiento se realiza con la población total, esto para exista diversidad en las soluciones. Después la población se divide en subpoblaciones y así enviarlas a cada proceso mediante la función Scatter (línea 20). Cada proceso (Esclavo) recibe una subpoblación, evalúa la factibilidad (Epanet) de los individuos de su población y aplica el operador de mutación (línea 21 y 22). Posterior evalúa el valor de aptitud (ver Ec.(3-2)) de cada individuo (línea 23). Los procesadores esclavos envían sus datos al nodo maestro mediante la función Gather (línea 24). Después el nodo maestro reemplaza la nueva población e incrementa el número de generaciones (línea 25 hasta línea 28). El procedimiento se repite hasta alcanzar el número total de generaciones.

Algoritmo 4-10. Pseudocódigo del algoritmo genético distribuido

```
1:  Parámetros (rank, nprocs, MEJOR_IND, SubP, P, NINDS,G, N_g)//Inicializar los parámetros de entrada
2:  MPI_Init(&argc, &argv)// Inicial ambiente MPI
3:  MPI_Comm_rank(MPI_COMM_WORLD, &rank)//obtiene el id del procesador
4:  MPI_Comm_size(MPI_COMM_WORLD, &nprocs)//número de procesos
5:  MPI_Barrier(MPI_COMM_WORLD);
6:  Generar MPI_Datatype (tanques, válvulas, tuberías)//Se genera el tipo de dato MPI
7:  CargarDatos () //Carga los datos de la instancia
8:  ConfiguracionInicial () //Genera configuración inicial
9:  Si (rank==0) Entonces
10:     Poblacion_inicial(P)//Genera la población inicial
11:     EvaluacionIndividuo(P)//Se evalúa la población
12:Fin Si
13:Mientras G <N_g Hacer
14:    Si (rank==0)Entonces
15:       MejorIndividuo(MEJOR_IND, P) //Guarda el mejor individuo
16:       Seleccion(P)//Seleccionar el # de invidivuos
17:       Cruzamiento(P)//Cruzar los mejores individuos
18:       DividirPoblacion(P,SubP)//Divide la poblacion en subpoblaciones
19:    Fin Si
20:    MPI_Scatter(&SubP ,NINDST, individuo, &P, NINDST, individuo, MPI_COMM_WORLD)
21:       EvaluacionFactibilidad (P) //Genera factibilidad de los individuos
22:       Mutación (P)//Se realiza la mutación
23:       EvaluacionIndividuo(P,NINDS)// Se evalúa la función de fitness
24:    MPI_Gather(&P, NINDST, individuo, & SubP, NINDST, individuo,0,MPI_COMM_WORLD)
25:       Si(rank==0)Entonces
26:          ReemplazoPoblacion(P)// Se reemplaza la nueva población por la anterior
27:       Fin Si
28:    G=G+1//Incremento de número de generación
29: Fin Mientras
30: MPI_Finalize()
```

Capítulo 5

Resultados Experimentales

Capítulo 5 Resultados Experimentales

En este capítulo se presentan los resultados obtenidos en las pruebas experimentales realizadas al algoritmo genético. Para ello se realizó un análisis de sensibilidad con el objetivo de mejorar el desempeño del algoritmo. La sintonización de los parámetros de control se realizó en el clúster Texcal. Se presenta el análisis estadístico para el algoritmo genético secuencial y distribuido. Se muestra el análisis de eficiencia del algoritmo genético distribuido.

5.1 Descripción del Equipo utilizado

Para las pruebas se utilizó la infraestructura del clúster Texcal ubicado en Jiutepec Morelos (UPEMOR). Las características del clúster Texcal se presentan en la Tabla 5-1 y Tabla 5-2.

Tabla 5-1 Infraestructura Hardware del Clúster de producción TEXCAL

ELEMENTO	HARDWARE
Comunicaciones	Switch 3COM 24/10/100/1000 Switch Infiniband Mellanox de 18 puertos de 40 Gb/s QDR
Nodo maestro CPU Total 12 cores Total 24 GB RAM Total 12 TB HD	1 Motherboard: • 2 procesadores Intel Xeon Six Core a 3.06 GHz, 12 MB cache. • 2 HD Enterprise, 7200 RPM de 500GB (para S.O.). • 6 HD Enterprise, 7200 RPM, 12 TB en total. • 6 módulos RAM de 4GB 1333MHZ DDR3. Total de 24 GB RAM. 1 tarjeta Infiniband 40Gb/s

Nodos de procesamiento CPU: 01 al 04 Total 48 cores Total 96 GB RAM Total 2 TB HD	1 Motherboard: • 2 procesadores Intel Xeon Six Core a 3.06 GHz, 12 MB cache. • 1 HD Enterprise, 7200 RPM de 500GB. • 6 módulos RAM de 4GB 1333MHZ DDR3. Total de 24 GB RAM. 1 tarjeta Infiniband 40Gb/s
Nodo de procesamiento GPU: 05 Total 896 cores Total 36 GB RAM Total 1 TB HD	1 Motherboard: • 1 procesador Intel Xeon Six Core a 3.06 GHz, 12 MB cache. • 2 HD Enterprise, 7200 RPM de 500GB. • 9 módulos RAM de 4GB 1333MHZ DDR3. Total de 36 GB RAM. • 2 tarjetas NVIDIA TESLA C2070, arquitectura Fermi, con 6 GB RAM DDR5 c/u. 448 cores c/u. • DVD/RW • Lector de memoria 1 tarjeta Infiniband 40Gb/s

Tabla 5-2 Distribución de recursos del clúster Texcal

Nodo	**Núcleos**
Texcal	12
node01	12
node02	12
node03	12
node04	12

5.2 Análisis de Sensibilidad

Para la sintonización de los parámetros de control de un algoritmo, es necesario llevar a cabo un análisis de sensibilidad. El análisis de sensibilidad es un componente importante en la construcción de modelos matemáticos,

computacionales y de simulación. La finalidad del análisis de sensibilidad, es encontrar la mejor sintonización de los parámetros de control del algoritmo de manera que este tenga el mejor funcionamiento posible en cuanto a la eficiencia y eficacia.

1. Selección de los parámetros de control

En este paso hay que seleccionar los parámetros de control de acuerdo al algoritmo, para ello es necesario realizar una revisión bibliográfica de los autores que hayan implementado dicho algoritmo y así determinar los parámetros adecuados:

- Porcentaje de selección (P_s)
- Porcentaje de mutación (P_m)
- Tamaño de población (T_{pob})
- Probabilidad de cruzamiento (P_c)
- Probabilidad de mutación (P_{mut})
- Número de generaciones (N_g)

2. Establecer rangos de evaluación

Después de haber identificado los parámetros de control, es necesario determinar los rangos de cada parámetro, lo cual permitirá realizar el análisis de sensibilidad de dicho algoritmo, en la Tabla 5-3 se presentan los rangos de cada parámetro de control.

Tabla 5-3 Rangos de los parámetros de control

Parámetro de control	Límite inferior	Límite superior
P_s	10%	70%
P_m	10%	70%
T_{pob}	100	1000
P_c	0.7	
P_{mut}	0.3	
N_g	100	

3. Pruebas para rangos de evaluación

Una vez establecidos los rangos para cada parámetro de control, se realizan las pruebas experimentales, ejecutando 30 pruebas por cada una de las muestras calculadas. Para la adecuada sintonización de los parámetros de control, es necesario realizar las pruebas de los valores correspondientes a una de las variables, pero manteniendo fijos los demás, hasta encontrar el valor que mejore la calidad de las soluciones. En cuanto se obtenga el mejor valor de dicha variable, se fija el valor y se comienza con la variación de otra variable, llevando a cabo el mismo proceso, hasta obtener el conjunto de valores que permita obtener el mejor desempeño del algoritmo en cuanto a eficacia y eficiencia.

En la Tabla 5-4 se muestra los resultados obtenidos de la sintonización de parámetros de control del algoritmo genético secuencial. La metodología utilizada se basa en determinar un porcentaje de individuos seleccionados del 10% al 70%, dejando fijo el porcentaje de mutación a un 10%. Se realizaron 30 ejecuciones con una población de 100 individuos y 100

generaciones. Se presenta el mejor resultado, peor y promedio de un total de 30 ejecuciones. De acuerdo a estos se determina el porcentaje de selección a un 40% debido a que se encuentra el mejor promedio de 283,533 (unidades monetarias) y el menor costo a 244,500 (unidades monetarias). El costo de la función objetivo corresponde al costo que implica agregar nuevos tanques, tuberías y válvulas al sistema. Para este trabajo se utilizaron costos de los componentes con base a un promedio de cotizaciones hechas en 2012 (véase Apéndice B).

El comportamiento de las soluciones de acuerdo al peor, promedio y mejor fitness correspondiente al porcentaje de selección se muestra en la Figura 5-1; se observa que el peor el costo se presenta en un 20%, pero esto va mejorando a través del número de generaciones hasta encontrar el mejor fitness

Tabla 5-4 Resultados del porcentaje de selección

	$C(P,T,V)$		
Selección (%)	Mejor	Promedio	Peor
10	274800	305038	343000
20	251800	292977	344400
30	259500	287607	308600
40	**244500**	**283533**	**327100**
50	257900	286424	308300
60	259500	287504	323000
70	259800	285158	329400

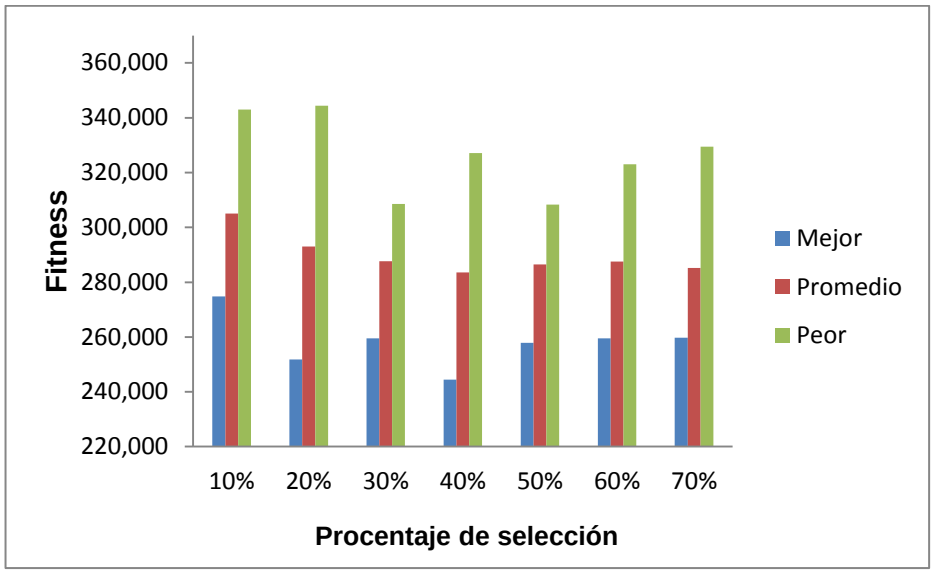

Figura 5-1 Comportamiento del mejor costo, peor y promedio con respeto al porcentaje de selección

La Tabla 5-5 muestra los resultados obtenidos del porcentaje de mutación del 10% hasta el 50%, dejando fijo el porcentaje de selección a un 40% debido a los resultados presentados en la Tabla 5-4. Se realizaron 30 ejecuciones con una población de 100 individuos y 100 generaciones. Se presenta el mejor, peor y promedio de un total de 30 ejecuciones. De acuerdo a los resultados se selecciona el porcentaje de mutación a un 30% debido a que se encuentra el mejor promedio de 274,328 y un costo mínimo de 244,800.

Las pruebas para el 60% y 70% no se realizaron debido a que el promedio empeora, así como el costo de la solución, además de que el tiempo se incrementa debido a que la evaluación de la factibilidad de los individuos se realiza con un porcentaje mayor. El comportamiento de las soluciones de acuerdo al peor, promedio y mejor fitness correspondiente al porcentaje de mutación se muestra en la Figura 5-2, se observa que el peor costo se presenta en un 50%, pero esto va mejorando a través del número de generaciones hasta encontrar el mejor fitness.

Tabla 5-5 Resultados del porcentaje de mutación

	$C(P,T,V)$		
Mutación (%)	Mejor	Promedio	Peor
10	244500	283533	327100
20	259800	280509	308300
30	**244800**	**274328**	**304800**
40	277475	299336	328042
50	278300	306170	330629

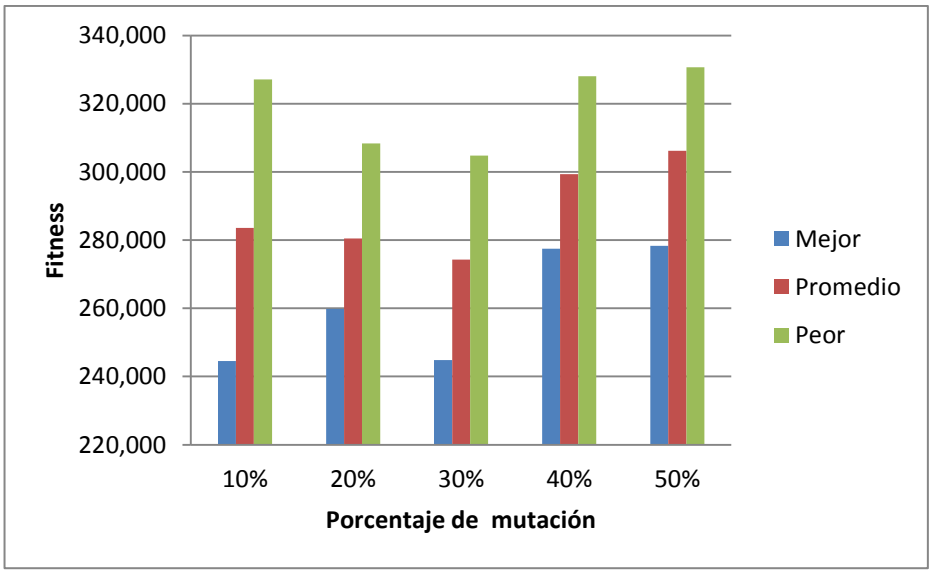

Figura 5-2 Comportamiento de las soluciones con respeto al porcentaje de mutación

La Tabla 5-6 muestra los resultados obtenidos del tamaño de población, dejando fijo el porcentaje de selección a un 40%, el porcentaje de mutación a un 30% y 100 generaciones. Se realizaron 30 ejecuciones con un tamaño de población de 100, 200,500, 800, 900 y 1000 individuos. La columna 2 muestra la mejor solución encontrada. En la columna 3 presenta el promedio y en la columna 4 la peor de las soluciones de las 30

ejecuciones. De acuerdo al comportamiento de las soluciones se observa que el mejor promedio es de 273,810 y menor costo encontrado es de 244,500, los cuales se obtuvieron con un tamaño de población de 1000 individuos (Figura 5-3).

Tabla 5-6 Resultados del tamaño de la población

	$C(P,T,V)$		
Número Individuos	Mejor	Promedio	Peor
100	254800	274328	304800
200	277120	295858	318754
500	263300	289500	302900
800	263700	287295	298075
900	263300	280893	297100
1000	**244500**	**273810**	**293300**

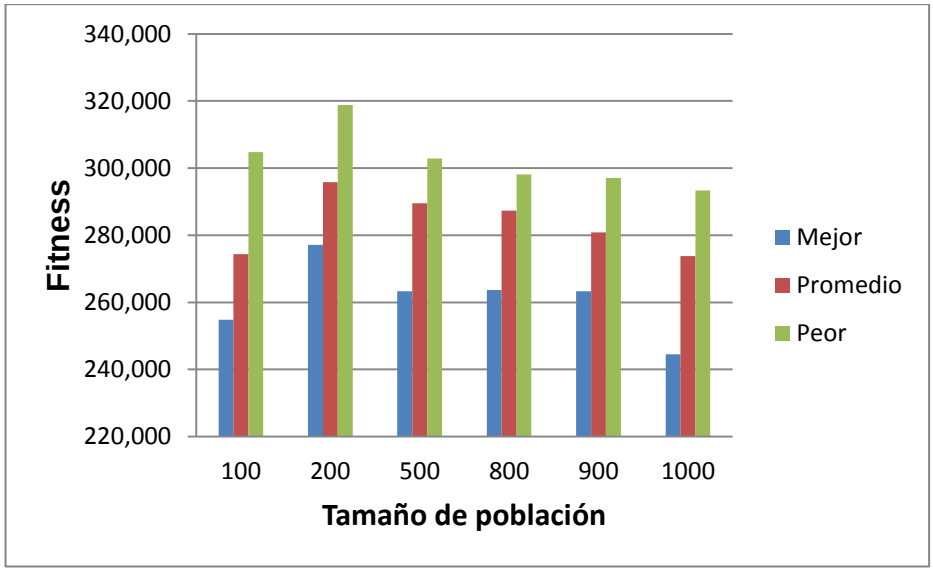

Figura 5-3 Comportamiento de las soluciones de acuerdo al tamaño de población

4. Sintonización de Parámetros

Una vez terminadas las pruebas experimentales para obtener los valores de cada uno de los parámetros de control definidos para todas las muestras, estos valores son definidos como los valores de sintonización, el valor de la probabilidad de cruzamiento se determinó de acuerdo a la literatura (ver Tabla 5-7).

Tabla 5-7 Valores de sintonización para los parámetros de control

Parámetro de control	Valor sintonizado
P_s	40%
P_m	30%
T_{pob}	1000
P_c	0.7
P_{mut}	0.3
N_g	100

La sintonización de parámetros se lleva a cabo con el objetivo de identificar los valores correspondientes a los parámetros de control, los cuales permitan obtener una mejora en el desempeño del algoritmo tanto en eficacia como en eficiencia.

5.3 Algoritmo Genético secuencial AGS-FRM

Después del análisis de sensibilidad, se realizaron las pruebas finales para obtener los resultados del algoritmo genético secuencial AGS-FRM para la red del FRM. Se muestra la convergencia del algoritmo. También se presenta el análisis estadístico de los resultados y por último se define el cálculo de la complejidad del algoritmo.

5.3.1 Convergencia del algoritmo genético

La convergencia del algoritmo, permite establecer en qué punto el algoritmo alcanza la estabilidad. En la Figura 5-4 se presenta el comportamiento del algoritmo genético secuencial con base al número de generaciones y al costo promedio de un total de 30 ejecuciones. Se utilizó un tamaño de población de 100 individuos, 2 mil generaciones, 30% de porcentaje de mutación y 40% porcentaje de selección. Se observa que aún no se alcanza un valor estable del costo promedio y por lo tanto tampoco la convergencia del algoritmo. Esto significa que el costo de solución aún sigue mejorando. A pesar de esto ya no se realizaron pruebas con un número mayor de generaciones, debido a que el tiempo promedio para 2 mil generaciones fue de 51 horas, por lo que se tendría una mayor inversión de tiempo.

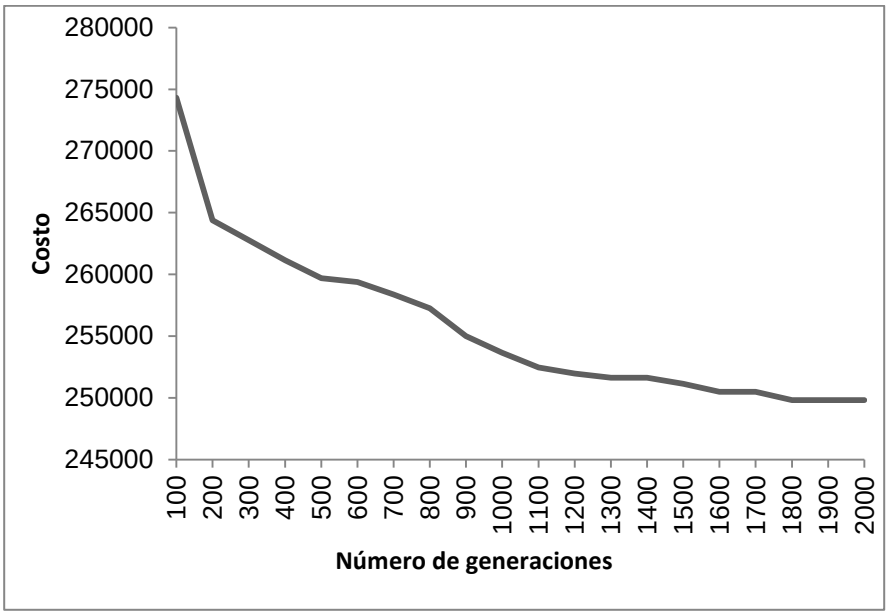

Figura 5-4 Comportamiento de la convergencia del algoritmo AGS-FRM

De acuerdo a los resultados presentados en la Figura 5-4 en ninguno de los porcentajes de selección la convergencia del algoritmo se alcanza, se realizaron pruebas con 2 mil generaciones pero el algoritmo sigue obteniendo buenas soluciones, pero a pesar de eso ya no se realizaron pruebas con un número mayor de generaciones debido al tiempo computación que se requiere.

En la Figura 5-5 se presenta los resultados obtenidos del algoritmo con diferentes tamaños de población. Las pruebas se realizaron para 200, 500, 800 y 900 individuos. Cada prueba con 2 mil generaciones, 40% de selección y 30% de mutación. También se observa que el costo promedio sigue mejorando conforme el número de generaciones.

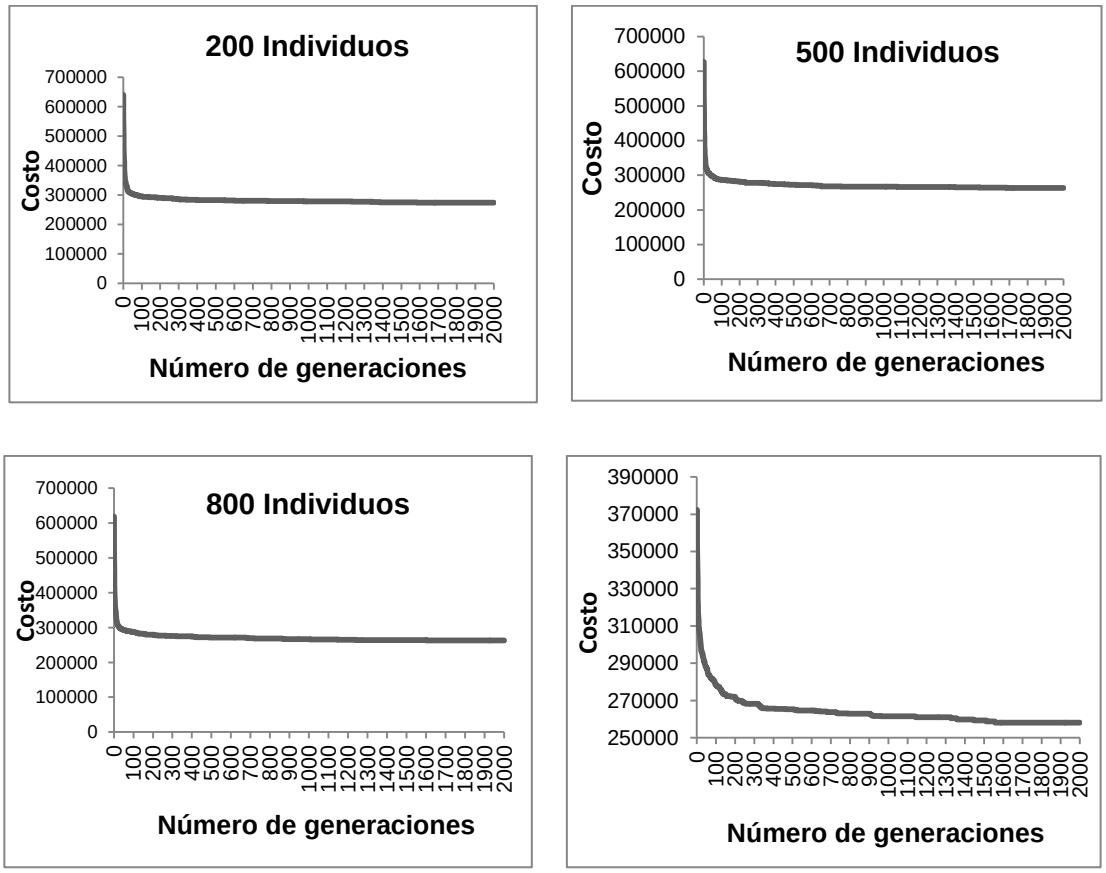

Figura 5-5 Comportamiento del costo promedio para el tamaño de la población

5.3.2 Análisis Estadístico

El análisis estadístico es elemental para evaluar el comportamiento del algoritmo con respecto a la instancia o instancias evaluadas y así determinar el desempeño del mismo. En este trabajo se evalúa una instancia de un caso real, por lo que se trabaja con una sola instancia. Después de realizar el análisis de sensibilidad y determinar los parámetros de control sintonizados, se realizaron 30 ejecuciones para la instancia con un tamaño de población de 1000 individuos, 40% de selección, 30% de mutación y 100 generaciones.

En la Tabla 5-8 se presenta la mejor solución encontrada con respecto a la función objetivo, la peor solución, el promedio de las 30 ejecuciones y su desviación estándar, la cual permite conocer que tan dispersas son las soluciones con respecto a la media. De acuerdo al resultado presentado para la moda, se observa que el costo se repite con mayor frecuencia para la distribución de datos. La desviación estándar es pequeña, por lo tanto la media es un valor confiable de la solución que el algoritmo produce. La mediana y la moda están dentro de $\pm\sigma$ lo que confirma la tendencia central de los datos.

Tabla 5-8 Resultados del costo de la red del FRM mediante el algoritmo genético secuencial

Medida	Costo (Unidades monetarias)
Peor	295,975
Promedio	281,618
Mejor	259,500
Media ± Desviación estándar	281,618 ±7,883
Mediana	279,987
Moda	278,300

De acuerdo al mejor costo obtenido de 259,500 y con las características adecuadas de la configuración de la red del FRM, se tendría que agregar 3 tanques de almacenamiento, 3 tuberías y 8 válvulas reductoras de presión. El resumen de los datos de la solución de la red completa se presenta en la Tabla 5-9.

Tabla 5-9 Resumen de los datos de la mejor solución encontrada y la solución actual de la red del FRM

Tuberías	Nodos	Tanques	Embalses	Válvulas	Demanda Estándar (L/S)	Presión mínima (mca[4])	Presión máxima (mca)	Diámetros tuberías (mm)
367	350	9	1	8	0.003	10	60	22.7 - 101.6

La localización óptima de las válvulas, la configuración adecuada de los tanques de almacenamiento y los diámetros necesarios para las tuberías que permiten mejorar la distribución de agua en la red del FRM se explica a continuación. De acuerdo a los símbolos presentados en la Tabla 3-1, se observa en la Figura 5-6 las diferentes tuberías donde se agrega una válvula reguladora de presión (PRV), esto con el objetivo de controlar la presión en todo el sistema. También se agrega un tanque en la sección 1, otro en la sección 2 y otro tanque de almacenamiento en la sección ensueño. En estas secciones es donde se presenta presiones menores a 10 mca a la configuración actual incluso negativas. De acuerdo a la escala de los colores de la presión para la configuración propuesta, se observa que no hay nodos con color azul marino, esto indica que todos los nodos cumplen con la presión mínima requerida. Tampoco se presentan nodos de color rojo, por lo que no se tienen presiones mayores a 60 mca que esa es una de las restricciones.

[4]mca: significa metros columna de agua

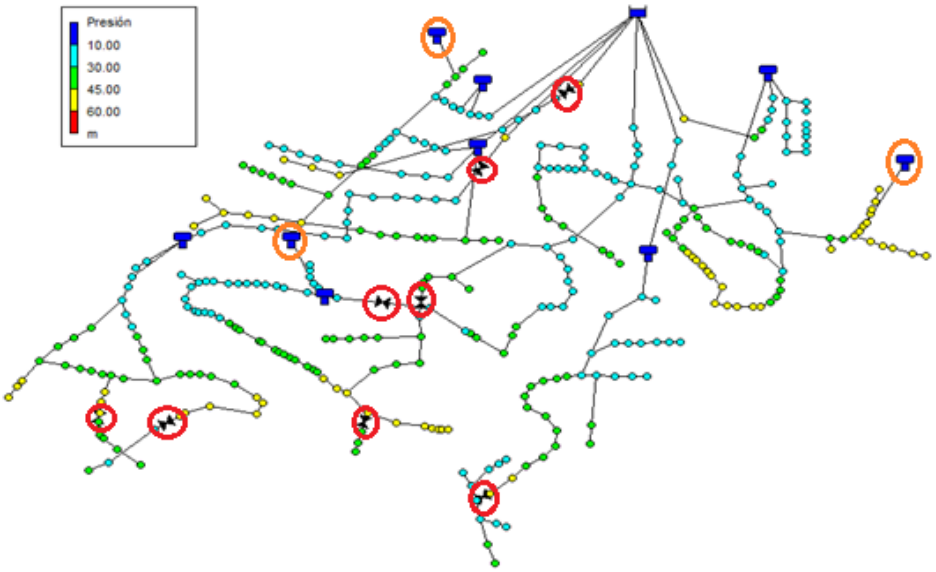

Figura 5-6 Simulación hidráulica de la mejor solución

Por lo tanto, ésta configuración es la adecuada para obtener el menor costo posible además de una solución factible, debido a que como ya se mencionó la presión en cada nodo cumple con el conjunto de restricciones.

En la Figura 5-7 se presenta la presión obtenida en cada nodo, se observa que las presiones son adecuadas y balanceadas. La distribución de agua es equivalente para toda la red.

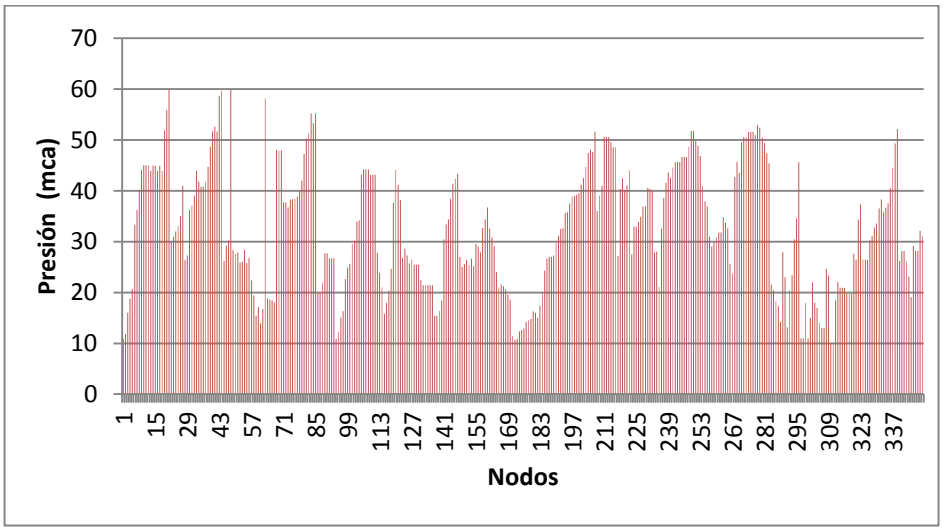

Figura 5-7 Presiones en cada nodo de la mejor solución

De la Tabla 5-10 a la Tabla 5-12 se presentan las características de los elementos agregados a la red del FRM. Estos datos son los adecuados para la mejor solución encontrada por el algoritmo AGS-FRM, la cual logró un costo de 259,500. Se agregaron 3 tanques con diferentes características: por ejemplo el tanque 1006 tiene un diámetro de 14.09 m, el nivel máximo de 13 m y el nivel inicial de la simulación de 6.50 m. En cuanto a las nuevas tuberías solo se agregaron 3, las cuales conectan a cada uno de los tanques con el resto de la red, por ejemplo la tubería 366 conecta al tanque 1007 con el nodo 279, la longitud de la tubería es de 260 m con un diámetro de 76.00 mm. Con respecto a las válvulas reductoras de presión (PRVs), la colocación óptima para esta solución es de 8 válvulas cada una con diferentes parámetros, por ejemplo la válvula 4021 conecta al nodo 44 con el nodo 45, el diámetro es de 38.10 mm y la presión de tarado es de 26.22 m. La válvula 4107 conecta al nodo 222 con el nodo 223 con un diámetro de 76.20 mm y la presión de tarado es de 27.49 m.

Tabla 5-10 Datos obtenidos de los nuevos tanques agregados a la red FRM

ID_Tanque	Altura (m)	Nivel Inicial (m)	Nivel Mínimo (m)	Nivel Máximo (m)	Diámetro (m)
1006	2412	6.50	0.00	13.00	14.09
1007	2425	4.00	0.00	8.00	7.90
1008	2329	4.50	1.00	9.00	12.66

Tabla 5-11 Datos obtenidos de las nuevas tuberías agregadas en la red

ID_Tubería	Nodo Origen	Nodo Destino	Longitud (m)	Diámetro (mm)
365	1006	120	250.00	101.00
366	1007	279	260.00	76.00
367	1008	168	180.00	50.00

Tabla 5-12 Características de las válvulas (PRV) implementas a la red FRM

ID_Válvula	Nodo Origen	Nodo Destino	Diámetro (mm)	Presión de tarado
4010	21	22	38.10	30.00
4021	44	45	38.10	26.22
4023	48	49	50.80	28.28
4030	63	64	19.05	18.84
4079	164	165	50.80	27.86
4100	207	208	50.80	36.02
4107	222	223	76.20	27.49
4161	339	340	31.70	26.22

5.3.3 Análisis de la complejidad el algoritmo genético

Para obtener la complejidad de un algoritmo, es necesario realizar un estudio para conocer su comportamiento, medir su rendimiento y el uso de los recursos.

Un algoritmo se enfoca en dos recursos importantes para resolver un problema, *tiempo* y *espacio*. El tiempo de un algoritmo es el número máximo $f(n)$ de pasos necesarios para resolver un problema de tamaño n, siendo n cualquier número natural arbitrario pero fijo. La complejidad se define generalmente en términos del análisis del peor caso (Salazar, 2001; Talbi, 2009).

La función temporal del algoritmo propuesto se presenta en ecuación (5-1):

$$\boldsymbol{f(g,n) = 32g + \frac{145}{2}gn + \frac{56}{3}gmn + \frac{82}{3}gxn + 4gk + \frac{347}{3}gtn + \frac{416}{3}gn^2 + \frac{52}{3}gtn^2 + \frac{336}{3}gvn^2 + \frac{30}{3}gxn^2 + \frac{26}{3}gvn + 17gx + 2gm + 10gmx + 26gv + 2} \quad \textbf{(5-1)}$$

Donde g es el número de generaciones, n es el número de individuos de la población, x es el número de válvulas, m es el número de tuberías nuevas, t es el número de tanques, v es el número de vértices y k es el número de diámetros comerciales utilizados en la red del FRM.

La complejidad asintótica para el problema es:

$$O(gn^2)$$

5.4 Algoritmo Genético distribuido AGD-FRM

Para la distribución de los procesos del algoritmo AGD-FRM se utiliza el modelo Maestro-Esclavo con comunicación colectiva a través de la librería MPI. La distribución de procesos se realiza en el clúster Texcal y se utilizaron 4 nodos (node01, node02, node03 y node04), cada nodo cuenta con 12 núcleos de procesamiento.

Para el análisis estadístico del algoritmo genético distribuido AGD-FRM se distribuye las tareas con n número de procesos. En este trabajo se utiliza el modelo Maestro-Esclavo, debido a que después de realizar un profiling, se analizó que el módulo que más tiempo computacional requiere es la evaluación de la factibilidad de la población, por lo que fue la principal razón por la cual se determinó este proceso de distribución. El nodo Maestro se encarga de dividir la población total entre subpoblaciones. La cantidad de subpoblaciones es determinada por el número de procesos, por lo que también se calcula el número de individuos que formaran parte de

cada subpoblación, por ejemplo si se tiene un tamaño de población de 960 individuos, la cual es divida entre en número de procesos, es decir si se distribuye en 4 procesos, cada subpoblación tendría 240 individuos (ver Tabla 5-13). Es importante mencionar que el número de subpoblaciones debe ser igual al número de procesos.

Tabla 5-13 Tamaño de las subpoblaciones de proceso por nodo

Número de procesos	Total de nodos	Procesos por nodo	Individuos por Subpoblación
4	4	1	240
8	4	2	120
16	4	4	60
32	4	8	30

5.4.1 Análisis estadístico

En la Tabla 5-14 se presenta los resultados de la mejor, peor y el promedio de 30 ejecuciones para cada número de procesos. Se observa que el mejor costo al igual que el mejor promedio encontrado fue con una distribución en 4 procesos. La quinta columna etiquetada representa el cálculo de la desviación estándar, permite conocer que tan dispersas son el resto de las soluciones con respecto a la media.

Tabla 5-14 Resultados del algoritmo genético distribuido

Número de procesos	Mejor	Promedio	Peor	Media ± σ	Mediana
4	244,500	253,631	301,324	253,631 ±16,780	244,800
8	277,475	297,009	306,726	297,009 ±11,697	301,324
16	279,075	295,783	318,331	295,783 ±16,663	292,863
32	284,400	300,550	319,911	300,550 ±11,516	300,037

En la Figura 5-8, se muestra la red del FRM con los nuevos componentes, los cuales permitieron mejorar la distribución de agua en la red. Considerando la localización óptima de las válvulas, la configuración adecuada de los tanques de almacenamiento y los diámetros necesarios para las tuberías. De acuerdo a los símbolos presentados en la Tabla 3-1, se observa que para ésta solución se agrega en diferentes tuberías una válvula, esto con el objetivo de unificar la presión en todo el sistema. Se agrega un tanque a la sección 1 y otro a la sección 2. También se agrega un nuevo tanque a la sección ensueño. Estas secciones presentaban presiones menores a 10 mca incluso negativas. De acuerdo a la escala de colores de la presión que arroja Epanet, se observa que ya no existen nodos con color azul marino, esto indica que todos los nodos cumplen con la presión mínima requerida. De igual manera no se presentan nodos de color rojo, por lo que las presiones están dentro del rango de la presión máxima requerida.

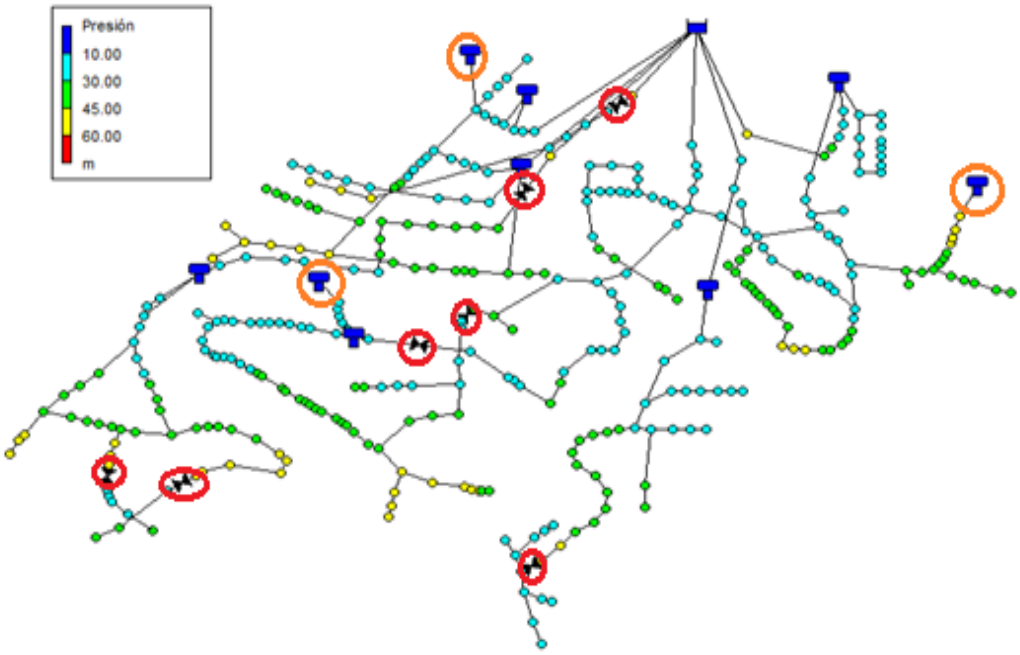

Figura 5-8 Simulación hidráulica de la mejor solución la red del FRM con 4 procesos de distribución

Ésta configuración logra un comportamiento adecuado. Las presiones están dentro de los rangos de las restricciones por lo que no es necesaria ya ninguna modificación. En la Figura 5-9 se presenta la presión obtenida en cada nodo, como se observa las presiones están regularizadas, ya no se tienen presiones negativas, ni presiones mayores a 60 mca. La distribución de agua es más uniforme para toda la red.

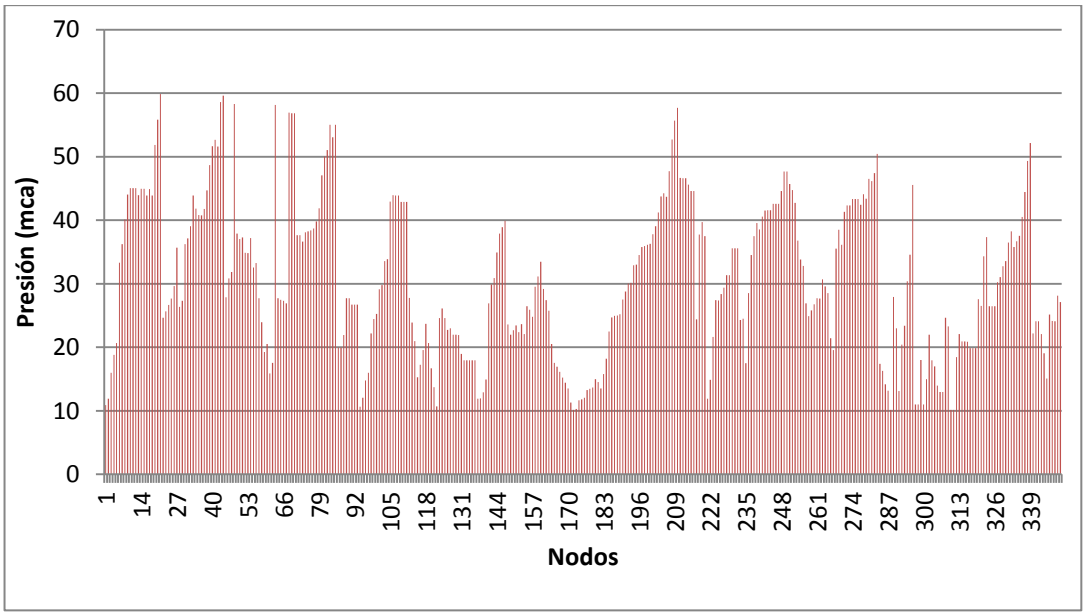

Figura 5-9 Presiones de los nodos para la solución obtenida con un costo de 244,500

De la Tabla 5-15 a la Tabla 5-17 se presentan las características de los elementos agregados a la red del FRM. Estos datos son los adecuados para la mejor solución encontrada por el algoritmo AGD-FRM, esta solución alcanzó un costo de 244,500. Se agregaron 3 tanques con diferentes características, por ejemplo el tanque 1007 tiene un diámetro de 5 m, un nivel máximo de agua de 12 m y nivel inicial de 6 m. En cuanto a las nuevas tuberías se agregaron 3, las cuales conectan a cada uno de los tanques con el resto de la red, por ejemplo la tubería 367 conecta al tanque 1008 con el nodo 170, la longitud de la tubería es de 250 m con un diámetro de 31.00 mm. Con respecto a las válvulas reductoras de presión (PRVs), la colocación

óptima para ésta solución es de 7 válvulas cada una con diferentes parámetros, por ejemplo la válvula 4106 conecta al nodo 220 con el nodo 221, el diámetro es de 76.20 mm y la presión de tarado es de 11.92 m. La válvula 4010 conecta al nodo 21 con el nodo 22 el diámetro es de 38.10 mm y la presión de tarado es de 24.67 m. La configuración adecuada y la localización óptima de los elementos contribuyen a minimizar el costo de la solución.

Tabla 5-15 Datos de los nuevos tanques incorporados a la red del FRM

ID_Tanque	Altura (m)	Nivel Inicial (m)	Nivel Mínimo (m)	Nivel Máximo (m)	Diámetro (m)
1006	2385	6.00	0.00	12.00	5.00
1007	2428	6.50	0.00	12.00	5.00
1008	2324	6.50	1.00	13.00	5.00

Tabla 5-16 Datos de las nuevas tuberías de la red del FRM

ID_Tubería	Nodo Origen	Nodo Destino	Longitud	Diámetro
365	1006	118	200.00	101.00
366	1007	283	240.00	76.00
367	1008	170	250.00	31.00

Tabla 5-17 Datos de las válvulas agregadas a la red del FRM

# válvula	ID_Válvula	Nodo Origen	Nodo Destino	Diámetro	Presión de tarado
1	4010	21	22	38.10	24.67
2	4021	44	45	38.10	27.85
3	4023	48	49	50.80	37.90
4	4030	63	64	19.05	27.74
5	4079	164	165	50.80	34.17
6	4106	220	221	76.20	11.92
7	4161	339	340	31.70	22.19

El número de cambios a la red del FRM original se basa en incrementar a 3 tanques de almacenamiento, 7 válvulas reguladoras de presión y 3 tuberías con 690 metros de longitud. Considerando que

inicialmente se tenían 6 tanques de almacenamiento, cero válvulas y 192,624 metros de longitud de tubería, se considera que el número de cambios o incrementos en el sistema original fue mínimo y por lo tanto el costo de estos cambios, resulta ser mínimo.

5.4.2 Análisis de eficiencia del algoritmo

El análisis de la eficiencia de un algoritmo es muy importante, porque permite evaluar el comportamiento del mismo con respecto al tiempo de ejecución que conlleva encontrar buenas soluciones para un cierto tipo de problema. En la Figura 5-10 se muestra el tiempo menor, mayor y promedio de 30 ejecuciones para un número de procesos de 4, 8, 16 y 32. De acuerdo a los resultados se observa que conforme se aumenta el número de procesos, el tiempo mejora considerablemente. Por lo que, si se continuará haciendo pruebas con un número mayor de procesos, la eficiencia del algoritmo mejoraría aún más, pero ya no se realizaron debido a que la eficacia no mejora conforme se aumenta el número de procesos (ver Tabla 5-14).

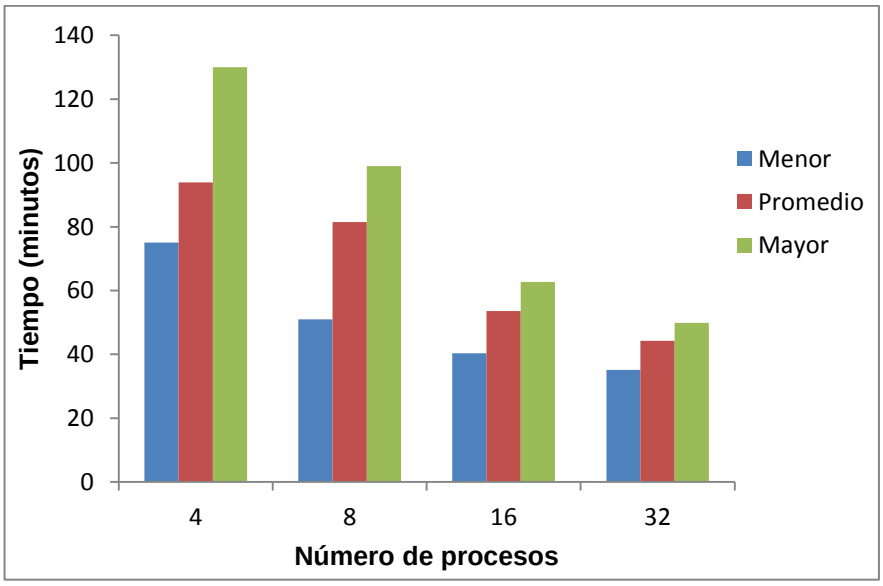

Figura 5-10 Tiempo de ejecución del algoritmo AGD-FRM

El tiempo de ejecución de un algoritmo secuencial es el mismo en cualquier plataforma. Sin embargo, para un algoritmo paralelo el tiempo de ejecución depende del tamaño de problema, del número de procesadores y de los parámetros de comunicación entre los procesos.

La finalidad del cómputo paralelo es disminuir el tiempo de ejecución de un algoritmo. Para evaluar el rendimiento de un algoritmo distribuido se utilizan diferentes métricas: la métrica más empleada es el *Speedup*, el cual permite comparar la ganancia de velocidad del algoritmo distribuido con respecto al algoritmo secuencial. El *Speedup* se calcula con la siguiente fórmula:

$$Speedup = \frac{Tiempo\ de\ ejecución\ del\ algoritmo\ secuencial}{Tiempo\ de\ ejecución\ del\ algoritmo\ paralelo}$$

La eficiencia se muestra en comparar que tanto se acerca la medida de aceleración del algoritmo con respecto al speedup ideal. La métrica de Speedup se clasifica en los siguientes puntos (Martínez-Oropeza, 2015):

- Speedup lineal: es el speedup ideal
- Speedup Sublineal: es una aceleración menor al speedup ideal
- Speedup Superlineal: es una aceleración mayor al speedup ideal

La aceleración del algoritmo muestra un comportamiento de un Speedup muy por abajo del ideal. Se calculó el speedup del menor tiempo, mayor y promedio. Sin embargo para los tres casos está muy por abajo del ideal, esto es debido a la comunicación entre los nodos del clúster para la trasmisión de datos. Para este trabajo se utilizó para la conectividad entre los nodos del clúster un cable par trenzado de categoría 6, el cual tiene una velocidad de transmisión de 1000Mbps, debido a que el ancho de banda de infiniband con una velocidad de trasmisión de 40Gbps no estaba disponible. Esto hace que se la latencia se eleve en el envío de información, lo cual se ve reflejado en el cálculo del speedup (Figura 5-11 a la Figura 5-13).

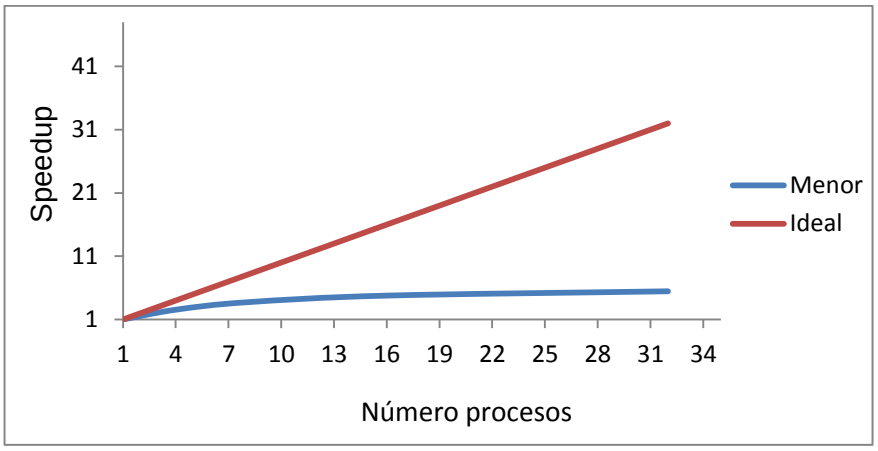

Figura 5-11 Comportamiento de speedup para un tiempo menor del algoritmo

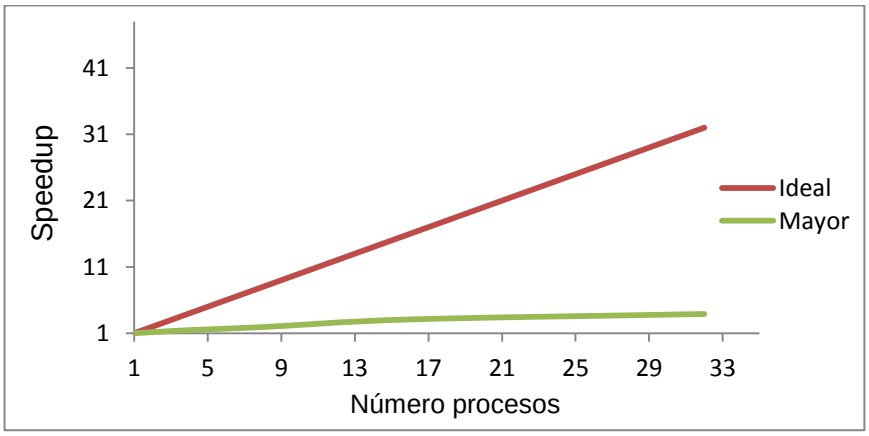

Figura 5-12 Comportamiento de speedup para un tiempo mayor del algoritmo

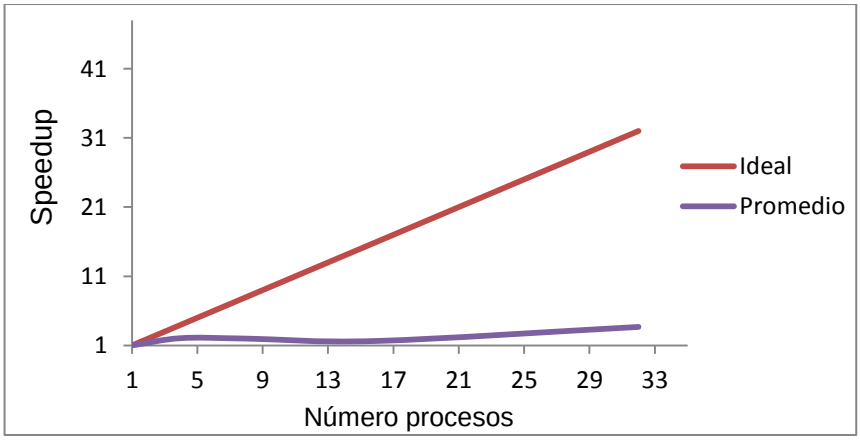

Figura 5-13 Comportamiento de speedup para un tiempo promedio del algoritmo

Capítulo 6

Conclusiones y Trabajo Futuro

Capítulo 6 Conclusiones y Trabajo Futuro

6.1 Conclusiones

La metodología de solución implementada presenta buenas soluciones para el problema de distribución de agua del Fraccionamiento Real Montecasino. Ésta permitió obtener soluciones factibles, al cumplir con el modelo de satisfacción de restricciones y permitió encontrar un costo mínimo de los cambios realizados en la red para hacer ésta operable, al involucrar un modelo de optimización.

Las deficiencias del diseño que actualmente tiene la red del FRM, hacen que aproximadamente un 30% de los usuarios del Fraccionamiento Real de Montecasino carezcan del abasto de agua. El algoritmo genético propuesto en este trabajo de investigación permitió junto con Epanet, presentar una propuesta de solución factible con cambios mínimos a la red del FRM y con un costo mínimo, cumpliendo con las necesidades al 100% de los usuarios del Fraccionamiento Real Montecasino.

El análisis estadístico de los resultados obtenidos por el algoritmo genético AGS-FRM arroja una solución promedio de 294,056 unidades monetarias. El valor de la desviación estándar es pequeño (del orden de unidades de millar), por lo que el valor del promedio es confiable y representativo de la solución producida por el algoritmo.

La implementación del algoritmo genético distribuido AGD-FRM en un clúster computacional permite mejorar la eficiencia del algoritmo aproximadamente un 75% con respecto al algoritmo genético secuencial. A pesar de esta mejoría, se puede entender que el Speedup no dio un buen valor en la eficiencia del algoritmo debido al ancho de banda que se utilizó para realizar las pruebas. En este caso se utilizó para la conectividad entre

nodos un cable de par trenzado de categoría 6, el cual tiene un ancho de banda de 250 MHz para transmitir 1000Mbps. Esto hace que se eleve la latencia en el envío de información, lo cual se ve reflejado en el cálculo del speedup.

6.2 Trabajo Futuro

La propuesta de solución desarrollada durante este trabajo de investigación, servirá como base para las siguientes actividades a realizar como trabajo futuro:

- Implementar la metodología de solución y el algoritmo propuesto para otros estudios de caso real.
- Hacer uso de la infraestructura Grid Morelos, utilizando varios clusters para la distribución del algoritmo genético, mediante la implementación del modelo de islas, el cual permite mejorar la calidad de las soluciones.
- Realizar pruebas experimentales haciendo uso del ancho de banda Infiniband con una velocidad de trasmisión de 40Gbps.
- Resolver la red del FRM mediante otra metodología, como es el redireccionamiento de los diámetros de las tuberías y bombeo.
- Crear un módulo que enlace el algoritmo genético con una aplicación en Web. Esto con la finalidad de hacer amigable la metodología propuesta y tenerla como una aplicación académica para presentación a los municipios del estado y del país en los que exista problemas de abastecimiento en las redes hidráulicas para ofrecer una herramienta que pueda dar solución a este tipo de problemas que se presentan comúnmente en la sociedad.

Referencias

Alba, E., Luque, G., & Nesmachnow, S. (2013). Parallel metaheuristics: recent advances and new trends. *International Transactions in Operational Research*, *20*(1), 1–48. http://doi.org/10.1111/j.1475-3995.2012.00862.x

Ali, M. E. (2015). Knowledge-Based Optimization Model for Control Valve Locations in Water Distribution Networks. *Journal of Water Resources Planning and Management*, *141*(1), 1–7. http://doi.org/10.1061/(ASCE)WR.1943-5452.0000438.

Alperovits, E., & Shamir, U. (1977). Design of optimal water distribution systems. *Water Resources Research*, *13*(6), 885–900. http://doi.org/10.1029/WR013i006p00885

Amor, B. H. (2008). *Intelligent Exploration for Genetic Algoritms Using Self-Organizing Maps in Evolutionary Computation*. VDM Verlag Dr. Mullen.

Araujo, L. S., Ramos, H., & Coelho, S. T. (2006). Pressure Control for Leakage Minimisation in Water Distribution Systems Management. *Water Resources Management*, *20*(1), 133–149. http://doi.org/10.1007/s11269-006-4635-3

Arunkumar, M., & V.E., N. M. (2011). Water Demand Analysis of Municipal Water Supply Using Epanet Software. *International Journal on Applied Bioengineering*, *5*(1), 9–19. Retrieved from http://journals-sathyabama.com/archives/iabe_abstract.php?id=53

Ávila-Melgar, E. Y. (2015). *Algoritmo Evolutivo en Ambiente GRID para el Problema de Diseño de Redes de Distribución de Agua*. Tesis Doctoral. Universidad Autónoma del Estado de Morelos. Retrieved from http://148.218.108.136:8080/pdf/TesisErikaA_2015.pdf

Ávila-Melgar, E. Y., Cruz-Chávez, M. A., & Martínez-Bahena, B. (2016). General Methodology for Using Epanet as an Optimization Element in

Evolutionary Algorithms in a Grid Computing Environment to Water Distribution Network Design. *Journal Water Science and Technology: Water Supply*.(Enviado)

Baños, R., Gil, C., Reca, J., & Montoya, F. G. (2010). A memetic algorithm applied to the design of water distribution networks. *Applied Soft Computing*, *10*(1), 261–266. http://doi.org/10.1016/j.asoc.2009.07.010

Barney, B. (2015). Message Passing Interface (MPI). Retrieved January 1, 2016, from https://computing.llnl.gov/tutorials/mpi/

Bhave, P. R. (1991). *Analysis of Flow in Water Distribution Networks* (Technomic). Lancaster, Pennsylvania.

Cattafi, M., Gavanelli, M., Nonato, M., Alvisi, S., & Franchini, M. (2011). Optimal placement of valves in a water distribution network with CLP(FD). *Theory and Practice of Logic Programming*, *11*(4-5), 731–747. http://doi.org/10.1017/S1471068411000275

Cerny, V. (1985). Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. *Journal of Optimization Theory and Applications*, *45*(1), 41–51. http://doi.org/10.1007/BF00940812

Chang, J., Bai, T., Huang, Q., & Yang, D. (2013). Optimization of Water Resources Utilization by PSO-GA. *Water Resources Management*, *27*(10), 3525–3540. http://doi.org/10.1007/s11269-013-0362-8

Cisty, M. (2010). Hybrid Genetic Algorithm and Linear Programming Method for Least-Cost Design of Water Distribution Systems. *Water Resources Management*, *24*(1), 1–24. http://doi.org/10.1007/s11269-009-9434-1

Comisión Nacional del agua (CNA). (2007). *Manual de Agua Potable, Alcantarillado y Saneamiento* (2007th ed.). Mexico.

Cook, W. J., Cunningham, W. H., Pulleyblank, W. R., Schrijver, A. (1997). *Combinatorial Optimization* (1ra ed.). John Wiley & Sons.

Cruz-Chávez, M. A., Ávila-Melgar, E. Y., Cruz-Rosales, M. H., Martínez-Bahena, B., Flores-Sánchez, G. (2014). Search Space Analysis for the

Combined Mathematical Model (Linear and Nonlinear) of the Water Distribution Network Design Problem. In L. R. et Al. (Ed.), *Artificial Intelligence and Soft Computing* (pp. 347–359). Springer International Publishing.

Cruz-Chávez, M.A., Zavala-Díaz, J. C., Mariano-Romero, C. E., Juárez-Pérez, F., Avila-Melgar, E. Y. (2009). Calendarización en Redes de Distribución de Agua. In *7to Congreso Internacional de Cómputo en Optimización y Software* (pp. 52–66). Retrieved from http://www.gridmorelos.uaem.mx/~mcruz/paper8.pdf

D'Ambrosio, C., Lodi, A., Wiese, S., & Bragalli, C. (2015). Mathematical programming techniques in water network optimization. *European Journal of Operational Research*, *243*(3), 774–788. http://doi.org/10.1016/j.ejor.2014.12.039

Dai, P. D., & Li, P. (2014). Optimal Localization of Pressure Reducing Valves in Water Distribution Systems by a Reformulation Approach. *Water Resources Management*, *28*(10), 3057–3074. http://doi.org/10.1007/s11269-014-0655-6

Darvini, G., & Soldini, L. (2015). Pressure control for WDS management. A case study. *Procedia Engineering*, *119*, 984–993. http://doi.org/10.1016/j.proeng.2015.08.989

Darwin, C. (1859). On the origins of species by means of natural selection. *London: Murray*, 247. http://doi.org/10.1126/science.146.3640.51-b

Den, B. M., Stützle, T., & Dorigo, M. (2001). Design of iterated local search algorithms. In *Proceedings of EvoWorkshops* (pp. 441–451). http://doi.org/10.1007/3-540-45365-2_46

EPANET. (2016). EPANET. Retrieved February 4, 2016, from http://www.epa.gov/water-research/epanet#toolkit

Ewald, G., Kurek, W., Brdys, M. A., & Member, S. (2008). Grid Implementation of a Parallel Multiobjective Genetic Algorithm for Optimized Allocation of Chlorination Stations in Drinking Water

Distribution Systems : Chojnice Case Study. *IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS*, *38*(4), 497–509. http://doi.org/Doi 10.1109/Tsmcc.2008.923864

Geem, Z. (2015). Multiobjective Optimization of Water Distribution Networks Using Fuzzy Theory and Harmony Search. *Water*, *7*(7), 3613–3625. http://doi.org/10.3390/w7073613

Gendreau, M. (2003). An Introduction to Tabu Search. *Handbook of Metaheuristics*, *57*(July), 37–54. http://doi.org/10.1007/0-306-48056-5

Gendreau, M., & Potvin, J. Y. (2005). Metaheuristics in combinatorial optimization. *Annals of Operations Research*, *140*(1), 189–213. http://doi.org/10.1007/s10479-005-3971-7

Giustolisi, O., Savic, D., & Kapelan, Z. (2008). Pressure-Driven Demand and Leakage Simulation for Water Distribution Networks. *Journal of Hydraulic Engineering*, *134*(5), 626–635. http://doi.org/10.1061/(ASCE)0733-9429(2008)134:5(626)

Glover, F. (1889). Tabu Search-Part I. *ORSA Journal on Computing*, *1*(3), 190–206. http://doi.org/10.1287/ijoc.1.3.190

Glover, F. (1989). Tabu Search - Part I. *ORSA Journal on Computing*, *1*(3), 190–206. http://doi.org/10.1287/ijoc.1.3.190

Glover, F. (1990). Tabu Search: A tutorial. *Interfaces*. http://doi.org/10.1287/inte.20.4.74

Glover, F. (1996). Tabu search and adaptive memory programming - advances, applications and challenges. *Interfaces in Computer Science and Operations Research*, 1–75. http://doi.org/10.1007/978-1-4615-4102-8_1

Glover, F., Laguna, M., & Marti, R. (2007). Principles of tabu search. *Approximation Algorithms and Metaheuristics*, *23*, 1–12. http://doi.org/10.1016/j.ejor.2004.08.004

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*. *Addison Wesley* (Vol. Addison-We).

http://doi.org/10.1007/s10589-009-9261-6

Gropp, W., Lusk, E., & Skjellum, A. (1999). *Using MPI: Portable Parallel Programming with the Message-passing Interface* (Second). Cambridge, Massachusetts: The MIT Press.

Gupta, I., Bassin, J. K., Gupta, A., & Khanna, P. (1993). Optimization of water distribution system. *Environmental Software*, *8*(2), 101–113. http://doi.org/10.1016/0266-9838(93)90020-I

Hansen, P., & Mladenovic, N. (2003). Variable neighborhood search. *Handbook of Metaheuristics*, *24*(11), 145–184. http://doi.org/10.1007/0-306-48056-5_6

Hansen, P., & Mladenović, N. (2001). Variable neighborhood search: Principles and applications. *European Journal of Operational Research*, *130*(3), 449–467. http://doi.org/10.1016/S0377-2217(00)00100-4

Hansen, P., & Mladenović, N. (2005). Variable neighborhood search. In *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques* (pp. 211–238). http://doi.org/10.1007/0-387-28356-0_8

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems*. Cambridge,MA: MIT Press.

Hoos, H. H., & Stützle, T. (2004). Stochastic Local Search: Foundations and Applications. *Stochastic Local Search Foundations and Applications*, *3777*, 126–169. Retrieved from http://dl.acm.org/citation.cfm?id=983505

Hui Zhang, Ting-Lin Huang, Wen-Jie, H. (2009). Application of Heuristic Genetic Algorithm for Optimal Layout of Flow Measurement Stations in Water Distribution Networks. *2009 Fifth International Conference on Natural Computation*, (4), 140–143. http://doi.org/10.1109/ICNC.2009.513

Hurtado-Guzmán, V. H. (2009). *Algoritmos Genéticos y Epanet 2.0 para la Localización Óptima de Válvulas Reductoras de Presion en Redes de Distribución de Agua Potable*. Universidad Nacional Autónoma de

México. Retrieved from http://www.ptolomeo.unam.mx:8080/xmlui/bitstream/handle/132.248.52.100/1153/Tesis.pdf?sequence=1

Iglesias, R. P. L., Mares, G. F. J., Jiménez, L. P. A., & García, P. R. (2003). GENERALIDADES SOBRE LAS REDES DE ABASTECIMIENTO. In *Ingeniería hidráulica en los abastecimientos de agua* (pp. 1–820). Valencia.

Juárez, P. F. (2013). *ALGORITMO GENÉTICO HÍBRIDO COOPERATIVO EN AMBIENTE GRID PARA TALLERES CON FLUJO FLEXIBLE*. Tesis Doctoral. Universidad Autónoma del Estado de Morelos.

Kang, D., & Lansey, K. (2010). Real-Time Optimal Valve Operation and Booster Disinfection for Water Quality in Water Distribution Systems. *Journal of Water Resources Planning and Management-Asce*, *136*(4), 463–473. http://doi.org/10.1061/(ASCE)WR.1943-5452.0000056

Kessler, A., & Shamir, U. (1989). Analysis of the linear programming gradient method for optimal design of water supply networks. *Water Resources Research*, *25*(7), 1469–1480. http://doi.org/10.1029/WR025i007p01469

Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by Simulated Annealing. *Science*, *220*(4598), 671–680. http://doi.org/10.1126/science.220.4598.671

Korte, B., & Vygen, J. (2008). Combinatorial Optimization Theory and Algorithms. In L. J. R. L. Graham, B. B. Korte, B. L. Lovász, P. A. Wigderson, & B. G. M. Ziegler (Eds.), *Algorithms and Combinatorics* (Fourth). Retrieved from http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Algorithms+and+Combinatorics+24#9

Kurek, W., & Ostfeld, A. (2013). Multi-objective optimization of water quality, pumps operation, and storage sizing of water distribution systems. *Journal of Environmental Management*, *115*, 189–197. http://doi.org/10.1016/j.jenvman.2012.11.030

Lei, T., Guang, H., & Junfei, Q. (2011). Design of Water Distribution Network via Ant Colony Optimization. In *The 2nd International Conference on Intelligent Contro and Information Processing* (pp. 366–370). IEEE Computer Society.

Liberatore, S., & Sechi, G. M. (2009). Location and calibration of valves in water distribution networks using a scatter-search meta-heuristic approach. *Water Resources Management*, *23*(8), 1479–1495. http://doi.org/10.1007/s11269-008-9337-6

Liu, Y., Duan, H., & Feng, X. (2008). The Design of Water-reusing Network with a Hybrid Structure Through Mathematical Programming. *Chinese Journal of Chemical Engineering*, *16*(1), 1–10. http://doi.org/10.1016/S1004-9541(08)60026-9

Lourenço, H. R., Martin, O. C., & Stutzle, T. (2001). Iterated Local Search. *SSRN Electronic Journal*. http://doi.org/10.2139/ssrn.273397

Martínez-Bahena , B., Cruz-Chávez , M.A., Peralta-Abarca, J. del Carmen, Juárez-Chávez, J. Y., Ortíz-Huerta, A., Moreno-Bernal, P. (2014). Analysis of a Town's Water Distribution System. In *2014 International Conference on Mechatronics, Electronics and Automotive Engieering* (pp. 206 – 211). IEEE Computer Society.

Martínez-Bahena, B., Cruz-Chavez, M. A., Diaz-Parra, O., Rangel, M. G. M., Rosales, M. H. C., Abarca, J. D. C. P., & Chavez, J. Y. J. (2012). Neighborhood Hybrid Structure for Minimum Spanning Tree Problem. In *2012 IEEE Ninth Electronics, Robotics and Automotive Mechanics Conference* (pp. 191–196). IEEE. http://doi.org/10.1109/CERMA.2012.38

Martínez-Oropeza, A. (2015). *Algoritmo Genético Cooperativo Paralelizado en Ambiente Grid para el Problema de Ruteo Vehicular con Ventanas de Tiempo*. Tesis Doctoral.Universidad Autónoma del Estado de Morelos. Retrieved from http://148.218.108.136:8080/pdf/tesisAlina_2015.pdf

Mladenović, N., & Hansen, P. (1997). Variable neighborhood search.

Computers & Operations Research, *24*(11), 1097–1100. http://doi.org/10.1016/S0305-0548(97)00031-2

Montesinos, P., Garcia Guzman, A., & Ayuso, J. L. (1999). Water Distribution Network Optimization Using a Modified Genetic Algorithm. *Water Resources Research*, *35*(11), 3467–3473.

Nicolini, M., & Zovatto, L. (2009). Optimal Location and Control of Pressure Reducing Valves in Water Networks. *Journal of Water Resources Planning and Management*, *135*(3), 178–187. http://doi.org/10.1061/(ASCE)0733-9496(2009)135:3(178)

Osman, I.H., Kelly, J. P. (1996). *Meta-Heuristics: Theory and Applications* (Kluwer Aca). USA.

Ostfeld, A., Oliker, N., & Salomons, E. (2014). Multiobjective Optimization for Least Cost Design and Resiliency of Water Distribution Systems. *Journal of Water Resources Planning and Management*, *140*(12), 04014037. http://doi.org/10.1061/(ASCE)WR.1943-5452.0000407

Papadimitriou, C. H., & Steiglitz, K. (1998). *Combinatorial Optmization. Algorithms and Complexity*. Mineola. New York: Dover Publications, INC.

Pecci, F., Abraham, E., & Stoianov, I. (2015). Mathematical programming methods for pressure management in water distribution systems. In *Procedia Engineering* (Vol. 119, pp. 937–946). http://doi.org/10.1016/j.proeng.2015.08.974

Reca, J., Martínez, J., Gil, C., & Baños, R. (2008). Application of Several Meta-Heuristic Techniques to the Optimization of Real Looped Water Distribution Networks. *Water Resources Management*, *22*(10), 1367–1379. http://doi.org/10.1007/s11269-007-9230-8

Reeves, C. R., & Rowe, J. E. (2002). *Genetic Algorithms Principles and Perspectives A Guide to GA Theory*. *Kluwer Academic Publishers*. New York, Bosto,Dordrecht,Londos Moscow. http://doi.org/10.1007/b101880

Reis, L. F. R., Porto, R. M., Chaudhrf, F. H. (1997). OPTIMAL LOCATION

OF CONTROL VALVES IN PIPE NETWORKS BY GENETIC ALGORITHM. *Journal of Water Resources Planning and Management*, *123*(1), 317–326.

Rossman, L. A. (1999). The EPANET Programmer's Toolkit for Analysis of Water Distribution Systems. *29th Annual Water Resources Planning and Management Conference*, *39*(1), 1–10. http://doi.org/10.1061/40430(1999)39

Rossman, L. A. (2000). Epanet 2 User 's Manual. *National Risk Management Research Laboratory Office of Research and Development. U.S. Environmental Protection Agency Cincinnati*. http://doi.org/10.1177/0306312708089715

Salazar, G. J. J. (2001). *Programación Matemática* (Díaz de Sa). España.

Saldarriaga, J., & Salcedo, C. A. (2015). Determination of Optimal Location and Settings of Pressure Reducing Valves in Water Distribution Networks for Minimizing Water Losses. *Procedia Engineering*, *119*, 973–983. http://doi.org/10.1016/j.proeng.2015.08.986

Savic, D. A., Walters, G. A. (1997). Genetic Algorithms for Least-Cost Design of Water. *Journal of Water Resources Planning and Management*, *123*(2), 67–77.

Shamir, U. (1974). Optimal Design and Operation of Water Distribution Systems. *Water Resources Research*, *10*(1), 27–36.

Sherali, H. D., & Smith, E. P. (1997). No Title. *Journal of Global Optimization*, *11*(2), 107–132. http://doi.org/10.1023/A:1008207817095

Shu, S., & Zhang, D. (2010). Calibrating water distribution system model automatically by genetic algorithms. *Proceedings - 2010 International Conference on Intelligent Computing and Integrated Systems, ICISS2010*, 16–19. http://doi.org/10.1109/ICISS.2010.5654995

Talbi, E. G. (2009). *Metaheuristics: From Design to Implementation*. Hoboken, New Jersey: John Wiley & Sons, Inc. http://doi.org/10.1002/9780470496916

Van Dijk, M., Van Vuuren, S., & Van Zyl, J. (2008). Optimising water distribution systems using a weighted penalty in a genetic algorithm. *Water SA*, *34*(5), 537–548.

Vasan, A., & Simonovic, S. P. (2010). Optimization of Water Distribution Network Design Using Differential Evolution. *Journal of Water Resources Planning and Management*, *136*(2), 279–287. http://doi.org/10.1061/(ASCE)0733-9496(2010)136:2(279)

Weickgenannt, M., Kapelan, Z., Blokker, M., & Savic, D. A. (2010). Risk-Based Sensor Placement for Contaminant Detection in Water Distribution Systems. *Journal of Water Resources Planning and Management*, *136*(6), 629–636. http://doi.org/10.1061/(ASCE)WR.1943-5452.0000073

Wetzel, A. (1983). *Evaluation of the Effectiveness of Genetic Algorithms in Combinatorial Optimization*. Pittsburgh.

Wright, R., Abraham, E., Parpas, P., & Stoianov, I. (2015). Optimized control of pressure reducing valves in water distribution networks with dynamic topology. *Procedia Engineering*, *119*, 1003–1011. http://doi.org/10.1016/j.proeng.2015.08.994

Apéndice A Datos de las tuberías de la Red del FRM

ID	Origen	Destino	Longitud	Diámetro	ID	Origen	Destino	Longitud	Diámetro
1	1000	1	991	19.05	62	60	61	937	38.1
2	1	2	589.1	19.05	63	61	62	738.4	38.1
3	1000	3	1567	200	64	62	1000	500	38.1
4	3	4	194.8	38.1	65	2000	63	1318	19.05
5	4	5	121.2	38.1	66	63	64	300.6	19.05
6	5	6	1373	38.1	67	64	65	325.1	19.05
7	6	7	498.3	38.1	68	65	66	240.5	19.05
8	7	8	444.4	38.1	69	66	67	1037	19.05
9	8	9	594.3	38.1	70	67	68	47.8	200
10	9	10	661.6	38.1	71	68	69	690	19.05
11	10	11	149.1	38.1	72	69	70	726	19.05
12	11	12	228.7	38.1	73	2000	1001	4007	50.8
13	9	13	773.2	38.1	74	1001	71	2939	50.8
14	13	14	213.5	38.1	75	71	72	445	50.8
15	14	15	277.7	38.1	76	72	73	361	50.8
16	15	16	285.7	38.1	77	71	74	776	50.8
17	16	17	215.6	38.1	78	74	75	174	50.8
18	17	18	245.7	38.1	79	75	76	221	50.8
19	17	19	582.1	38.1	80	76	77	502	50.8
20	19	20	175.4	38.1	81	77	78	181.2	50.8
21	20	21	280	38.1	82	79	78	91.44	50.8
22	21	22	302.6	38.1	83	79	80	1614	50.8
23	22	23	154.9	38.1	84	80	81	418.2	50.8
24	23	24	69.6	38.1	85	81	82	237	50.8
25	24	25	117.8	38.1	86	82	83	177	50.8
26	25	26	514.3	38.1	87	83	84	544	50.8
27	26	27	721.2	38.1	88	83	85	113.9	50.8
28	5	28	822.5	38.1	89	1001	86	1454	50.8
29	28	29	164.4	38.1	90	86	87	557	38.1
30	29	30	322.4	38.1	91	87	88	950.8	38.1
31	30	31	381.6	38.1	92	88	89	2059	19.05
32	31	32	263.5	38.1	93	89	90	642.2	50.8
33	32	33	617.8	38.1	94	90	91	230	50.8
34	18	33	873.3	38.1	95	91	92	422	50.8
35	33	34	564.3	38.1	96	92	93	246.9	50.8
36	34	35	259.5	38.1	97	1001	94	925	50.8
37	35	36	217.2	38.1	98	94	95	81.13	50.8
38	36	37	300.7	38.1	99	95	96	337.8	50.8
39	37	38	598.8	38.1	100	96	97	233.8	50.8
40	38	39	612.9	38.1	101	97	98	611.2	50.8
41	39	40	230.4	38.1	102	98	99	667.3	50.8
42	40	41	420.2	38.1	103	99	100	197.4	50.8
43	41	42	1141	38.1	104	100	101	108.1	50.8
44	42	43	610.9	38.1	105	101	102	320.9	50.8
45	43	44	169.1	38.1	106	102	103	215.9	50.8
46	44	45	713.4	38.1	107	103	104	1720	50.8
47	45	46	1398	38.1	108	104	105	495.6	38.1
48	46	47	542.3	38.1	109	105	80	2271	50.8
49	2000	48	3589	50.8	110	105	106	163.7	38.1
50	48	49	516.1	50.8	111	106	107	616.1	38.1
51	49	50	272.3	50.8	112	107	108	174.4	38.1
52	50	51	242.1	50.8	113	108	109	207	38.1
53	51	52	1089	50.8	114	109	110	208.4	38.1
54	52	53	336.7	50.8	115	110	111	186.3	38.1
55	53	54	215.8	50.8	116	2000	112	3592	50.8
56	54	55	1195	38.1	117	112	113	264	50.8
57	55	56	193.4	38.1	118	113	114	199	50.8
58	56	57	716	38.1	119	114	1002	1026	50.8
59	57	58	487.9	38.1	120	1002	115	1032	50.8
60	58	59	487.9	38.1	121	115	116	221	50.8
61	59	60	237	38.1	122	116	117	252	50.8

123	117	118	235	50.8
124	118	119	437	50.8
125	119	120	285	50.8
126	120	121	285	50.8
127	121	122	649	50.8
128	2000	123	2200	50.8
129	123	124	266	50.8
130	124	125	221	50.8
131	125	126	338	50.8
132	126	127	478	50.8
133	127	128	383	50.8
134	128	129	240	50.8
135	129	130	216	50.8
136	130	131	377	50.8
137	131	132	200	50.8
138	132	133	167	50.8
139	133	134	237	50.8
140	134	135	220	50.8
141	135	136	274	50.8
142	136	137	1374	50.8
143	349	350	289.5	25.4
144	137	138	523.9	50.8
145	138	139	445	50.8
146	139	140	287	50.8
147	140	131	324	50.8
148	136	141	251	50.8
149	141	142	842	50.8
150	142	143	302	50.8
151	143	144	420	50.8
152	144	145	1135	38.1
153	145	146	220	38.1
154	146	147	321	38.1
155	127	148	436	76.2
156	148	149	1605	76.2
157	149	150	675	76.2
158	150	151	417	76.2
159	151	152	379	76.2
160	152	153	331	76.2
161	153	154	290	76.2
162	154	155	491	76.2
163	155	156	289	76.2
164	156	157	214	76.2
165	157	158	812	76.2
166	158	159	261	76.2
167	159	160	510	76.2
168	160	161	818	76.2
169	161	162	146	76.2
170	162	163	131.1	76.2
171	163	164	617.1	76.2
172	164	165	2793	50.8
173	165	166	503.2	50.8
174	166	167	211.6	50.8
175	167	168	107.4	50.8
176	168	169	239.4	50.8
177	169	170	147.7	50.8
178	165	1003	336	50.8
179	1003	171	336	50.8
180	171	172	1036	50.8
181	172	173	198	50.8
182	173	174	389.4	50.8
183	174	175	186.8	50.8
184	175	176	257.9	50.8
185	176	177	198.5	50.8
186	177	178	226.6	50.8
187	178	179	206.5	50.8
188	179	180	314	50.8
189	180	181	543.4	50.8
190	181	182	414.5	25.4
191	181	183	252	50.8
192	183	184	375.5	50.8
193	184	185	247.3	50.8
194	185	186	212.7	50.8
195	186	187	209.8	50.8
196	187	188	17.6	50.8
197	188	189	168	50.8
198	189	190	274.2	50.8
199	190	191	226.5	50.8
200	191	192	196.7	50.8
201	192	193	77.9	50.8
202	193	194	572.1	50.8
203	194	195	91.1	50.8
204	195	196	362	50.8
205	196	197	213.8	50.8
206	197	198	84.32	50.8
207	198	199	136	50.8
208	199	200	133.6	50.8
209	200	201	342.6	50.8
210	201	202	163	50.8
211	202	203	106.5	50.8
212	203	204	336.4	50.8
213	204	205	323.1	50.8
214	205	206	274.4	50.8
215	206	207	715.9	50.8
216	207	208	511.1	50.8
217	208	209	346.3	50.8
218	209	210	61.4	19.05
219	207	211	731.8	31.7
220	211	212	814.3	31.7
221	212	213	128	31.7
222	213	214	55.67	31.7
223	214	215	60.77	31.7
224	215	216	1750	31.7
225	150	217	852	76.2
226	217	218	2065	76.2
227	218	219	417.8	76.2
228	218	220	667	76.2
229	220	221	177.5	76.2
230	221	222	156.4	76.2
231	222	223	744.1	76.2
232	223	224	555.3	50.8
233	224	225	1038	31.7
234	225	226	338.7	31.7
235	226	227	269.3	31.7
236	227	228	476.9	31.7
237	228	229	326.3	31.7
238	224	230	936.9	50.8
239	230	231	493.3	50.8
240	231	232	526.3	50.8
241	232	206	970.9	50.8
242	118	98	1327	50.8
243	127	233	702	101.6
244	233	234	424.2	101.6
245	234	235	653.9	31.7
246	234	236	302.3	50.8
247	236	237	156.4	50.8
248	237	238	103.8	50.8
249	238	239	568.1	50.8
250	239	240	247.7	50.8
251	239	241	370	50.8
252	241	242	150.7	50.8
253	242	243	176.4	50.8
254	243	244	135.6	50.8
255	244	245	203.2	50.8
256	245	246	185.6	50.8

257	246	247	77.7	50.8
258	247	248	457.8	50.8
259	248	249	472.8	50.8
260	249	250	310.9	50.8
261	250	251	254.3	50.8
262	251	252	367.1	50.8
263	252	253	181.8	50.8
264	253	254	280.2	50.8
265	254	255	204.2	50.8
266	255	256	47.19	50.8
267	256	257	494.2	50.8
268	257	258	129.8	50.8
269	259	258	649.8	50.8
270	260	259	186	50.8
271	261	260	140.7	50.8
272	262	261	270.5	50.8
273	263	262	100	200
274	264	263	679	50.8
275	265	264	347	50.8
276	236	265	305.9	50.8
277	266	258	862.9	50.8
278	267	266	251.8	50.8
279	267	268	1528	50.8
280	268	269	285.8	19.05
281	268	270	109.9	50.8
282	270	271	427.8	50.8
283	271	272	286.7	50.8
284	272	273	354.6	50.8
285	273	274	77.61	50.8
286	274	275	610	50.8
287	275	276	344	50.8
288	271	277	138.9	50.8
289	277	278	176.7	50.8
290	278	279	84.65	50.8
291	279	280	335.1	50.8
292	280	281	42.4	50.8
293	281	282	212.6	50.8
294	282	283	326.6	50.8
295	284	267	625.8	101.6
296	285	284	232.1	101.6
297	286	285	379.6	101.6
298	286	287	264.1	101.6
299	287	288	1394	19.05
300	289	287	2013	19.05
301	286	236	1762	101.6
302	290	289	238	38.1
303	1004	290	2938	101.6
304	291	1004	635	76.2
305	292	291	668.7	76.2
306	293	292	153.6	76.2
307	294	293	462.6	76.2
308	295	294	181.1	76.2
309	296	295	3220	76.2
310	2000	296	2206	76.2
311	1004	297	733	50.8
312	297	298	917.6	50.8
313	298	299	355.2	38.1
314	297	300	485.5	50.8
315	300	301	217	50.8
316	301	302	1244	50.8
317	302	303	131.5	50.8
318	303	304	330.4	50.8
319	304	305	113.8	50.8
320	305	306	529	50.8
321	306	307	46.4	50.8
322	2000	308	2613	38.1
323	308	309	429.1	38.1
324	309	1005	2709	38.1
325	1005	310	884	38.1
326	310	311	355.7	25.4
327	310	312	750	38.1
328	312	313	426.6	38.1
329	313	314	719.9	25.4
330	314	315	289.1	25.4
331	315	316	73.5	25.4
332	316	317	226.2	25.4
333	317	318	81.91	25.4
334	318	319	192.3	25.4
335	313	320	880.9	38.1
336	320	321	352.8	25.4
337	321	322	1545	19.05
338	322	323	221.4	19.05
339	321	324	508.9	25.4
340	324	325	234.3	50.8
341	325	326	94.97	50.8
342	320	327	350.8	31.7
343	327	328	240.9	31.7
344	328	329	399.9	31.7
345	329	330	218.2	31.7
346	330	331	127.8	31.7
347	331	332	419.6	31.7
348	332	333	925.6	31.7
349	333	334	235	31.7
350	334	335	273	31.7
351	335	336	727.5	50.8
352	336	337	206.4	31.7
353	337	338	449.3	31.7
354	338	339	692.3	31.7
355	339	340	298.1	31.7
356	340	341	528.6	25.4
357	341	342	211.6	25.4
358	341	343	618.6	25.4
359	343	344	273.5	25.4
360	344	345	228.4	25.4
361	340	346	421.6	25.4
362	346	347	527.4	25.4
363	347	348	229.5	25.4
364	346	349	1045	25.4

Apéndice B Diámetros y costos

Para este trabajo se utilizaron costos de los componentes con base a un promedio de cotizaciones hechas en 2012. Para los tanques el costo es de 15,000 por cada metro de diámetro.

Tabla B.1. Diámetros comerciales y costos para las tuberías

Diámetro (mm)	Costo (unidad monetaria)
12.7	100
19.05	200
25.4	300
31.75	400
38.1	500
50.8	350
76.2	480
101.6	560

Tabla B.2. Diámetros comerciales y costos para las válvulas

Diámetro (mm)	Costo (unidad monetaria)
12.7	1800
19.05	2300
25.4	2500
31.75	2800
38.1	3200
50.8	3500
76.2	3800
101.6	4020

Apéndice C Ejemplo de un archivo .INP

```
[TITLE]

[JUNCTIONS]
;ID       Demand     Pattern
2           150         100
3           160         100
4           155         120
5           150         270
7           160         200
6           165         330

[RESERVOIRS]
;ID     Head        Pattern
1       210             ;

[TANKS]
;ID  Elevation InitLevel  MinLevel  MaxLevel  Diameter

[PIPES]
;ID      Node1    Node2       Length     Diameter  Roughness MinorLoss Status
1           1          2          1000       457.2     130          0          Open   ;
2           2          3          1000       254.0     130          0          Open   ;
3           2          4          1000       406.4     130          0          Open   ;
4           4          5          1000       101.6     130          0          Open   ;
5           4          6          1000       406.4     130          0          Open   ;
6           6          7          1000       254.0     130          0          Open   ;
7           3          5          1000       254.0     130          0          Open   ;
8           5          7          1000        25.4     130          0          Open   ;


[VALVES]
;ID   Node1  Node2  Diameter  Type Setting   MinorLoss


[STATUS]
;ID   Status/Setting


[OPTIONS]
 Units               LPS
 Headloss            H-W
 Specific Gravity    1
 Viscosity           1
 Unbalanced          Continue 10

[COORDINATES]
;Node      X-Coord   Y-Coord
2           6792.87   7327.39
3           4342.98   7327.39
4           6837.42   5389.75
5           4276.17   5456.57
7           4298.44   3808.46
6           6837.42   3808.46
1           9220.49   7371.94

[END]
```

Apéndice D Tipos de datos MPI

```
typedef struct{
        int NTuberia;
        int Vorigen;
        int Vdestino;
        int estado;
        float diametro;
        float longitud;
}tuberia;
```

```
typedef struct{
        int id_dep;
          int altura;
        float NivelIni;
        float NivelMin;
        float NivelMax;
        float diametro;
        float coordX;
        float coordY;
}deposito;
```

```
typedef struct{
          int id_val;
        int NArriba;
        int NAbajo;
        int estado;
        float diametro;
        float csgna;
}valvula;
```

```
typedef struct{
        tuberia TUBSNUEVAS[NTUBSNUEVAS];
        deposito DEPSNUEVOS[NDEPSNEW];
        int TUBSTEMP[NTUBERIAS];
        float VALS[NTUBERIAS];
}Individuo;
```

```
//Tipo de dato para tuberías
 int bloqueTUBS[6]={1,1,1,1,1,1};
MPI_Datatype Tubs;
MPI_Datatype tipoDatoTub[6]={MPI_INT,MPI_INT, MPI_INT,MPI_INT,MPI_FLOAT,MPI_FLOAT};
MPI_Aint movTub[6]={0, sizeof(int), 2*sizeof(int),
3*sizeof(int),4*sizeof(int),(4*sizeof(int))+(sizeof(float))};//desplazamietos offsetof(bytes)
MPI_Type_struct(6, bloqueTUBS, movTub, tipoDatoTub, &Tubs);
MPI_Type_commit(&Tubs);

//Tipo de dato para depósitos
 int bloqueDeps[8]={1,1,1,1,1,1,1,1};
MPI_Datatype Deps;
MPI_Datatype tipoDatoDeps[8]={MPI_INT, MPI_INT,  MPI_FLOAT, MPI_FLOAT, MPI_FLOAT,
MPI_FLOAT, MPI_FLOAT, MPI_FLOAT};
MPI_Aint movDeps[8]={0, sizeof(int), 2*sizeof(int),
2*sizeof(int)+sizeof(float),2*sizeof(int)+(2*sizeof(float)),
        2*sizeof(int)+(3*sizeof(float)),(2*sizeof(int))+(4*sizeof(float)),
(2*sizeof(int))+(5*sizeof(float))};//se puede sustituir con foo
MPI_Type_struct(8, bloqueDeps, movDeps, tipoDatoDeps, &Deps);
MPI_Type_commit(&Deps);

//Tipo de dato anidado
int bloqueInd [4]= {NTUBSNUEVAS, NDEPSNEW,NTUBERIAS,NTUBERIAS};
MPI_Datatype individuo;
 MPI_Datatype tipoDatoInd[4]={Tubs, Deps, MPI_INT,MPI_FLOAT};
 MPI_Aint movInd[4]={0,(NTUBSNUEVAS*sizeof(tuberia)), (NTUBSNUEVAS*sizeof(tuberia))+
(NDEPSNEW*sizeof(deposito)),(NTUBSNUEVAS*sizeof(tuberia))+ (NDEPSNEW*sizeof(deposito))+
(NTUBERIAS*sizeof(int))};
 MPI_Type_struct (4, bloqueInd, movInd, tipoDatoInd, &individuo); //se crea un tipo de dato
 MPI_Type_commit (&individuo);
```

Apéndice E Código de los algoritmos

Código del algoritmo genético secuencial

```
/*Autor: Beatriz Martínez Bahena
Algoritmo genético secuencial para el problema
de redes de distribución de agua*/
#include "toolkit.h"//Funciones de Epanet
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
#include<time.h>
#include<math.h>
/*Declaración constantes*/
#define NVERTICES 350 //#vértices
#define NTUBERIAS 364 //#tuberías
#define NTUBSNUEVAS 3 //# tuberías nuevas
#define NDEPS 6//#depósitos
#define NDEPSNEW 3//#depósitos nuevos
#define NEMBS 1//#embalses
#define NVALS 5
#define NDIAMCOMERS 10//#diámetros comerciales
#define NDIAM 8//#diámetros disponibles
#define NINDS 1000//#individuos
#define gens 100//#generaciones
#define prueba 1
#define PCRUZ 0.7
#define PRESIONMAX 60.0//Presión máxima
#define PRESIONMIN 10.0//Presión mínima
/*PORCENTAJES*/
#define TOTALDEPS (NDEPS+NDEPSNEW+NEMBS)//Total de depósitos
#define TOTALVERS ((NVERTICES+NTUBERIAS)-TOTALDEPS)//Total de vértices
#define PMUT (30 *NINDS)/100
#define PSEL (40 *NINDS)/100
/*Información para los tanques*/
#define RNivelMin 0
#define RNivelMax 15
#define AlturaMin 5
#define AlturaMax 15
#define LargoMin 5
#define LargoMax 15
#define AnchoMin 5
#define AnchoMax 15
/*Información para la simulación en Epanet*/
#define TIPOV "PRV" //(Válvula reductora de presión)
#define CoeRug 100
#define Units              "LPS"
#define Headloss           "H-W"
#define Specific_Gravity   1
#define Viscosity          1
#define Unbalanced         "Continue 10"
/*Declaración de estructuras de datos*/
typedef struct{
        int NTuberia;
        int Vorigen;
        int Vdestino;
        int estado;
        float diametro;
        float longitud;
}tuberia;
typedef struct{
        int idvertice;
```

```
		int cota;
		float demanda;
		float coordX;
		float coordY;
}vertice;
typedef struct{
		int id_dep;
		int altura;
		float coordX;
		float coordY;
}embalse;
typedef struct{
		int id_dep;
		int altura;
		float NivelIni;
		float NivelMin;
		float NivelMax;
		float diametro;
		float coordX;
		float coordY;
}deposito;
typedef struct{
    int id_val;
		int NArriba;
		int NAbajo;
		int estado;
		float diametro;
		float csgna;
}valvula;
typedef struct{
		int id_dep;
		int RCotaMin;
		int RCotaMax;
		float RMINCoordX;
		float RMAXCoordX;
		float RMINCoordY;
		float RMAXCoordY;
		int  NODOS[5];
}RangoTanques;
typedef struct{
 tuberia TUBSNUEVAS[NTUBSNUEVAS];
 deposito DEPSNUEVOS[NDEPSNEW];
 int TUBSTEMP[NTUBERIAS];
 float VALS[NTUBERIAS];
}Individuo;
/* Arreglos de estructuras para los datos originales*/
vertice VERTICES[NVERTICES];//vértices originales
embalse EMBALSES[NEMBS];//embalse originales
deposito DEPOSITOS[NDEPS];//deps originales
tuberia TUBERIAS[NTUBERIAS];//tubs originales
valvula VALVULAS[NTUBERIAS];
RangoTanques RTANQUES[NDEPS];//rango para deps
Individuo POB[NINDS*4],POBLACION[NINDS*4],MEJORIND[2],POBTEMP[2];
/*Declaración de funciones*/
void Cargar_datos(char *,int);
void ConfiguracionInicial(int rank, Individuo *POB);
void Crear_archivo_INP(int ind,int rank,Individuo *POB,int band);
void EvaluacionEpanet(int ,int *,int *,int );
void AgregarIndividuosPoblacion(Individuo *POBLACION,Individuo *POB,int posic,int ind);
void Agregar_Tanques(int ind,Individuo *POB);
int buscar_tuberia(int);
void TuberiasDisponibles(int ind, Individuo *POB);
void CopiarTuberias(int ind, Individuo *POB);
int BuscarLista(int tub,int *ListaTabu,int cont,int ind, Individuo *POB);
void InicializarT();
void InicializarV();
int NumeroTubs();
```

```
int NumeroVers();
void InicializarListaTabu();
void inicializarVals(int ind, Individuo *POBLACION);
void inicializarValvulas(int ind,Individuo *POB);
int NumeroValvulas(int ind,Individuo *POB);
void InicializarTank(int ind, Individuo *POB);
void EvaluacionIndividuo(Individuo *POB,int Ninds);
void Seleccion();
int seleccion_invididuo();
int FactibilidadInds(int ind, Individuo *POB,int rank);
int FactibilidadVals(int ind, Individuo *POB);
void CruzamientoIndividuos(int ind1,int ind2,Individuo *POB,int rank);
void Cruzamiento(Individuo *POBLACION,Individuo *POB,int rank);
void GuardarTiempo(time_t inicio, time_t final,int);
void Mutacion(Individuo *POBLACION,Individuo *POB);
void MutacionIndividuo(int ind,int gen,Individuo *POB,int ID_VAL);
void intercambioINFTanques(Individuo *POBLACION,int ind,int ind2,int t);
void InicializarPoblacion(Individuo *POBLACION);
void Agregar_Valvulas(int ind, Individuo *POB,int *Nval, int *Nvertice);
void CambiosDepositos(int ind,Individuo *POB);
void LeerArchivoINP(int ind,int rank);
int Activar_Valvulas(int ind,Individuo *POB);
void ObtenerPresiones(int);
void Poblacion_inicial(Individuo *POBLACION, Individuo *POB,int rank);
void ActivarValvulasFijas(int ind,Individuo *POB);
void ActualizarValvulas(int ind, Individuo *POB);
void CorroborarDatos();//temporal
void FactibilidadIndsCruzados(Individuo *POBLACION,Individuo *POB,int Vposic);
void ComprobarPoblacion(Individuo *POBLACION);//temporal
void CorroborarDatosPOB(int ind, Individuo *POBLACION);
void ImprimirMejorInd(int gen,int *iters,int rank,Individuo *MEJORIND);
void ReemplazoPoblacion(Individuo *POBLACION, Individuo *POB);
void CruzamientoIndividuos2(int ind1,int ind2,Individuo *POBLACION,int rank);
void MejorIndividuo(Individuo *MEJORIND, Individuo *POB,int gen);
void MutTanques(int ind, Individuo *POB,int t);
void DividirPoblacion(Individuo *POBLACION);
/*variables globales*/
float FitMejorInd=1000000;
int NTotalVals=0,NtotalLinks=0,NtotalNodes=0,presMin=0,presMax=0,MejorInd;
/*Declaración de arreglos globales*/
float DIAMCOMER[NDIAM]={12.7,19.05,25.4,31.75,38.1,50.8,76.2,101.6};
float COSTOSDIAM[8]={100,200,300,400,500,350,280,540};//verificar si estos costos son en pesos
float COSTOSDIAMVALS[8]={1800,2300,2500,2800,3200,3500,3800,4000};//verificar si estos costos son en pesos
int TUBSFIJAS[5]={22,46,66,50,355};
float FITNESS[NINDS];
float SOLS[gens];
int ListaTabu[TOTALVERS];
int PresionNodos[TOTALVERS];
int TUBS[NTUBERIAS];
int INDSELECION[PSEL];
int main(){
   int band1=1,i,band2=2,band3=3,band4=4,band5=5,band6=6,rank=0,band=0,gen,iters=0,posic=0,ind,Ninds;
    time_t inicio,final;
   float costoGral;
   char  ArchV[40]="vertices.txt";
   char  ArchE[40]="embalses.txt";
   char  ArchD[40]="depositos.txt";
   char  ArchT[40]="tuberias.txt";
   char  ArchRT[40]="RangosTanques.txt";
   char  ArchN[40]="nodos.txt";
   srand(time(NULL)/prueba);//modificar la semilla
   /*Cargar informacion de la instancia*/
   inicio=time(NULL);
   Cargar_datos(ArchV,band1);//Información de vértices
   Cargar_datos(ArchE,band2);//Información de Embalses
   Cargar_datos(ArchD,band3);//Información  de Depósitos
   Cargar_datos(ArchT,band4);//Información de Tanques
```

```
   Cargar_datos(ArchRT,band5);//Información de Rangos Tanques
   Cargar_datos(ArchN,band6);//Información de Nodos
   Ninds=NINDS;
   ConfiguracionInicial(rank,POB);//Se generar el archivo .inp inicial y se enviar a Epanet
   Poblacion_inicial(POBLACION,POB,rank);//Se genera población
   EvaluacionIndividuo(POBLACION,Ninds);
   for(gen=0;gen<gens;gen++){
      MejorIndividuo(MEJORIND,POBLACION,gen);//Calcular el mejor individuo
      Seleccion();//Seleccionar el # de invidivuos
      Cruzamiento(POBLACION,POB,rank);//Cruzar los mejores individuos
      FactibilidadIndsCruzados(POBLACION,POB,rank);
      Mutacion(POBLACION,POB);
      EvaluacionIndividuo(POBLACION,Ninds);
      ReemplazoPoblacion(POBLACION,POB);
   }
   ENclose();//cerrar archivo de Epanet
   final=time(NULL);
   GuardarTiempo(inicio,final,rank);/*Guardar el tiempo en el archivo*/
return 0;
}
/*Funcion para cargar los datos de la instancia*/
void Cargar_datos(char *Arch,int archivo){
  int i,c;
  FILE *fp;
  if((fp=fopen(Arch,"rt"))==NULL){
     printf("\nNo se pudo leer el archivo:%s",Arch);
     exit(1);
  }
  else{
    for(i=0;!feof(fp);i++){
                if (archivo==1){
               fscanf(fp,"%d%d%f%f%f",&VERTICES[i].idvertice,&VERTICES[i].cota,&VERTICES[i].demanda,&VERTIC
ES[i].coordX,&VERTICES[i].coordY);
                }
                else if (archivo==2){

fscanf(fp,"%d%d%f%f",&EMBALSES[i].id_dep,&EMBALSES[i].altura,&EMBALSES[i].coordX,&EMBALSES[i].coord
Y);
                }
                else if (archivo==3){

fscanf(fp,"%d%d%f%f%f%f%f%f",&DEPOSITOS[i].id_dep,&DEPOSITOS[i].altura,&DEPOSITOS[i].NivelIni,&DEPO
SITOS[i].NivelMin,&DEPOSITOS[i].NivelMax,&DEPOSITOS[i].diametro,&DEPOSITOS[i].coordX,&DEPOSITOS[i].c
oordY);
                }
                else if (archivo==4){

fscanf(fp,"%d%d%d%f%f",&TUBERIAS[i].NTuberia,&TUBERIAS[i].Vorigen,&TUBERIAS[i].Vdestino,&TUBERIAS[i].l
ongitud,&TUBERIAS[i].diametro);
          TUBERIAS[i].estado=1;
                }
                else if (archivo==5){

fscanf(fp,"%d%d%d%f%f%f%f",&RTANQUES[i].id_dep,&RTANQUES[i].RCotaMin,&RTANQUES[i].RCotaMax,&RT
ANQUES[i].RMINCoordX,&RTANQUES[i].RMAXCoordX,&RTANQUES[i].RMINCoordY,&RTANQUES[i].RMAXCoor
dY);
                }
                else if(archivo==6){
                  for(c=0;c<5;c++)
                    fscanf(fp,"%d",&RTANQUES[i].NODOS[c]);
     }
    }
  }
 fclose(fp);
}
```

```
/*********************************************
*               CODIGO EPANET               *
*********************************************/
/*Funcion crear archivo .inp para enviarlo a epanet*/
void Crear_archivo_INP(int ind,int rank,Individuo *POB,int band){
    int i,nodo,dep,val,t,emb;
    FILE *fichero;
            char nombre[60];
    char pattern[10]=";";
            sprintf(nombre, "IND-%d-%d.inp",ind,rank);
            fichero = fopen(nombre,"w");//elimina, crear un actualización
            //Imprimir vertices
            fprintf(fichero,"\n[JUNCTIONS]\n");
            fprintf(fichero,";ID Nudo\tCota\tDemanda\n");
    for(nodo=0;nodo<NVERTICES;nodo++){

fprintf(fichero,"%d\t%d\t%.3f\n",VERTICES[nodo].idvertice,VERTICES[nodo].cota,VERTICES[nodo].demanda);
            }
            //Imprimir embalses
            fprintf(fichero,"\n[RESERVOIRS]\n");
            fprintf(fichero,";ID Nudo\tAltura\n");
            for(emb=0;emb<NEMBS;emb++){
                    fprintf(fichero,"%d\t%d\n",EMBALSES[emb].id_dep,EMBALSES[emb].altura);
            }
            //Imprimir depositos
            fprintf(fichero,"\n[TANKS]\n");
            fprintf(fichero,";ID Nudo\tAltura\tNivelMin\tNivelIni\tNivelMax\tdiametro\tcurva\n");
            for(dep=0;dep<NDEPS;dep++){//depositos originales

fprintf(fichero,"%d\t%d\t%.2f\t%.2f\t%.2f\t%.2f\t%s\n",DEPOSITOS[dep].id_dep,DEPOSITOS[dep].altura,DEPOSIT
OS[dep].NivelIni,DEPOSITOS[dep].NivelMin,DEPOSITOS[dep].NivelMax,DEPOSITOS[dep].diametro,pattern);
            }
            for(dep=0;dep<NDEPSNEW;dep++){//depósitos nuevos

fprintf(fichero,"%d\t%d\t%.2f\t%.2f\t%.2f\t%.2f\t%s\n",POB[ind].DEPSNUEVOS[dep].id_dep,POB[ind].DEPSNUEVO
S[dep].altura,POB[ind].DEPSNUEVOS[dep].NivelIni,POB[ind].DEPSNUEVOS[dep].NivelMin,POB[ind].DEPSNUEV
OS[dep].NivelMax,POB[ind].DEPSNUEVOS[dep].diametro,pattern);
            }
            fprintf(fichero,"\n[PIPES]\n");
    fprintf(fichero,";ID Linea\tNudo2\tLongitud\tDiáosidad\tPéado\n");
    if(band==0){
      for(t=0;t<NTUBERIAS;t++)

fprintf(fichero,"%d\t%d\t%d\t%.2f\t%.2f\t%d\t%d\t%s\t%s\n",TUBERIAS[t].NTuberia,TUBERIAS[t].Vorigen,TUBERIA
S[t].Vdestino,TUBERIAS[t].longitud,TUBERIAS[t].diametro,CoeRug,0,"open",pattern);
    }
    if(band==1){
      for(t=0;t<NTUBERIAS;t++){
        if(POB[ind].TUBSTEMP[t]==1)

fprintf(fichero,"%d\t%d\t%d\t%.2f\t%.2f\t%d\t%d\t%s\t%s\n",TUBERIAS[t].NTuberia,TUBERIAS[t].Vorigen,TUBERIA
S[t].Vdestino,TUBERIAS[t].longitud,TUBERIAS[t].diametro,CoeRug,0,"open",pattern);
          else

fprintf(fichero,"%d\t%d\t%d\t%.2f\t%.2f\t%d\t%d\t%s\t%s\n",TUBERIAS[t].NTuberia,TUBERIAS[t].Vorigen,TUBERIA
S[t].Vdestino,TUBERIAS[t].longitud,TUBERIAS[t].diametro,CoeRug,0,"closed",pattern);
        }
    }
    //imprimir tuberías nuevas
            for(t=0;t<NTUBSNUEVAS;t++){
      if(POB[ind].TUBSNUEVAS[t].estado==1)

fprintf(fichero,"%d\t%d\t%d\t%.2f\t%.2f\t%d\t%d\t%s\t%s\n",POB[ind].TUBSNUEVAS[t].NTuberia,POB[ind].TUBSNU
EVAS[t].Vorigen,POB[ind].TUBSNUEVAS[t].Vdestino,POB[ind].TUBSNUEVAS[t].longitud,POB[ind].TUBSNUEVAS[
t].diametro,CoeRug,0,"open",pattern);
              else
```

```
fprintf(fichero,"%d\t%d\t%d\t%.2f\t%.2f\t%d\t%d\t%s\t%s\n",POB[ind].TUBSNUEVAS[t].NTuberia,POB[ind].TUBSNU
EVAS[t].Vorigen,POB[ind].TUBSNUEVAS[t].Vdestino,POB[ind].TUBSNUEVAS[t].longitud,POB[ind].TUBSNUEVAS[
t].diametro,CoeRug,0,"closed",pattern);

                }
                //Imprimir propiedades
                fprintf(fichero,"\n[OPTIONS]\n");
                fprintf(fichero,"Units\t%s\n",Units);
                fprintf(fichero,"Headloss\t%s\n",Headloss);
                fprintf(fichero,"Specific Gravity \t%d\n",Specific_Gravity);
                fprintf(fichero,"Viscosity\t%d\n", Viscosity);
                fprintf(fichero,"Unbalanced\t%s\n",Unbalanced);
                //Imprimir valvulas
    fprintf(fichero,"\n[VALVES]\n");
                fprintf(fichero,";ID lítnudoAgArr\tNudoAgAbj\tDiáo\tTipo\tConsigna\n");
    if(band==0){
                  for(val=0;val<NTotalVals;val++)

fprintf(fichero,"%d\t%d\t%d\t%.2f\t%s\t%.2f\n",VALVULAS[val].id_val,VALVULAS[val].NArriba,VALVULAS[val].NAba
jo,VALVULAS[val].diametro,TIPOV,POB[ind].VALS[val]);
                  fprintf(fichero,"\n[STATUS]\n");
                  fprintf(fichero,";Link\tStatus/Setting\n");
                  for(val=0;val<NTotalVals;val++)
        fprintf(fichero,"%d\t%.2f\t%s\n",VALVULAS[val].id_val,POB[ind].VALS[val],"closed");
    }
    if(band==1){
                  for(val=0;val<NTotalVals;val++)
       if(POB[ind].VALS[val]!=0)

fprintf(fichero,"%d\t%d\t%d\t%.2f\t%s\t%.2f\n",VALVULAS[val].id_val,VALVULAS[val].NArriba,VALVULAS[val].NAba
jo,VALVULAS[val].diametro,TIPOV,POB[ind].VALS[val]);
                  fprintf(fichero,"\n[STATUS]\n");
                  fprintf(fichero,";Link\tStatus/Setting\n");
                  for(val=0;val<NTotalVals;val++)
        if(POB[ind].VALS[val]!=0)
          fprintf(fichero,"%d\t%.2f\t%s\n",VALVULAS[val].id_val,POB[ind].VALS[val],pattern);
    }
    //Imprimir coordenadas
                fprintf(fichero,"\n[COORDINATES]\n");
                fprintf(fichero,";ID Nudo\tCoord X\tCoord Y\n");
                for(i=0;i<NVERTICES;i++){

fprintf(fichero,"%d\t%.2f\t%.2f\n",VERTICES[i].idvertice,VERTICES[i].coordX,VERTICES[i].coordY);
                }
                for(emb=0;emb<NEMBS;emb++){

fprintf(fichero,"%d\t%.2f\t%.2f\n",EMBALSES[emb].id_dep,EMBALSES[emb].coordX,EMBALSES[emb].coordY);
                }
                //Imprimir depósitos
                for(dep=0;dep<NDEPS;dep++){

fprintf(fichero,"%d\t%.2f\t%.2f\n",DEPOSITOS[dep].id_dep,DEPOSITOS[dep].coordX,DEPOSITOS[dep].coordY);
                }
                 //Imprimir depósitos nuevos
                for(dep=0;dep<NDEPSNEW;dep++){

fprintf(fichero,"%d\t%.2f\t%.2f\n",POB[ind].DEPSNUEVOS[dep].id_dep,POB[ind].DEPSNUEVOS[dep].coordX,POB[i
nd].DEPSNUEVOS[dep].coordY);
                }
                fprintf(fichero,"\n[END]\n");
                //cerrar archivo
    fclose(fichero);
}
void ConfiguracionInicial(int rank, Individuo *POB){
  int Nval=0,cont=0;
  Agregar_Tanques(1,POB);//Agregar los nuevos tanques de forma aleatoria
```

```
  Agregar_Valvulas(1,POB,&NTotalVals,&cont);//Agregar las válvulas de forma aleatoria
  Crear_archivo_INP(1,0,POB,0);//Bandera IND,
  LeerArchivoINP(1,0);
}
void LeerArchivoINP(int ind,int rank){
    FILE  *f;
    char ArchINP[60];
    sprintf(ArchINP, "IND-%d-%d.inp",ind,rank);
    ENopen(ArchINP," ","");//Abrir archivo .INP
    ENgetcount(EN_LINKCOUNT,&NtotalLinks);
    ENgetcount(EN_NODECOUNT,&NtotalNodes);//Número de nodos
}
void Agregar_Valvulas(int ind, Individuo *POB,int *Nval, int *cont){
 int val,id=4000,band=0,i,j,aux=0,band2=0;
 float csgna,Vcsgna;
 for(val=0;val<NTUBERIAS;val++){
    band=0;
  for(i=0;i<NDEPS;i++){
     if(TUBERIAS[val].Vdestino==DEPOSITOS[i].id_dep || TUBERIAS[val].Vorigen==DEPOSITOS[i].id_dep){
        band=1;
        break;
     }
  }
  for(j=0;j<NDEPSNEW;j++){
     if(TUBERIAS[val].Vorigen==POB[ind].DEPSNUEVOS[j].id_dep ||
TUBERIAS[val].Vdestino==POB[ind].DEPSNUEVOS[j].id_dep){
        band=1;
        break;
     }
  }
  if(TUBERIAS[val].Vorigen==EMBALSES[0].id_dep || TUBERIAS[val].Vdestino==EMBALSES[0].id_dep){
       band=1;
  }
  if(band==0){
     band2=0;
     aux=*cont;
     band2=BuscarLista(val,ListaTabu,aux,ind,POB);//verificar si la tubería no conecte a otra válvula
     if(band2==0){
        VALVULAS[*Nval].id_val=id;
        VALVULAS[*Nval].NArriba=TUBERIAS[val].Vorigen;
        VALVULAS[*Nval].NAbajo=TUBERIAS[val].Vdestino;
        VALVULAS[*Nval].diametro=TUBERIAS[val].diametro;
        VALVULAS[*Nval].csgna= 0;// cerrado =0
        VALVULAS[*Nval].estado=0;
        ListaTabu[*cont]=TUBERIAS[val].Vorigen;
        ListaTabu[(*cont)+1]=TUBERIAS[val].Vdestino;
        *Nval+=1;
        *cont+=2;
        id++;
     }
  }
 }
}
/*Funcion para buscar que un nodo no se encuentre en la lista, para no repetir válvulas en un nodo*/
int BuscarLista(int tub,int *ListaTabu,int cont,int ind, Individuo *POB){
  int j,bandera=0;
  for(j=0;j<cont;j++){
              if(TUBERIAS[tub].Vorigen==ListaTabu[j] || TUBERIAS[tub].Vdestino==ListaTabu[j]){
       bandera=1;
                        break;
              }
              else if(j==(cont)-1)
       bandera=0;
  }
  if(bandera==1)
    return 1;
  else
```

```
    return 0;
}

/*Función para agregar nuevos tanques al sistema con diferentes características*/
void Agregar_Tanques(int ind,Individuo *POB){
  float DIAMETRO,NIVELINCIAL,DIAMTUB,CoordX,CoordY,ANCHO,LARGO,VOLUMEN;
  int
RMinCoordX,RMaxCoordX,RMinCoordY,RMaxCoordY,RCoordX,RCoordY,RCotaMAX,RCotaMIN,ALTURAMAX;
  int t,posic=0,ID_DEP,INDICE,VERTICE,DPOSIC,NTUB=NTUBERIAS,COTA;
  for(t=0;t<NDEPSNEW;t++){
                ID_DEP=RTANQUES[t].id_dep;
                RCotaMIN=RTANQUES[t].RCotaMin;
                RCotaMAX=RTANQUES[t].RCotaMax;
        COTA=rand()% (RCotaMAX-RCotaMIN)+RCotaMIN;
                RMinCoordX=100*RTANQUES[t].RMINCoordX;
                RMaxCoordX=100*RTANQUES[t].RMAXCoordX;
                RCoordX= rand()% (RMaxCoordX-RMinCoordX)+ RMinCoordX;
        CoordX=(float)RCoordX/100;//CoordX
        RMinCoordY=100*RTANQUES[t].RMINCoordY;
        RMaxCoordY=100*RTANQUES[t].RMAXCoordY;
        RCoordY= rand()% (RMaxCoordY-RMinCoordY)+ RMinCoordY;
        CoordY=(float)RCoordY/100;//CoordY
        LARGO=rand()% (LargoMax-LargoMin)+ LargoMin;//Largo del tanque
        ANCHO=rand()% (AnchoMax-AnchoMin)+ AnchoMin;// Ancho del tanque
        DIAMETRO=1.128*(sqrt(LARGO*ANCHO));
        ALTURAMAX=rand()% (AlturaMax-AlturaMin)+ AlturaMin;//Altura del tanque
                VOLUMEN=LARGO*ANCHO*ALTURAMAX;//Volumen M3
        //printf("\nLargo %.2f Ancho %.2f Diametro %.2f Volumen %2f",LARGO,ANCHO,DIAMETRO,VOLUMEN);
        NIVELINCIAL=(float)ALTURAMAX/2;//Nivel Medio del tanque
                /*Asignar información de los nuevos depósitos*/
                POB[ind].DEPSNUEVOS[posic].id_dep=ID_DEP;
        POB[ind].DEPSNUEVOS[posic].altura=COTA;
                POB[ind].DEPSNUEVOS[posic].coordX= CoordX;
        POB[ind].DEPSNUEVOS[posic].coordY= CoordY;
        POB[ind].DEPSNUEVOS[posic].NivelMax= ALTURAMAX;
        POB[ind].DEPSNUEVOS[posic].NivelIni= NIVELINCIAL;
        POB[ind].DEPSNUEVOS[posic].diametro= DIAMETRO;
        POB[ind].DEPSNUEVOS[posic].NivelMin=0;
                /*Crear nueva tuberia para enlazar el deposito con algun nodo*/
                INDICE=rand()%5;//No. de tuberias disponibles
                VERTICE=RTANQUES[t].NODOS[INDICE];//enlace
                DPOSIC=rand()%NDIAM;//No. de diametros disponibles
                DIAMTUB=DIAMCOMER[DPOSIC];//diametro aleatorioPOB
                /*Asignar informacion de las nuevas tuberias*/
        POB[ind].TUBSNUEVAS[posic].NTuberia=NTUB+1;
        POB[ind].TUBSNUEVAS[posic].diametro=DIAMTUB;
        POB[ind].TUBSNUEVAS[posic].Vorigen=ID_DEP;
        POB[ind].TUBSNUEVAS[posic].Vdestino=VERTICE;
        POB[ind].TUBSNUEVAS[posic].estado=1;//open
        POB[ind].TUBSNUEVAS[posic].longitud=250;
                NTUB++;
                posic++;
   }
}
void Poblacion_inicial(Individuo *POBLACION, Individuo *POB,int rank){
  int inds=0,band=0;
  for(;;){//ciclo para individuos
     Agregar_Tanques(1,POB);
     CambiosDepositos(1,POB);//se actualizan los valores de los tanques en Epanet
     inicializarValvulas(1,POB);//checar esta
     band=0;
     ObtenerPresiones(0);
     if(presMin==0){
         ActivarValvulasFijas(1,POB);
         band=Activar_Valvulas(1,POB);
     }else{//Configuración de las mejores características de los tanques
        for(;;){
```

```
        Agregar_Tanques(1,POB);//cambiar características de los tanque
        CambiosDepositos(1,POB);
        ObtenerPresiones(0);
        if(presMin==0)
          break;
       }
       ActivarValvulasFijas(1,POB);
       band=Activar_Valvulas(1,POB); //Configuración de las válvulas activadas
     }
     if(band==1){//agregar individuos
       POBLACION[inds]=POB[1];//AgregarIndividuosPoblacion(POBLACION,POB,inds,1);
       inds++;
     }
     if(inds==NINDS)
       break;
  }//ciclo infinito
}
void CambiosDepositos(int ind,Individuo *POB){
    int tank,link=NTUBERIAS+1,index,cont=0;
    index=NVERTICES+NDEPS+NEMBS+1;// índice de donde empiezan los nuevos tanques
    for(tank=index;tank<=NtotalNodes; tank++){
        ENsetnodevalue(tank,EN_ELEVATION,POB[ind].DEPSNUEVOS[cont].altura);
        ENsetnodevalue(tank,EN_MINLEVEL,POB[ind].DEPSNUEVOS[cont].NivelMin);
        ENsetnodevalue(tank,EN_MAXLEVEL,POB[ind].DEPSNUEVOS[cont].NivelMax);
        ENsetnodevalue(tank,EN_TANKLEVEL,POB[ind].DEPSNUEVOS[cont].NivelIni);
        ENsetnodevalue(tank,EN_TANKDIAM,POB[ind].DEPSNUEVOS[cont].diametro);
        /*Datos para la nueva tuberia*/
        ENsetlinkvalue(link,EN_DIAMETER,POB[ind].TUBSNUEVAS[cont].diametro);
        ENsetlinkvalue(link,EN_NODEEND,POB[ind].TUBSNUEVAS[cont].Vdestino);
        cont++;
        link++;
    }
}
void inicializarValvulas(int ind,Individuo *POB){
  int i,index,l,t;
  for(i=0;i<NTotalVals;i++){
      POB[ind].VALS[i]=0.0;//closed
  }
  for(i=0;i<NTUBERIAS;i++){
     POB[ind].TUBSTEMP[i]=1;//open
  }
  for(i=0;i<NTUBERIAS;i++){
     TUBERIAS[i].estado=1;//open
  }
  index=NTUBERIAS+NTUBSNUEVAS+1;
  for(t=1;t<=NTUBERIAS+NTUBSNUEVAS;t++)
     ENsetlinkvalue(t,EN_INITSTATUS,1);//Se abren todas las tuberías
  for(l=index;l<=NtotalLinks;l++){
     ENsetlinkvalue(l,EN_INITSETTING,0);//Cambiar el valor de tarado a cero
     ENsetlinkvalue(l,EN_INITSTATUS,0);//Se cierra la válvula
  }
}
/*Actualizar los parámetros de las válvulas*/
void ActualizarValvulas(int ind, Individuo *POB){
    int v=0,t,l,index,tub=0;
    index=NTUBERIAS+NTUBSNUEVAS+1;
    for(t=1;t<=NTUBERIAS;t++){
      if(POB[ind].TUBSTEMP[t-1]==1)//open
        ENsetlinkvalue(t,EN_INITSTATUS,1);
      else
        ENsetlinkvalue(t,EN_INITSTATUS,0);
    }
    for(t=NTUBERIAS+1;t<index;t++){
      if(POB[ind].TUBSNUEVAS[tub].estado==1)//open
        ENsetlinkvalue(t,EN_INITSTATUS,1);
      else
        ENsetlinkvalue(t,EN_INITSTATUS,0);
```

```
      tub++;
    }
    for(l=index;l<=NtotalLinks;l++){
       if(POB[ind].VALS[v]!=0){//activo
          ENsetlinkvalue(l,EN_INITSETTING,POB[ind].VALS[v]);//Cambiar el valor de tarado se cambió a Vcsgna
       }else{
          ENsetlinkvalue(l,EN_INITSETTING,0);
          ENsetlinkvalue(l,EN_INITSTATUS,0);
       }
       v++;
    }
}
void ActivarValvulasFijas(int ind,Individuo *POB){
 int val,t,tub,l,index,idVal;
 char id[10];
 index=NTUBERIAS+NTUBSNUEVAS+1;
 for(t=0;t<5;t++){
    tub=TUBSFIJAS[t];
    for(val=0;val<NTotalVals;val++){
      if(VALVULAS[val].NArriba==TUBERIAS[tub-1].Vorigen && VALVULAS[val].NAbajo==TUBERIAS[tub-
1].Vdestino){
        for(l=index;l<=NtotalLinks;l++){
           ENgetlinkid(l,id);
           idVal=atoi(id);
           if(VALVULAS[val].id_val==idVal){
              POB[ind].VALS[val]=30;//generar de forma aleatoria con punto decima
              VALVULAS[val].estado=1;//Activo
              POB[ind].TUBSTEMP[tub-1]=0;//cerrado
              ENsetlinkvalue(l,EN_INITSETTING,30);//activar la válvula con el valor de tarado
              ENsetlinkvalue(tub,EN_INITSTATUS,0);//actualizar estado de la tubería en Epanet
              break;
           }
        }
        break;
      }
    }
  }
}
int Activar_Valvulas(int ind,Individuo *POB){
   float Vcsgna;
   int val=0,posic=0,ntubs,tub,l,idVal,index,band=0,rank=0,csgna,k;
   char id[10];
   ObtenerPresiones(0);//para saber cuántas nodos no cumplen con las presiones adecuadas
   for(k=0;k<100;k++){//iteraciones para activar las válvulas, que pueden ser disponibles
     band=0;
     InicializarT();//SE AGREGO EL DIA 11 de Abril
     TuberiasDisponibles(ind,POB);//analizar las tuberías disponibles para activar válvulas
     ntubs=NumeroTubs();//Numero de tuberías disponibles
     posic=rand()%ntubs;
     tub=TUBS[posic];
     index=NTUBERIAS+NTUBSNUEVAS+1;
     for(val=0;val<NTotalVals;val++){//Se activa la válvula
        if((VALVULAS[val].NArriba==TUBERIAS[tub-1].Vorigen && VALVULAS[val].NAbajo==TUBERIAS[tub-
1].Vdestino)){
          for(l=index;l<=NtotalLinks;l++){
             ENgetlinkid(l,id);
             idVal=atoi(id);
             if(VALVULAS[val].id_val==idVal){
                csgna=rand()%(5050-1050)+1050;//10.5*100; 40.5*100
                Vcsgna=(float)csgna/100;
                POB[ind].VALS[val]=Vcsgna;//generar de forma aleatoria con punto decima
                VALVULAS[val].estado=1;// activar válvula =1
                POB[ind].TUBSTEMP[tub-1]=0;// cerrado =0
                ENsetlinkvalue(l,EN_INITSETTING,Vcsgna);//activar la válvula con el valor de tarado
                ENsetlinkvalue(tub,EN_INITSTATUS,0);//actualizar estado de la tubería en Epanet
                break;
             }
```

```
          }
          break;
        }
    }
    ObtenerPresiones(0);
    if(presMin>=1){//si hay presiones menores a 10 mca
      VALVULAS[val].estado=0;//cerrada
      POB[ind].VALS[val]=0;//Cerrar la válvula
      POB[ind].TUBSTEMP[tub-1]=1;//Abrir la tubería =1
      ENsetlinkvalue(l,EN_INITSETTING,0);//Desactivar la válvula en Epanet
      ENsetlinkvalue(l,EN_INITSTATUS,0);//cerrar la válvula en Epanet
      ENsetlinkvalue(tub,EN_INITSTATUS,1);//Abrir la tubería en Epanet
      ObtenerPresiones(0);
    }
    if((presMin==0)&&(presMax==0)){
      band=1;
      break;
    }
  }
  if(band==0){
     band=0;
    if(presMax>=1)
      band=FactibilidadInds(1,POB,rank);
  }
  if(band==1)
    return 1;
  else
    return 0;
}
void InicializarT(){
  int i;
  for(i=0;i<NTUBERIAS;i++)
     TUBS[i]=0;
}
void TuberiasDisponibles(int ind, Individuo *POB){
 int j,tub=0,band=0,cont=0,nvers;
 nvers=NumeroVers();
 do{
            for(j=0;j<NTUBERIAS;j++){
              if(PresionNodos[cont]==TUBERIAS[j].Vorigen || PresionNodos[cont]==TUBERIAS[j].Vdestino){
       band=buscar_tuberia(TUBERIAS[j].NTuberia);
       if(band==0){
         TUBS[tub]=TUBERIAS[j].NTuberia;
         tub++;
       }
             }
            }
            cont++;
  }while(cont<nvers);
}/*Contar el número de vértices con presiones fuera del rango establecido*/
int NumeroVers(){
  int j,cont=0;
  for(j=0;j<NVERTICES;j++){
    if(PresionNodos[j]!=0)
                cont++;
  }
  return cont;
}
int buscar_tuberia(int tub){
  int j;
  for(j=0;j<NTUBERIAS;j++){
            if(TUBS[j]==tub){
                    return 1;
                    break;
            }
            else if(j==NTUBERIAS-1)
      return 0;
```

```
  }
}
int NumeroTubs(){
  int j,cont=0;
  for(j=0;j<NTUBERIAS;j++){
     if(TUBS[j]!=0)
                          cont++;
  }
  return cont;
}
int FactibilidadInds(int ind, Individuo *POB,int rank){
  int Cotadec=0,dep=0,id_dep=0,f,index,nodo,tank,presMaxAnt=0,band=0;
  float cota,cotaIni=0;
  char id[10];
  for(f=0;f<40;f++){//30 iteraciones, Checar esto
      band=0;
      dep=rand()%NDEPSNEW;//NDEPSNEW=3
      cotaIni=POB[ind].DEPSNUEVOS[dep].altura;
      id_dep=POB[ind].DEPSNUEVOS[dep].id_dep;
      Cotadec=rand()%8+1;//Esta valor se fijó, pero se puede cambiar
      cota=POB[ind].DEPSNUEVOS[dep].altura-Cotadec;//el único campo que cambia
      index=NVERTICES+NDEPS+NEMBS+1;// índice de donde empiezan los nuevos tenues
      for(tank=index;tank<=NtotalNodes; tank++){
         ENgetnodeid(tank, id);
         nodo=atoi(id);//convertir de char a int
         if(id_dep==nodo){
           POB[ind].DEPSNUEVOS[dep].altura=cota;//se modifica la cota para el tanque
           ENsetnodevalue(tank, EN_ELEVATION,cota);//se modifica la cota en Epanet
           break;
         }
      }
      presMaxAnt=presMax;
      ObtenerPresiones(0);
      if(presMax>presMaxAnt && presMin==0){
        POB[ind].DEPSNUEVOS[dep].altura=cotaIni;
        ENsetnodevalue(tank, EN_ELEVATION,cotaIni);//Se hace una conversión de double a float
        ObtenerPresiones(0);
      }
      if(presMin>=1){
        POB[ind].DEPSNUEVOS[dep].altura=cotaIni;
        ENsetnodevalue(tank, EN_ELEVATION,cotaIni);//Se hace una conversión de double a float
        ObtenerPresiones(0);
      }
      if(presMin==0 && presMax==0){
         band=1;
         break;
      }
   }
   if(band==0){
      band=0;
      band=FactibilidadVals(ind,POB);//Llamar la función
   }
   if(band==1)
     return 1;
   else
     return 0;
}
int FactibilidadVals(int ind, Individuo *POB){
  int posicVal=0,j,index,csgna=0,idVal,presMaxAnt=0,l,band=0;
  float Vcsgna,csgnaIni=0;
  char id[10];
  for(j=0;j<30;j++){//Checar este número de operaciones
     band=0;
     do{
       posicVal=rand()%NTotalVals;//solo las activas
     }while(POB[ind].VALS[posicVal]==0);
     csgnaIni=POB[ind].VALS[posicVal];
```

```
    csgna=rand()%(5050-1050)+1050;//10.5*100; 40.5*100
    Vcsgna=(float)csgna/100;
    index=NTUBERIAS+NTUBSNUEVAS+1;
    for(l=index;l<=NtotalLinks;l++){
       ENgetlinkid(l,id);
       idVal=atoi(id);
       if(VALVULAS[posicVal].id_val==idVal){
          POB[ind].VALS[posicVal]=Vcsgna;
          ENsetlinkvalue(l,EN_INITSETTING,Vcsgna);//Cambiar el valor de tarado
          break;
       }
    }
    presMaxAnt=presMax;
    ObtenerPresiones(0);
    if(presMin>=1){
      POB[ind].VALS[posicVal]=csgnaIni;
      ENsetlinkvalue(l,EN_INITSETTING,csgnaIni);//Cambiar el valor de tarado
      ObtenerPresiones(0);
    }
    if(presMax>presMaxAnt && presMin==0){
       POB[ind].VALS[posicVal]=csgnaIni;
       ENsetlinkvalue(l,EN_INITSETTING,csgnaIni);
       ObtenerPresiones(0);
    }
    if(presMin==0 && presMax==0){
       band=1;
       break;
    }
  }
  if(band==1)
    return 1;
  else
    return 0;
}
void ObtenerPresiones(int band){//cambiar a ingles
   int i,node=0;
   char id[10];
   float pressure=0;
   presMin=0,presMax=0;//variables globales
   InicializarV();//Inicializar el arreglo PresionNodos
   ENsolveH();
   for(i=1; i<=NVERTICES; i++){
      pressure=0;
      ENgetnodeid(i, id);
      node=atoi(id);//convertir de char a int
      ENgetnodevalue(i, EN_PRESSURE,&pressure);//obtener Presión
       if(pressure<PRESIONMIN)
          presMin+=1;
       if(pressure>PRESIONMAX){
          PresionNodos[presMax]=node;
          presMax+=1;
       }
   }
}
void InicializarV(){
  int i;
  for(i=0;i<NVERTICES;i++)
     PresionNodos[i]=0;
}
void EvaluacionIndividuo(Individuo *POB,int Ninds){
 int costoMDO=15000,x,t,v,z,ind;//costoMDH costo del material y mano de obra
 float costoTank,diam=0,sum1,costoVals,costoTotal;
 for(ind=0;ind<Ninds;ind++){
    costoTotal=0,sum1=0,costoVals=0,costoTank=0;
    for(t=0;t<NTUBSNUEVAS;t++){ //costo de los tanques
       sum1=0;
       diam=POB[ind].TUBSNUEVAS[t].diametro;
```

```
      for(x=0;x<NDIAM;x++){
        if(diam==DIAMCOMER[x]){
            sum1+=COSTOSDIAM[x]*POB[ind].TUBSNUEVAS[t].longitud;//Obtener el costo del diametro de la nueva
tuberia
            break;
        }
      }
      costoTank+=(costoMDO*POB[ind].DEPSNUEVOS[t].diametro)+sum1;
   }
   for(v=0;v<NTotalVals;v++){ //costo de las válvulas
     if(POB[ind].VALS[v]!=0){//activo
       diam=VALVULAS[v].diametro;
       for(z=0;z<NDIAM;z++){
         if(diam==DIAMCOMER[z]){
           costoVals+=COSTOSDIAMVALS[z];
           break;
         }
       }
     }
   }
  FITNESS[ind]=costoTank+costoVals;//Costo del tanque y las válvulas
 }
}
void MejorIndividuo(Individuo *MEJORIND, Individuo *POB,int gen){
  int j,ind;
  float menor=0;
  menor=FITNESS[0];
  for(j=0;j<NINDS;j++){
            if(FITNESS[j]<=menor){
      menor=FITNESS[j];
              ind=j;
            }
  }
  if(menor<=FitMejorInd){
    MejorInd=ind;
    FitMejorInd=menor;
  }
  SOLS[gen]=FitMejorInd;
  MEJORIND[0]=POB[MejorInd];
  if(gen==gens-1)
     Crear_archivo_INP(0,prueba,MEJORIND,1);//Se crea el archivo .inp del mejor individuo de la generación
}
/*****************************************************************************************
*************************METODO DE SELECCION  X TORNEO ****************************
*********************** MUESTRA DE 2 INDIVIDUOS***********************************
*****************************************************************************************/
void Seleccion(){
 int i=0,iter,ind1,ind2;
 for(iter=0;iter<PSEL;iter++){
    ind1=seleccion_invididuo();
    ind2=seleccion_invididuo();
    INDSELECION[i]=ind1;
    INDSELECION[i+1]=ind2;
    i+=2;
 }
}
int seleccion_invididuo(){
 int ind1=0,ind2=0;
    ind1=rand()%NINDS;
            ind2=rand()%NINDS;
            while(ind2==ind1){
              ind2=rand()%NINDS;
    }
            if(FITNESS[ind1]<=FITNESS[ind2])
      return ind1;
            else if (FITNESS[ind2]<=FITNESS[ind1])
      return ind2;
```

```
}
/*****************************************************************************
**************************METODO DE CRUZAMIENTO***********************************
*****************************************************************************/
void Cruzamiento(Individuo *POBLACION,Individuo *POB,int rank){
 int ind1=0,ind2=0,posic=0,posic1=0,posic2=0,ind=0;
 float pc=0;
 Individuo POBTEMP[2];
 //InicializarPoblacion(POB);
 for(;;){
    posic1=rand()%PSEL;
    ind1=INDSELECION[posic1];
    do{
      posic2=rand()%PSEL;
    }while(posic2==posic1);
    ind2=INDSELECION[posic2];
    pc=(float)rand()/RAND_MAX;
    if(pc<PCRUZ){
      inicializarVals(posic,POBTEMP);
      inicializarVals(posic+1,POBTEMP);
      POBTEMP[posic]=POBLACION[ind1];//AgregarIndividuosPoblacion(POBTEMP,POBLACION,posic,ind1);
      POBTEMP[posic+1]=POBLACION[ind2];// AgregarIndividuosPoblacion(POBTEMP,POBLACION,posic+1,ind2);
      CruzamientoIndividuos(posic,posic+1,POBTEMP,rank);
      POB[ind]=POBTEMP[posic];//AgregarIndividuosPoblacion(POBTEMP,POBLACION,posic,ind1);
      POB[ind+1]=POBTEMP[posic+1];//AgregarIndividuosPoblacion(POB,POBTEMP,ind+1,posic+1);
      ind+=2;
    }else{
      POB[ind]=POBLACION[ind1];// AgregarIndividuosPoblacion(POB,POBLACION,ind,ind1);
      POB[ind+1]=POBLACION[ind2];// AgregarIndividuosPoblacion(POB,POBLACION,ind+1,ind2);
      ind+=2;
    }
    if(ind==NINDS*2)
        break;
 }
 }
 void inicializarVals(int ind, Individuo *POBLACION){
    int val;
    for(val=0;val<NTotalVals;val++)
       POBLACION[ind].VALS[val]=0;
 }
/*********************Aplicación de cruzamiento Método uno************************
  * Elije de forma aleatoria una válvula del ind1 y una válvula del ind2 y cambian**
  * su presión de tarado, también se elige de forma un tanque y cambia información**
  *****************************************************************************/
void CruzamientoIndividuos(int ind1,int ind2,Individuo *POB,int rank){
  int t,posicVal1,posicVal2,i,Nvals1,Nvals2,Nvals,posic,NT;
  float csgna;
  NT=rand()%3+1;
  for(t=0;t<NT;t++){
    intercambioINFTanques(POB,ind1,ind2,t);//checar que pasa si cruzamos los tres punto
  }
  Nvals1=NumeroValvulas(ind1,POB);
  Nvals2=NumeroValvulas(ind2,POB);
  if(Nvals1<=Nvals2)
      Nvals=Nvals1;
  else
      Nvals=Nvals2;
  posic=rand()%Nvals;
  for(i=0;i<posic;i++){//Modificar este punto
     do{
       posicVal1=rand()%NTotalVals;//solo las activas
     }while(POB[ind1].VALS[posicVal1]==0);//closed
     do{
       posicVal2=rand()%NTotalVals;//solo las activas
     }while(POB[ind2].VALS[posicVal2]==0);//closed
        csgna=POB[ind1].VALS[posicVal1];
     POB[ind1].VALS[posicVal1]=POB[ind2].VALS[posicVal2];
```

```
    POB[ind2].VALS[posicVal2]=csgna;
  }
}
void intercambioINFTanques(Individuo *POBLACION,int ind1,int ind2,int t){
   int aux,aux1,aux2,aux3,aux4;
   aux=POBLACION[ind1].DEPSNUEVOS[t].diametro;
   aux1=POBLACION[ind1].DEPSNUEVOS[t].NivelMax;
   aux2=POBLACION[ind1].DEPSNUEVOS[t].NivelMin;
   aux3=POBLACION[ind1].DEPSNUEVOS[t].NivelIni;
   aux4=POBLACION[ind1].TUBSNUEVAS[t].diametro;//Diametro de la nueva tubería
   POBLACION[ind1].DEPSNUEVOS[t].diametro=POBLACION[ind2].DEPSNUEVOS[t].diametro;
   POBLACION[ind1].DEPSNUEVOS[t].NivelMax=POBLACION[ind2].DEPSNUEVOS[t].NivelMax;
   POBLACION[ind1].DEPSNUEVOS[t].NivelMin=POBLACION[ind2].DEPSNUEVOS[t].NivelMin;
   POBLACION[ind1].DEPSNUEVOS[t].NivelIni=POBLACION[ind2].DEPSNUEVOS[t].NivelIni;
   POBLACION[ind1].TUBSNUEVAS[t].diametro=POBLACION[ind2].TUBSNUEVAS[t].diametro;
   POBLACION[ind2].DEPSNUEVOS[t].diametro=aux;
   POBLACION[ind2].DEPSNUEVOS[t].NivelMax=aux1;
   POBLACION[ind2].DEPSNUEVOS[t].NivelMin=aux2;
   POBLACION[ind2].DEPSNUEVOS[t].NivelIni=aux3;
   POBLACION[ind2].TUBSNUEVAS[t].diametro=aux4;
}
/*Contar el número de válvulas activadas*/
int NumeroValvulas(int ind,Individuo *POB){
 int j,cont=0;
  for(j=0;j<NTotalVals;j++){
    if(POB[ind].VALS[j]!=0)//activo
                        cont++;
  }
  return cont;
}
/*** Funcion para hacer los individuos factibles***/
void FactibilidadIndsCruzados(Individuo *POBLACION,Individuo *POB,int rank){
   int ind,posic=0,Factible=0,i,cont=0,NumFac=0,Num=0,IND[NINDS],infact=0;
   for(ind=0;ind<NINDS*2;ind++){
      CambiosDepositos(ind,POB);// Se actualizan los valores de los depósitos en Epanet
      ActualizarValvulas(ind,POB);//Se actualizan los valores de las válvulas en Epanet
      ObtenerPresiones(0);
      if(presMin>=1 || presMax>=1){
        Factible=FactibilidadInds(ind,POB,rank);
        if(Factible==0){
          infact++;
        }
        if(Factible==1){
          POBLACION[posic]=POB[ind];//AgregarIndividuosPoblacion(POBLACION,POB,posic,ind);
          posic++;
          NumFac++;
          IND[posic]=ind;
        }
      }else{
         POBLACION[posic]=POB[ind];// AgregarIndividuosPoblacion(POBLACION,POB,posic,ind);
         posic++;
         Num++;
         IND[posic]=ind;
      }
     cont++;
   if(posic==(NINDS))
      break;
   }
}
/*****************************************************************************
*************************METODO DE MUTACION*************************************
*****************************************************************************/
void Mutacion(Individuo *POBLACION, Individuo *POB){
  int
iter,Factible=0,ind,ind1,posic=0,posic1=0,gen,i,t,genes,Vals1[NTotalVals],Nvals=0,id_val,val1=0,posic_val=0,NT;
  for(;;){//ciclo
    ind=rand()%(NINDS);//elegir individuo a mutar
```

```
    inicializarVals(posic,POBTEMP);
    POBTEMP[posic]=POBLACION[ind];//AgregarIndividuosPoblacion(POBTEMP,POBLACION,posic,ind);
    CambiosDepositos(posic,POBTEMP);// Se actualizan los valores de los depósitos en Epanet
    ActualizarValvulas(posic,POBTEMP);//Se actualizan los valores de las válvulas en Epanet
    val1=0;
    for(i=5;i<NTotalVals;i++){
      if(POBTEMP[posic].VALS[i]!=0){//activo
         Vals1[val1]=VALVULAS[i].id_val;
         val1++;
       }
    }
    genes=rand()%val1;
    for(gen=0;gen<genes;gen++){
       do{
          posic_val=rand()%val1;
       }while(Vals1[posic_val]==-1);
       id_val=Vals1[posic_val];
       MutacionIndividuo(posic,gen,POBTEMP,id_val);
       Vals1[posic_val]=-1;
    }
    NT=rand()%3+1;
    for(t=0;t<NT;t++){
        MutTanques(posic,POBTEMP,t);
    }
    CambiosDepositos(posic,POBTEMP);
    ObtenerPresiones(0);
    if(presMin>=1 || presMax>=1){
       Factible=FactibilidadInds(posic,POBTEMP,0);
       if(Factible==1){
          POB[posic1]= POBTEMP[posic];//AgregarIndividuosPoblacion(POB,POBTEMP,posic1,posic);
          posic1++;
       }
    }
    else{
       POB[posic1]=POBTEMP[posic];//AgregarIndividuosPoblacion(POB,POBTEMP,posic1,posic);
       posic1++;
    }
    if(posic1==PMUT)
       break;
  }
  for(ind=posic1;ind<(NINDS);ind++){
     ind1=rand()%(NINDS);
     POB[ind]=POBLACION[ind1];//AgregarIndividuosPoblacion(POB,POBLACION,ind,ind1);
  }
}
/**Cambiar de forma aleatoria las características de los tanques, como diámetros y medias**/
void MutTanques(int ind, Individuo *POB,int t){
  float DIAMETRO,NIVELINCIAL,ANCHO,LARGO;
  int ID_DEP,ALTURAMAX;
  ID_DEP=RTANQUES[t].id_dep;
  LARGO=rand()% (LargoMax-LargoMin)+ LargoMin;//Largo del tanque
  ANCHO=rand()% (AnchoMax-AnchoMin)+ AnchoMin;// Ancho del tanque
  DIAMETRO=1.128*(sqrt(LARGO*ANCHO
  ALTURAMAX=rand()% (AlturaMax-AlturaMin)+ AlturaMin;//Altura del tanque
  NIVELINCIAL=(float)ALTURAMAX/2;//Nivel Medio del tanque
  POB[ind].DEPSNUEVOS[t].id_dep= ID_DEP;
  POB[ind].DEPSNUEVOS[t].NivelMax= ALTURAMAX;
  POB[ind].DEPSNUEVOS[t].NivelIni= NIVELINCIAL;
  POB[ind].DEPSNUEVOS[t].diametro= DIAMETRO;
  POB[ind].DEPSNUEVOS[t].NivelMin=0;
}
 void MutacionIndividuo(int ind,int gen,Individuo *POB,int ID_VAL){
  int rank=0,idVal,index=0,l,tub,val,LISTAVALS[NTotalVals],v=0;
  float cgna=0;
  char id[10];
  for(val=0;val<NTotalVals;val++){//Se activa la válvula
     if(VALVULAS[val].id_val==ID_VAL){
```

```
      cgna=POB[ind].VALS[val];
      for(tub=1;tub<=NTUBERIAS;tub++){
         if(VALVULAS[val].NArriba==TUBERIAS[tub-1].Vorigen && VALVULAS[val].NAbajo==TUBERIAS[tub-
1].Vdestino){
            POB[ind].TUBSTEMP[tub-1]=1;//open
            ENsetlinkvalue(tub,EN_INITSTATUS,1);
            POB[ind].VALS[val]=0;//desactivada =0
            VALVULAS[val].estado=0;//closed
            break;
         }
      }
      break;
   }
 }
 index=NTUBERIAS+NTUBSNUEVAS+1;
 for(l=index;l<=NtotalLinks;l++){
    ENgetlinkid(l,id);
    idVal=atoi(id);
    if(ID_VAL==idVal){
       ENsetlinkvalue(l,EN_INITSETTING,0);//activar la válvula con el valor de tarado
       ENsetlinkvalue(l,EN_INITSTATUS,0);//actualizar estado de la tubería en Epanet
       break;
    }
 }
 ObtenerPresiones(0);
 if(presMin>=1 || presMax>=1){
   POB[ind].TUBSTEMP[tub-1]=0;//closed
   ENsetlinkvalue(tub,EN_INITSTATUS,0);
   POB[ind].VALS[val]=cgna;//generar de forma aleatoria con punto decima
   VALVULAS[val].estado=1;//activo
   ENsetlinkvalue(l,EN_INITSETTING,cgna);//activar la válvula con el valor de tarado
 }
}
void GuardarTiempo(time_t inicio, time_t final,int rank){
              FILE *fichero;
              char nombre[20];
             double f_Tiempo=0;
              sprintf(nombre, "tiempo-%d.txt",prueba);
              fichero=fopen(nombre, "a+");
              f_Tiempo=difftime(final,inicio);//Tiempo en segundos
              fprintf(fichero,"\n%d\t%.4lf\n",rank,f_Tiempo/60);
              fclose(fichero);
}


/**** Reemplazo de la nueva población a la anterior para la siguiente generación****/
void ReemplazoPoblacion(Individuo *POBLACION, Individuo *POB){
 int ind;
 for(ind=0;ind<NINDS;ind++)
    POBLACION[ind]=POB[ind];//AgregarIndividuosPoblacion(POBLACION,POB,ind,ind);
}
```

Algoritmo genético distribuido con comunicación colectiva

```
int main(int argc, char *argv[]){

 int band1=1,band2=2,band3=3,band4=4,band5=5,band6=6,ind,rank,nprocs;
 int tam,x,i,emb,ver,tub,dep,Nvals=0,band=0,presMin=0,presMax=0,gen,iters=0,val,subP,Ninds;
 time_t inicio,final;
   float costoGral;
    char  ArchV[40]="vertices.txt";
    char  ArchE[40]="embalses.txt";
    char  ArchD[40]="depositos.txt";
    char  ArchT[40]="tuberias.txt";
    char  ArchRT[40]="RangosTanques.txt";
    char  ArchN[40]="nodos.txt";
    //Iniciar Ambiente MPI
    MPI_Init(&argc,&argv);
    MPI_Status status;
    //Tipo de dato para tuberias
    int bloqueTUBS[6]={1,1,1,1,1,1};
              MPI_Datatype Tubs;
              MPI_Datatype tipoDatoTub[6]={MPI_INT,MPI_INT, MPI_INT,MPI_INT,MPI_FLOAT,MPI_FLOAT};
              MPI_Aint movTub[6]={0, sizeof(int), 2*sizeof(int),
3*sizeof(int),4*sizeof(int),(4*sizeof(int))+(sizeof(float))};//desplazamietos offsetof(bytes)
              MPI_Type_struct(6, bloqueTUBS, movTub, tipoDatoTub, &Tubs);
              MPI_Type_commit(&Tubs);
              //Tipo de dato para Depositos
    int bloqueDeps[8]={1,1,1,1,1,1,1,1};
              MPI_Datatype Deps;
              MPI_Datatype tipoDatoDeps[8]={MPI_INT, MPI_INT,  MPI_FLOAT, MPI_FLOAT, MPI_FLOAT,
MPI_FLOAT, MPI_FLOAT, MPI_FLOAT};
              MPI_Aint movDeps[8]={0, sizeof(int), 2*sizeof(int), 2*sizeof(int)+sizeof(float),2*sizeof(int)+(2*sizeof(float)),
              2*sizeof(int)+(3*sizeof(float)),(2*sizeof(int))+(4*sizeof(float)), (2*sizeof(int))+(5*sizeof(float))};
              MPI_Type_struct(8, bloqueDeps, movDeps, tipoDatoDeps, &Deps);
              MPI_Type_commit(&Deps);
    //Tipo de dato anidado (Ind)
              int bloqueInd[4]={NTUBSNUEVAS,NDEPSNEW,NTUBERIAS,NTUBERIAS};
    MPI_Datatype individuo;
    MPI_Datatype tipoDatoInd[4]={Tubs,Deps,MPI_INT,MPI_FLOAT};
    MPI_Aint movInd[4]={0,(NTUBSNUEVAS*sizeof(tuberia)), (NTUBSNUEVAS*sizeof(tuberia))+
(NDEPSNEW*sizeof(deposito)),
                        (NTUBSNUEVAS*sizeof(tuberia))+ (NDEPSNEW*sizeof(deposito))+ (NTUBERIAS*sizeof(int))};
    MPI_Type_struct(4, bloqueInd, movInd, tipoDatoInd, &individuo);//se crea un tipo de dato
    MPI_Type_commit(&individuo);
    //obtener valores de # de procesos y id de c/proceso
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
              MPI_Comm_size(MPI_COMM_WORLD, &nprocs);
              MPI_Barrier(MPI_COMM_WORLD);
    //Generar semilla
    srand(time(NULL)/(rank+1));//modificar la semilla
              //Cargar datos de la instancia en todos los nodos
    //inicio=(time(NULL));
    Cargar_datos(ArchV,band1);
    Cargar_datos(ArchE,band2);
    Cargar_datos(ArchD,band3);
    Cargar_datos(ArchT,band4);
    Cargar_datos(ArchRT,band5);
    Cargar_datos(ArchN,band6);
    ConfiguracionInicial(0,POB);

    if(rank==0){
      Ninds=NINDS;
      Poblacion_inicial(POBLACION,POB,rank);
      EvaluacionIndividuo(POBLACION,Ninds);
    }
```

```
    inicio=(time(NULL));
    for(gen=0;gen<gens;gen++){
        if(rank==0){
          MejorIndividuo(MEJORIND,POBLACION,gen);//Calcular el mejor individuo
          Seleccion();//Seleccionar el # de invidivuos
          Cruzamiento(POBLACION,POB,rank);//Cruzar los mejores individuos
          DividirPoblacion(POB);//Divide la población en subpoblaciones
        }
        MPI_Scatter(&POBTOTAL,NINDST, individuo, &POB2, NINDST, individuo, 0, MPI_COMM_WORLD);
         FactibilidadIndsCruzados(POBLACION,POB2,rank);//Sale POBLACION
         Mutacion(POBLACION,POB2);//Sale POB2

        MPI_Gather(&POB2, NINDST, individuo, &POBTOTAL, NINDST, individuo, 0, MPI_COMM_WORLD);
        if(rank==0){
          ReemplazoPoblacion(POBLACION);
          EvaluacionIndividuo(POBLACION,NINDS);
        }
    }
    ENclose();//cerrar archivo de Epanet
    /*if(rank==0)
      imprimirSOLS();*/
    final=time(NULL);
    GuardarTiempo(inicio,final,rank);
    MPI_Finalize();
    return 0;
}
```

El resto de las funciones son las mismas que del algoritmo genético secuencial.