Universidad Autónoma del Estado de Morelos

Instituto de Investigación en Ciencias Básicas y Aplicadas
Centro de Investigación en Ingeniería y Ciencias Aplicadas

Modelo de satisfacción de restricciones aplicado a sistemas de tuberías para fluidos

Tesis para obtener el grado de:
Doctorado en ingeniería y Ciencias Aplicadas con opción terminal en Tecnología Eléctrica

Presenta:
M.C. Ariadna Ortiz Huerta

Director: Dr. Marco Antonio Cruz Chávez
Co-asesor Dr. Sergio Alonso Serna Barquera

Sinodales:
Dra. Margarita Tecpoyotl Torres, Dr. Martín Gerardo Martínez Rangel, Dr. Jesús Perfecto Xamán Villaseñor, Dr. Álvaro Zamudio Lara, Dr. Arturo Molina Ocampo.

Febrero 2020

Cuernavaca, Morelos

UNIVERSIDAD AUTÓNOMA DEL ESTADO DE MORELOS

Instituto de Investigación en Ciencias Básicas y Aplicadas

INSTITUTO DE INVESTIGACIÓN EN CIENCIAS BÁSICAS Y APLICADAS

Jefatura de Posgrado en Ingeniería y Ciencias Aplicadas

"1919-2019: en memoria del General Emiliano Zapata Salazar"

Cuernavaca, Morelos, a 08 de noviembre de 2019.

DR. ROSENBERG JAVIER ROMERO DOMÍNGUEZ
COORDINADOR DEL POSGRADO EN
INGENIERÍA Y CIENCIAS APLICADAS
P R E S E N T E

Atendiendo a la solicitud para emitir DICTAMEN sobre la revisión de la TESIS "Modelo de satisfacción de restricciones aplicado a sistemas de tuberías para fluidos" que presenta la alumna **ARIADNA ORTIZ HUERTA,** para obtener el título de **DOCTORADO EN INGENIERÍA Y CIENCIAS APLICADAS** con opción terminal en **TECNOLOGÍA ELÉCTRICA.**

Nos permitimos informarle que nuestro voto es:

NOMBRE	DICTAMEN	FIRMA
DR. ÁLVARO ZAMUDIO LARA	Aprobado	
DR. ARTURO MOLINA OCAMPO	Aprobado	
DRA. MARGARITA TECPOYOTL TORRES	Aprobado	
DR. MARTÍN GERARDO MARTÍNEZ RANGEL (FCAel)	Aprobado	
DR. JESÚS PERFECTO XAMÁN VILLASEÑOR (CENIDET)	Aprobado	
DR. SERGIO ALONSO SERNA BARQUERA	Aprobado	
DR. MARCO ANTONIO CRUZ CHÁVEZ	Aprobado	

PLAZO PARA LA REVISIÓN 20 DÍAS HÁBILES (A PARTIR DE LA FECHA DE RECEPCIÓN DEL DOCUMENTO)

NOTA. POR CUESTION DE REGLAMENTACIÓN LE SOLICITAMOS NO EXCEDER EL PLAZO SEÑALADO, DE LO CONTRARIO LE AGRADECEMOS SU ATENCIÓN Y NUESTRA INVITACIÓN SERÁ CANCELADA.

Av. Universidad 1001 Col. Chamilpa, Cuernavaca Morelos, México, 62209
Tel. (777) 329 70 00, ext. 6208 / raquel.sotelo@uaem.mx

Dedicatorias

A mi **mamá Micaela** porque ha sido una de las personas
más importantes en mi vida,
y que gracias a su apoyo pude cumplir
con las mayores metas de mi vida.

A mis hijos **Lis y David** que desde que nacieron,
me ensañaron que la vida se disfruta con cada pequeño momento.
Gracias por su amor que me ha
despertado cada mañana, motivada a
seguir a pesar de los obstáculos que se presentan en la vida

A mis **hermanas** que son mis
confidentes y cómplices de cada travesura,
gracias, por esos **sobrinos Uzi y Yari**
que desde que nacieron, fueron como mis hijos.

Agradecimientos

A mis amigos de toda la vida **Lidia, Diana, Reina, Eva, Marco, Monse y Marisol** por su amistad incondicional a pesar de la distancia.

A mis amigos que me acompañaron en esta aventura y que desde los conocí me han apoyado en mi vida profesional y personal **Carmen, Mara, Pedro, Juanita, Alfonso, Marta, Yainier, Jazmín, Bety y Abigail.**

A mi **Director de tesis Dr. Marco Antonio Cruz Chávez**,

por sus enseñanzas y amistad.

A mi **Co-director de tesis Dr. Sergio**

por sus enseñanzas y correcciones de esta tesis.

A mis revisores **Dra. Margarita Tepoyotl, Dr. Martin Martínez, Perfecto Xamán Villaseñor** por sus correcciones de esta tesis,

en especial a la **Dr. Perfecto Xamán Villaseñor** ya que fue mi revisor externo y a pesar de no estar cerca se tomó la molestia de asistir a mis presentaciones

A la **Dra. Carmen Peralta** por su ayuda incondicional

desde que la conocí.

Al Dr. Jesús Xamán del Centro Nacional de Investigación y Desarrollo Tecnológico (**CENIDET**),

al Dr. Carlos Coello Coello del Centro de Investigación y de Estudios Avanzados (**CINVESTAV**)

por brindarme la oportunidad de realizar estancias doctorales en sus instituciones.

Finalmente A **CONACYT** y **TELMEX**

por brindarme el apoyo económico sin el cual no hubiera sido posible la realización de este sueño.

Índice

Lista de Figuras

Lista de Tablas

Resumen

El petróleo es muy cotizado ya que de él se derivan múltiples productos pero el producto derivado de él por excelencia es combustible que se usa para el transporte. Sin embargo cuando se trata de transportar este fluido se generan gastos que empieza desde el trazado de las tuberías para llevar el combustible de forma más fácil y económica a donde se requiere.

Este trabajo se basa en la problemática en donde se busca la factibilidad del trazado de tuberías, es decir que las presiones en cada punto se cumplan para que el fluido se pueda transportar de manera eficiente. Para solucionar el problema que se presenta en la búsqueda de la factibilidad del trazado de tuberías, se desarrolla una hiperheurística que resuelve un modelo de optimización propuesto para el diseño óptimo de tuberías que transportan combustible y se prueba con dos instancias, la primera es de tipo teórica la cual tiene características de la zona donde se va a trabajar y segunda es de tipo real las cual se localiza en el estado de Veracruz de la Llave, México. La instancia real proviene de un trabajo previo en el cual se hace el trazado considerando restricciones de zonas arqueológicas, poblados, mantos acuíferos, propiedad privada, pago por derechos de vía, pantanos, carreteras, autopistas y otros. Por lo que no se pueden mover los nodos así como las tuberías, solo se puede modificar los diámetros poner válvulas y/o bombas. La instancia real parte de la colonia de Playa Linda del puerto de Veracruz y llega hasta la población de Santa Rita, municipio de Veracruz de la Lave.

La hiperheurística resuelve un modelo propuesto de programación no lineal que describe las dos instancias a resolver, la cual fue ejecutada en el cluster Cuexcomate ubicado en el Universidad Autónoma del Estado de Morelos (UAEM), campus Chamilpa.

Abstract

The oil is very quoted since multiple products are derived from it but the product derived from it par excellence is fuel that is used for transport. However, when it comes to transporting this fluid, expenses are generated that start from the layout of the pipes to take the fuel more easily and economically to where it is required.

This work is based on the problem where the feasibility of pipeline design is sought, that is, the pressures at each point are met so that the fluid can be transported efficiently. To solve the problem that arises in the search for the feasibility of pipeline tracing, a hyperheuristic is developed that solves a proposed optimization model for the optimal design of pipelines that transport fuel and is tested with two instances, the first one is of type theoretical which has characteristics of the area where it is going to work and second is of a real type which is located in the state of Veracruz de la Llave, Mexico. The real instance comes from a previous work in which the layout is made considering restrictions of archaeological zones, towns, aquifers, private property, payment for rights of way, swamps, roads, highways and others. So you can not move the nodes as well as the pipes, you can only change the diameters put valves and / or pumps. The real instance starts from the colony of Playa Linda in the port of Veracruz and reaches the town of Santa Rita, municipality of Veracruz de la Lave.

Hyperheuristics solves a proposed non-linear programming model that describes the two instances to be resolved, which was executed in the Cuexcomate cluster located at the Autonomous University of the State of Morelos (UAEM), Chamilpa campus.

Capítulo1. Introducción

1.1. Introducción.

“El petróleo es una mezcla oleosa que se componen principalmente de hidrocarburos que se encuentran en yacimientos y se producen industrialmente perforando terrenos en los que existen acumulaciones de petróleo y gas” (Kirk & Othmer, 2002). Existen diversos productos que se obtienen de la transformación del petróleo como: gasolina, nafta, keroseno, aceite diesel o combustible y la industria Petroquímica es la encargada de realizarlo (Kirk & Othmer, 2002). El petróleo es muy cotizado ya que de él se derivan múltiples productos desde combustible hasta ropa, zapatos dispositivos electrónicos y otros, pero el producto derivado de él por excelencia es el combustible que se usa en el transporte y fabricación de productos que ayudan al ser humano a llevar una vida más cómoda.

El petróleo ha transformado la vida de las personas y la economía de las naciones. Su descubrimiento creó riqueza, modernidad, pueblos industriales prósperos y nuevos empleos, motivando el crecimiento de las industrias mencionadas. Se conoce que la formación del petróleo está asociada al desarrollo de rocas sedimentarias depositadas en ambientes marinos o próximos al mar, y que es el resultado de procesos de descomposición de organismos de origen vegetal y animal, que en tiempos remotos quedaron incorporados en esos depósitos.

Uno de los problemas que presenta la industria petrolera es la distribución de los diferentes productos los cuales son transportados por diferentes ductos (Djebedjian *et al.,* 2008), que reciben diferentes nombre de acuerdo al fluido que estén transportando. El gas natural es transportado a través de los llamados gasoductos, los hidrocarburos líquidos, especialmente

aceite, son transportados por medio de oleoductos y si es una mezcla de ambos, a través oleogasoducto (Kirk & Othmer, 2002). Dichos ductos forman una red mallada, ramificada o mixta.

La problemática principal es el costo para el transporte de combustibles así como buscar el mejor lugar para construir una planta generadora de hidrocarburos. Entonces el problema a resolver es el trazado de nuevas redes de tuberías y la construcción de plantas de generación de diferentes derivados del petróleo.

La importancia del estudio de este tipo de problemas radica en que son inspirados en problemáticas de la vida diaria que en ciertos casos propicia fuertes pérdidas económicas a diversas empresas, por lo que al encontrar buenas soluciones a estos problemas en su forma teórica, es posible realizar algunas modificaciones a los modelos y algoritmos, de modo que puedan ser aplicables a problemas reales.

El problema de Redes de Distribución de fluidos, es un problema de interés para los investigadores por su amplia aplicación práctica. Durante más de tres décadas ha sido estudiado ampliamente. Se han propuesto diferentes formulaciones matemáticas y un gran número de métodos de solución (Cruz *et al.,* 2009b). No obstante, en la práctica sólo se han resuelto instancias pequeñas debido a la enorme complejidad del problema.

El Diseño de Redes de Distribución de fluidos, consiste en elegir los componentes básicos que forman parte de la red de tal forma que la red resultante sea una red de costo mínimo. Los componentes para el diseño de una red son los siguientes: tuberías, bombas, válvulas y fuentes de abastecimiento que pueden ser inagotables o de cierta capacidad (EPANET, 2016). Las tuberías se encuentran disponibles, en diferentes comercios, con diferentes diámetros y materiales. La función que

desempeñan en una red de distribución de fluidos es la de llevarlo desde la fuente hasta el /los usuarios de la red. Las válvulas reguladoras, son elementos que ayudan a modular la presión obtenida en una red de distribución. Las bombas de potencia o estaciones de bombeo son elementos indispensables, en la distribución de fluidos. Finalmente, las fuentes de abastecimiento son elementos imprescindibles en la red de distribución de agua y pueden ser pozos de extracción o cualquier otra que sea permanente (Ávila Melgar, 2015; ITA, 2015; Martinez-Bahena, 2016).

Las redes de distribución de fluidos pueden clasificarse, de acuerdo con la topología que presenten, en redes en serie, redes ramificadas y redes malladas, donde:

Una red en serie es aquella que no contiene mallas ni ramificaciones; es una conexión entre dos o más nodos de forma lineal. Generalmente, tienen un nodo fuente, un nodo final y uno o más nodos intermedios, la dirección del flujo es fija, desde la fuente al otro extremo. Ésta es la topología más simple que existe para las redes de distribución de fluidos, se utilizan generalmente para el transporte de agua en pequeños poblados, de medicamentos, de alimentos, de combustible y todo aquel problema en donde no se requiera flujo constante en el caso que haya un corte en la tubería.

Una red ramificada es un conjunto de redes en serie y no contiene mallas. Estas redes presentan una estructura similar a la estructura de un árbol, presentan un nodo fuente, más de un nodo final y uno o más nodos intermedios. Generalmente, son redes que se utilizan para la distribución de agua en pequeñas comunidades rurales, zonas industriales o en zonas de riego. En la práctica, estas redes presentan como inconveniente la suspensión del servicio en diferentes puntos de la red cuando ocurren roturas o fugas en alguna tubería. Esto se debe a que en las redes

ramificadas, sólo existe un camino para llegar de un punto a otro en la red. Por esta razón, algunos usuarios se ven afectados con la suspensión del servicio cuando ocurren fallas en la red.

Una red mallada es una red que contiene circuitos o mallas, lo cual significa que para un usuario el fluido puede llegar a través de diferentes caminos. En estas redes, la interrupción del servicio ocasionado por rupturas en las tuberías ocurre con menor frecuencia, ya que el agua puede llegar a su destino utilizando diferentes trayectorias. Por esta razón, una rotura en una tubería, generalmente, no afecta gravemente a otros puntos de la red, así que en estas redes es menos frecuente que los usuarios tengan fallas en el servicio. A pesar de que es más costoso implementar las redes malladas que las redes ramificadas, se justifica la implementación porque las redes malladas ofrecen mayor confiabilidad. El empleo de estas redes es habitual en redes de distribución urbanas, redes de riego, en donde se exige al sistema una gran seguridad en el suministro de agua.

En el presente trabajo se diseña una red en serie debido a que este tipo de redes es el que se utiliza para el transporte de combustible.

1.2. Optimización combinatoria.

La optimización combinatoria es una rama de la optimización en matemáticas aplicadas y en ciencias computacionales, relacionada con la investigación de operaciones, instanciaría de algoritmos y teoría de la complejidad computacional (Cook *et al.,* 1977). Todos los problemas considerados en la literatura por la Optimización Combinatoria pueden ser clasificados de acuerdo a su tratabilidad, de donde surge *la Teoría de la Complejidad*, la cual se encarga de dar una clasificación a estos problemas, dividiéndolos en clases: P, NP y NP-duros.

Los problemas P, son aquellos que pueden ser resueltos por un algoritmo determinista en tiempo polinomial, es decir, que la relación entre el tamaño del problema y su tiempo de ejecución es polinómica, por lo que se consideran problemas tratables. Los problemas que se encuentran en la clase P generalmente se resuelven de forma óptima mediante el uso de métodos exactos (algoritmos voraces, algoritmos de ramificación y poda, algoritmos de divide y vencerás, entre otros). Los métodos exactos aseguran la obtención del óptimo global para resolver problemas de la clase P (Cook, 1995).

En el caso de los problemas NP, estos sólo pueden ser tratados por un algoritmo no determinista acotado en tiempo polinomial; por su parte, los problemas clasificados como NP-duros se caracterizan por su extrema complejidad, ya que desde el punto de vista computacional son considerados intratables, además de que sus instancias resueltas hoy en día son más pequeñas en comparación con las resueltas para problemas NP. Algunos de los problemas NP-duros más conocidos y tratados son: el problema del Agente Viajero (TSP) del que se deriva el problema de Ruteo de Vehículos con Ventanas de Tiempo (VRPTW), el problema de Calendarización de Máquinas en Talleres de Manufactura (JSSP), el problema de Satisfactibilidad (SAT), el problema de la Mochila, redes de distribución de fluidos, la repartición de grafos, el problema de árbol de expansión mínima, problemas de la ruta más corta, red de flujo máximo, redes de tuberías, entre otros (Martínez-Bahena *et al.,* 2012; Stützle, 2019).

Los problemas NP-duros pueden ser acotados en tiempo polinomial por medio de algoritmos conocidos como heurísticas, los cuales son procedimientos para resolver un problema de optimización bien definido y clasificado como difícil, mediante una aproximación intuitiva. De manera general, puede decirse que existen técnicas de solución conocidas como métodos exactos y técnicas de solución conocidas como métodos de

aproximación, en la que la estructura del problema se utiliza de forma inteligente para obtener una buena solución en un tiempo computacional razonable. Las heurísticas son consideradas como una de las mejores opciones para abordar problemas NP-Completos porque prometen encontrar buenas soluciones en tiempos de cómputo razonables, tal como lo es el espacio de búsqueda del problema de Diseño de Redes de Distribución de fluidos (Baños *et al.,* 2010).

Por ejemplo, para una instancia de prueba teórica del problema de Diseño de Redes de Distribución, con 20 tuberías y un conjunto discreto de 10 diámetros de tuberías a elegir, el espacio de búsqueda para el problema equivale a 1020 diseños diferentes. Aun cuando se realizaran 1 000 000 de operaciones por segundo, se necesitarían 3 000 000 de años para generar todos los diseños mediante un método de enumeración. De acuerdo con lo anterior, la aplicación de métodos exactos en problemas de gran tamaño suele ser inviable y es precisamente en estos casos cuando se recurre al uso de heurísticas, cuyo término generalmente se usa en contraposición al término exacto (Melián *et al.,* 2003; Ávila-Melgar, 2015).

Para darle solución a este tipo de problemas surgen las denominadas *metaheurísticas*, las cuales son una clase de métodos aproximados que están diseñados para resolver problemas complejos de optimización combinatoria, para los que los heurísticos clásicos no son efectivos, por lo que las metaheurísticas proporcionan un marco general, sin embargo las hiperheurísticas son relativamente nuevas las cuales incorporan los mejor de todas las heurísticas de tal forma que hace más eficiente el algoritmo (Cowling *et al.,* 2000; Garcia *et al.,* 2011; Villela *et al.,* 2013).

En general, las técnicas heurísticas se ajustan de manera fiel a un problema de optimización discreto y son muy útiles para abordar problemas donde los espacios de búsqueda son grandes y/o complejos. Ésta es la

razón principal que justifica el uso de técnicas heurísticas en el desarrollo de esta tesis, enfocándose así en el estudio e implementación de una hiperheurística.

El problema tratado en esta tesis, es representado por una red, donde sus arcos representan los ductos o estaciones de bombeo, y sus nodos son los puntos físicos de interconexión. Se consideran dos tipos de variables continuas de decisión: el flujo másico a través de cada arco de la red, y los niveles de presión en cada nodo. Así, desde la perspectiva de la optimización, el problema de minimizar los costos del diseño de la red para lograr una factibilidad en el trazado de tuberías es modelado como un problema de programación no lineal, donde tanto la función de costo y el conjunto de restricciones son típicamente no lineales y no convexos. Se conoce que algunos de los problemas de programación lineal no convexos se clasifican como problemas NP-duros, (Horst *et al.,* 1995; Savic, 1997; Ríos *et al.,* 2009; Papadimitriou & Steiglitz, 2013) sin embargo como la red que se va a trabajar es en serie, no se encontró información en la literatura que clasifique a este problema como un NP, no obstante pudiendo ser el problema en estudio un problema de tipo P en el cual se pudiera utilizar la programación con métodos clásicos de optimización (método de Newton, Newton Raphson, Jacobi, Gauss-Seidel entre otros), programar la solución con este tipo de métodos es más complicado de forma matemática que si se utilizan métodos heurísticos para optimizar, los cuales aportan soluciones en tiempos razonables y de muy buena calidad.

La investigación realizada en este trabajo de tesis se enfoca precisamente en el problema de diseño de redes de distribución de fluidos, utilizando para la solución de dicho problema una hiperheurística en forma secuencial, trabajando en conjunto con un simulador hidráulico llamado EPANET.

1.3. Antecedentes bibliográficos.

Diversos autores han aplicado al problema de diseño de redes de distribución de fluidos (Shamir, 1974; Savic, 1997; Van Dijk *et al.,* 2008; Liu *et al.,* 2008; Hui Zhang *et al.,* 2009; Shu, 2010; Weickgenannt *et al.,* 2010; Vasan, 2010; Lei *et al.,* 2011; Chang *et al.,* 2013; Kurek, 2013; Ostfeld *et al.,* 2014; D'Ambrosio *et al.,* 2015), quienes han planteado diferentes métodos de optimización de forma teórica. Pero solo algunos han trabajado en el este problema de optimización de redes con fluidos como el gas natural seco y líquido así como cualquier combustible.

Por ejemplo Borraz, 2004 que optimiza redes de gas natural, por medio de un método basado en técnicas no tradicionales de programación no lineal, y propone una heurística de búsqueda local con lista tabú, para obtener soluciones aproximadas de buena calidad. Sus resultados muestran que la heurística propuesta genera soluciones factibles y en menos tiempo debido a la lista tabú.

En cambio en el trabajo de Oteiza *et al.,* 2015 comparan dos metaheurísticas, recocido simulado y algoritmo genético, en un caso de prueba real para transportar gas natural líquido; los resultados demuestra algoritmo genético mejor desempeño que el algoritmo simulado. Oteiza *et al.,* 2018 presentan una hiperheurística para reducir los tiempos computacionales para encontrar una solución del diseño de redes de tuberías. Usan un algoritmo genético, un recocido simulado y una optimización de colonias de hormigas. Los autores emplearon instancias de pruebas reales para evaluar el impacto de la función de aprendizaje, finalmente, concluyeron que la hiperheurística demostró ser competitiva, ya que es capaz de explorar un amplio espacio de búsqueda de forma rápida y ofrecer soluciones satisfactorias (Oteiza *et al.,* 2018).

Sin embargo Liu *et al.,* 2008 mencionan que basados en la calidad de la energía, analizan la pérdida y la eficiencia de la energía de todos los componentes del sistema de transporte del oleoducto de crudo. La expresión de la pérdida de energía la determinaron y el modelo matemático del sistema de transporte por tuberías lo establecieron con el objetivo de minimizar la pérdida de energía total del sistema, que combina los dos algoritmos de optimización del algoritmo genético monobjetivo y multiobjetivo. Los autores utilizaron un oleoducto externo en el noreste de China como instancia de prueba real, la heurística la utilizaron para realizar una optimización jerárquica para la disposición de los equipos y los parámetros de operación de cada estación en el sistema de transporte de la tubería. Los resultados encontrados muestran que el costo total de transporte de petróleo y la pérdida total de energía del sistema de transporte por tuberías, después de la optimización mejoraron en comparación con los no optimizados.

Con base en esta resumida revisión bibliográfica, se aprecia que hasta la fecha no se encuentra una hiperheurística aplicada a problemas de redes de distribución de fluidos, para una instancia con bombas y válvulas. Por lo tanto, en este trabajo se propone obtener la factibilidad y optimización de una red de distribución de fluidos que corresponde a un hidrocarburo para una red teórica con características del relieve del Estado de Veracruz, México y una de prueba de este mismo estado.

1.4. Objetivo general.

Desarrollar una hiperheurística que permita una distribución de fluidos de manera eficiente en un sistema de redes en serie para minimizar costos.

1.4.1. Objetivo particular.

Obtener la factibilidad optimizada para una red de distribución de un fluido newtoniano, a través de una hiperheurística y de la satisfacción de restricciones de las ecuaciones gobernantes.

1.5. Alcance de la investigación.

- El modelo de optimización será evaluado mediante una hiperheurística y comprobando que se satisfacen las restricciones del sistema en el estudio con el simulador EPANET.
- La instancia de prueba teórica utilizada para el algoritmo, será una trayectoria creada de forma aleatoria.
- La instancia de prueba real será una trayectoria del estado de Veracruz, México.
- El fluido a trabajar será gasolina a temperatura ambiente.
- Se realiza una sintonización de los parámetros de control de la hiperheurística, para mejorar el tiempo de desempeño del algoritmo en referencia a la eficacia y eficacia.
- El algoritmo tendrá una función de aprendizaje que permitirá la búsqueda inteligente en cada punto de decisión.

1.6. Contribuciones de la tesis.

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

- Dar solución a un problema de optimización de aplicación real por medio de una hiperheurística para una red de distribución de fluidos cuyas características principales sea estado permanente, fluido newtoniano, isotérmico, desarrollado e incompresible.

- Diseño y desarrollo de una hiperheurística, basada en heurísticas de bajo nivel, con una función de aprendizaje.
- Se aplicará a una instancia de prueba real la cual viene de un problema de red de distribución de hidrocarburos, que permita distribuir el fluido a diferentes puntos ubicado en el estado de Veracruz, México.
- Se modificará el código libre de EPANET con la finalidad de configurar las características adecuadas para diferentes fluidos, para poderlo utilizar en la solución de la red de distribución de hidrocarburos en Veracruz, México.

1.7. Organización de la tesis.

Esta tesis doctoral se ha estructurado en seis capítulos, los cuales se describen a continuación.

Capítulo 1. Introducción

El primero de los capítulos se introduce al problema de optimización con el que se trabaja con un pequeño resumen de los trabajos realizados en redes de distribución con hidrocarburos en donde aplicaron heurísticas además, este capítulo contiene las contribuciones, alcances y los objetivos de esta tesis.

Capítulo 2. Estado del arte

En este capítulo se revisan los trabajos más relevantes referidos al problema de las redes de distribución de fluidos. El capítulo se divide en dos secciones solución de problemas de redes de distribución con métodos exactos y con heurísticas de bajo y alto nivel. Además, dado que en esta tesis se ha utilizado una hiperheurística también se agregan los trabajos encontrados.

Capítulo 3. Modelo matemático

En este apartado se explican los fundamentos teóricos del problema de redes de distribución de fluidos. Para ello, en primer lugar, se realiza una explicación general del mismo, donde se describen sus características y se exponen los detalles, requisitos específicos y el modelo matemático del caso de prueba que se ha abordado en esta tesis.

Capítulo 4. Metodología

Este capítulo describe de manera teórica las estrategias de resolución utilizadas en esta tesis doctoral. Por tanto, en primer lugar se explica de manera introductoria las aproximaciones algorítmicas que podrían aplicarse al problema que se aborda en este trabajo, se desarrollan los conceptos básicos sobre heurísticas de bajo nivel así como los de cada una de las técnicas concretas aplicadas en la resolución del problema abordado. Posteriormente se explica en detalle cómo se modela y resuelve el problema de redes de distribución de fluidos haciendo uso de las técnicas explicadas al inicio de este capítulo.

Capítulo 5. Resultados

En esta sección se describen las condiciones en las que se han realizado las pruebas y los experimentos llevados a cabo en nuestra investigación. A continuación se exponen y analizan los resultados de los ajustes paramétricos y diferentes configuraciones probadas para cada heurística de bajo nivel. Por último, se presentan y comparan los datos finales arrojados por cada estrategia. A este respecto, se analizan dos problemas teóricas, la que fue creada y la real, la cual está basada en la topología del estado de Veracruz.

Capítulo 6. Conclusiones y trabajos futuros

Finalmente, en esta sección se discuten las conclusiones alcanzadas tras la investigación desarrollada. De este modo, en este capítulo se expondrán, por un lado, las conclusiones alcanzadas tras el análisis de los resultados, y por otro, las principales aportaciones conseguidas con la realización de esta tesis doctoral. Finalmente, en este apartado se trazan posibles líneas de trabajo futuro que se desprenden del trabajo presentado en este documento.

Capítulo 2. Estado del arte

2.1. Introducción.

En este capítulo contiene conceptos importantes para el desarrollo de este trabajo además de un análisis bibliográfico de los trabajos que tienen relación con la investigación desarrollada. Debido al gran número de publicaciones que se encuentran en la bibliografía sobre el problema de Redes de distribución de fluidos, en este capítulo se ha limitado a la discusión de las mismas a aquellas que utilizan estrategias sin heurísticas, es decir métodos exactos, y con heurísticas para su optimización

2.2. *Conceptos relacionados a la hiperheurística*

2.2.1. Heurísticas.

Heurística deriva de la palara griega *heuriskein* que significa encontrar o descubrir y significa método con reglas empíricas para encontrar soluciones a problemas, basado en la experiencia el cual no tiene pruebas de optimalidad (Wetzel, 1983) y ésta sirve para darle solución a diversos problemas pero la desventaja que tiene es que no permite escapar de los óptimos locales sin embargo se encuentran estos óptimos locales en tiempo razonable (Burke *et al.,* 2005; Talbi, 2002; Talbi, 2009). A estas heurísticas se les conoce de bajo nivel, esto es debido el bajo nivel de abstracción, Algunos ejemplos son búsqueda local, búsqueda local iterada, lista tabú, Boltzman y otros.

2.2.2. Heurísticas de bajo nivel.

- ***Búsqueda local.***

El método básico de búsqueda local es el conocido como ascenso de colinas (Jacobson & Yücesan, 2004; Jacobson *et al.,* 2004). Dicho nombre, que para problemas de minimización se denomina descenso de colinas, se debe a que para admitir una determinada solución, ésta debe conllevar una mejora respecto a la anterior, es decir, no se permiten movimientos que empeoren la calidad de la solución. Por esta razón, el algoritmo de ascenso de colinas se detiene cuando se encuentra un óptimo local, no siendo efectivo en problemas combinatorios complejos), el ascenso de colinas solamente es capaz de alcanzar el mínimo local representado por la solución *s*', y no el mínimo global (*s*"). Por esta razón, esta técnica ha quedado relegada por otras ya que no permite escapar de óptimos locales.

El proceso inicia con una solución cualquiera conjunto de soluciones en el vecindario, del cual se elige una solución a través de un movimiento. Este movimiento se realiza a través de un proceso estocástico, el cual mejora la función objetivo, es decir, si se requiere minimizar, entonces si se cumple se reemplaza la solución anterior, si esta no mejora no es remplazada, y así se repite hasta alcanzar el criterio de paro de la búsqueda local (Černý, 1985; Kirkpatrick, Gelatt, & Vecchi, 1983a).

- **Búsqueda local iterada.**

Búsqueda local iterativa (ILS –*Iterated Local Search*) (Den Besten *et al.,* , 2001; Lourenço *et al.,* 2003; Stützle, 2019) mencionan que es una heurística que aplica de forma iterativa un método de búsqueda local o sea basa su funcionamiento en practicar en cada iteración una perturbación y

una operación de búsqueda local sobre la solución del problema con la que se está operando. Este método trabaja en conjunto con la búsqueda local, y es mediante éste que se obtiene la primera solución con la cual se iniciará el recorrido por el espacio de soluciones hasta mejorar progresivamente. Stützle afirman que ILS es una de las metodologías más sencillas y eficaces para evitar quedar atrapados en óptimos locales (Stützle, 2019).

- **Búsqueda tabú.**

Búsqueda tabú fue propuesta por Glover (1989,1990). Es en esencia un procedimiento de búsqueda orientado determinista simple; Laguna *et al.,* 1993), el cuál restringe la búsqueda a fin de optimizar la búsqueda local a través del almacenamiento de la historia del proceso en su memoria. Se prohíben los movimientos en la vecindad que tiene ciertas cualidades ya almacenadas a fin de dirigir el proceso de búsqueda lejos de las soluciones que se duplican o se asemejan a soluciones generadas anteriormente. La función de la memoria a corto plazo permite mediante ciertos olvidos, a fin de aceptar movimientos que no estén en la lista tabú más reciente (Laguna *et al.,* 1993), para evitar ciclismo a corto plazo (Gendreau, 2003; Glover *et al.,* 2000). Sin embargo el estado tabú de un movimiento no es absoluto. Los criterios de la aspiración permiten a un movimiento tabú ser seleccionado si es que éste logra un nivel de cierta calidad en la solución generada (Martinez-Bahena, 2011; Martínez-Bahena *et al.,* 2012).

La Búsqueda Tabú parte de dos conceptos básicos: vecindad y movimiento prohibido o como se llamará a partir de este momento, movimiento tabú. Estas definiciones caracterizan a la Búsqueda Tabú y hacen la diferencia con respecto a otras metodologías. El primero de los conceptos visto en forma aislada es el mismo que usan la mayoría de los

métodos de búsqueda local (Castillo, 2008). Diferentes aplicaciones se pueden encontrar en Rochat & Taillard en 1995.

- **Cruzamiento.**

Durante esta fase se cruzan o mezclan los individuos seleccionados en la fase anterior. Es decir, los genes de los dos padres se mezclan entre sí para dar lugar a los diferentes hijos. Existen diversos métodos de cruce, pero los más utilizados son los siguientes:

- Cruce basado en un punto: son recombinados dos individuos seleccionados para jugar el papel de padres, por medio de la selección de un punto de corte, para después intercambiar las secciones que se encuentran a la derecha de dicho punto. Es decir, los genes del padre1 a la izquierda del punto de corte forman parte del hijo1 y los situados a la derecha formaran parte del hijo 2, mientras que con el padre 2 sucederá lo contrario.
- Cruce punto a punto: este tipo de cruce es similar al anterior pero realizándose para cada gen de los padres. Por tanto, en este cruce los genes pares del padre1 formarán parte del hijo1 y los genes impares formarán parte del hijo 2, mientras que para el padre 2 sucederá lo contrario (Goldberg & Holland 1989).
- Cruce multipunto: en este tipo de cruce se selecciona aleatoriamente la cantidad de puntos que se van a utilizar para el cruce. De esta forma, y de manera análoga al anterior cruce, se irán intercambiando los genes para formar los dos nuevos hijos.
- Cruces específicos de codificaciones no binarias: Para este tipo de codificación se pueden definir, además de los anteriores, otros tipos de operadores de cruce:

- Media: el gen de la descendencia toma el valor medio de los genes de los padres. Tiene la desventaja de que únicamente se genera un descendiente en el cruce de dos padres.
- Media geométrica: cada gen de la descendencia toma como valor la raíz cuadrada del producto de los genes de los padres. Presenta el problema añadido de qué signo dar al resultado si los padres tienen signos diferentes.
- Extensión: se toma la diferencia existente entre los genes situados en las mismas posiciones de los padres y se suma el valor más alto o se resta del valor más bajo. Solventa el problema de generar un único descendiente.

2.2.3. Heurísticas de alto nivel.

Entonces surgió una nueva idea “Metaheurística” que fue introducido por Fred en 1986. Los métodos metaheurísticos 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 o alto nivel (Talbi, 2009) que se pueden utilizar como estrategias de guía en el diseño de la heurística para resolver problemas específicos de optimización (Talbi, 2002). Dentro de los algoritmos metaheurísticos se encuentran resultados óptimos globales en tiempo razonable.

Entonces en busca del camino a la solución varios autores han propuesto diferentes algoritmos que están basadas en analogías con procesos de la naturaleza. Las metaheurísticas se dividen en dos clases: basadas en trayectoria y las basadas en poblaciones (Alancay *et al.,* 2016), en las primeras requieren con 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 como es recocido simulado. Las metaheurísticas basadas en población (Glover, 2003), 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, optimización de colonia de hormigas, optimización por enjambre de partículas, búsqueda dispersa, evolución diferencial, estrategias evolutivas (Alba *et al.,* 2013; Gendreau, 2003; Talbi, 2009).

Recocido simulado.

Recocido simulado 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). Este algoritmo se basa en la analogía entre el proceso de recocido de sólidos y los problemas de optimización combinatoria.

Es una técnica de búsqueda local estocástica que aproxima el valor mínimo de la función de costo sobre un conjunto finito de soluciones. En SA los umbrales son positivos y estocásticos. El SA realiza una búsqueda aleatoria orientada que fue introducida como una analogía de la física estadística del proceso simulado de una fundición de metal hasta que su energía mínima se alcanza, por lo que las configuraciones son análogas a los estados de un sólido, mientras que el costo de la función *f* y el parámetro de control *c* son equivalentes a la energía y a la temperatura respectivamente (Kirkpatrick *et al.,* 1983; Cerny, 1985).

Algoritmos evolutivos.

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 y Glover, 2003.

2.2.4. Hiperheurísticas.

Las hiperheurísticas por su parte funcionan al más alto nivel de abstracción lo cual no trabajan directamente con los costos de las solución, sino que trabajan a nivel de heurísticas (Villela *et al.,* 2012). Fisher y Tomson en 1963 usaron por primera vez la utilizaron y posteriormente el termino fue introducido por (Cowling *et al.,* 2000), el cual se dice que es una heurística para elegir heurística (Burke *et al.,* 2003), o sea, un método que utiliza un mecanismo de control para seleccionar de entre varias posibles heurísticas de bajo nivel (Villela *et al.,* 2013) e incorporan lo mejor de todas las heurísticas, hace más eficiente el algoritmo (Garcia *et al.,* 2011)

Un objetivo de la hiperheurística es que dará lugar a sistemas más generales que son capaces de manejar una amplia gama de dominios de problemas en lugar de nuevas metaheurística que tiende a ser personalizada para un problema particular o una clase estrecha de los problemas. Las hiperheurísticas son en gran medida la elección de la heurística inteligente o algoritmo en una situación dada (Chakhlevitch & Cowling, 2008). En cierto sentido, funcionan a un nivel más alto de abstracción que las heurísticas y gestionan la elección o de lo que debería ser un método heurístico de bajo nivel para ser aplicado en un momento dado, dependiendo de las características de la región del espacio de soluciones en exploración, es decir, podría ser una hiperheurística operando como una metaheurística solo que a nivel de heurísticas, esta no tiene control de lo que sucede en las

heurísticas de bajo nivel, solo evalúa el desempeño al terminar de operar (Burke *at el*. 2009).

2.2.4.1. Clasificación de la hiperheurística.

De manera análoga a la clasificación de metaheurísticas. Las hiperheurísticas se pueden dividir en constructivas e iterativas. Las constructivas van generando cada parte de una solución en cada iteración y las iterativas inician con una solución completa y de forma iterativa van modificando la solución siguiendo una dirección de búsqueda en función de estructuras vecinas (Bai *et al.,* 2005).

Otra forma de clasificación de las hiperheurísticas es sin aprendizaje y con aprendizaje. En las primeras, las llamadas a las técnicas de bajo nivel se realizan acorde a una secuencia predeterminada. En cambio, las hiperheurísticas con aprendizaje van modificando las preferencias de elección en función del rendimiento histórico de las técnicas (Soubeiga, 2003).

Otros autores como Chakhlevitch & Cowling, 2008 clasifican las hiperheurísticas en cuatro categorías:

1. Aleatoria: Se elige la técnica a utilizar al azar. Existen algunas variantes en esta categoría. Por ejemplo luego de elegir una técnica, se aplica hasta que no ofrezca mejoras, en ese caso se vuelve a elegir otra técnica.

2. Codiciosa: En cada iteración se aplican todas las técnicas, seleccionando la que mejores resultados ofrezca, luego se cambia de técnica.

3. Basadas en metaheurísticas: La solución se representa como un conjunto de metaheurísticas que se deben aplicar siguiendo un cierto orden. Dicho orden se va modificando según las particularidades de cada metaheurísticas.

4. Con aprendizaje: La elección se basa en la eficacia acumulada de cada técnica desde el inicio del programa. En cada iteración se recompensa y castiga a las técnicas para ir estableciendo un orden de prioridades.

Esta última es la que se eligió para aplicarla en esta hiperheurística debido a que la función de aprendizaje permite una toma de decisión en cada punto de la hiperheurística.

2.2.5. Estructuras de vecindad.

Todo problema de optimización tiene un conjunto, ya sea finito o infinito de soluciones posibles, por lo que se requiere la utilización de técnicas que permitan la mejor explotación del espacio de soluciones, y por ende obtener soluciones de buena calidad. Este tipo de técnicas aplicadas a búsqueda local son mejor conocidas como estructuras de vecindad (Martinez-Oropeza, 2010).

El funcionamiento de las estructuras de vecindad es iterativo, debido a que permiten la mejor explotación del espacio de soluciones, de modo que define la vecindad así como el conjunto de todas aquellas soluciones que pueden ser alcanzables a partir de una solución *s* por medio de un movimiento (Cruz Chavez *et al.,* 2009a). Un movimiento puede ser una inserción, eliminación o intercambio de componentes en una solución. Para esta investigación un movimiento estará definido como un intercambio de dos partes de una solución (Cruz Chavez *et al.,* 2009a).

Las estructuras de vecindad se han implementado ampliamente a búsqueda local en diversos métodos heurísticos para tratar problemas de optimización, los cuales, de acuerdo a la naturaleza de los mismos, llegan a ser intratables mediante técnicas de cálculo determinístico, debido a su complejidad; o bien se han implementado para minimizar el tiempo computacional requerido para resolver este tipo de problemas.

2.3. Solución al problema de Redes de distribución de fluidos con métodos exactos.

Para permitir el consumo energético, mejorar la economía y los beneficios sociales (Alperovits & Shamir, 1977) desde hace más de tres décadas, se han implementado métodos exactos y heurísticos así como modelos de optimización para encontrar el diseño óptimo de una red de distribución de fluidos. Los métodos exactos proporcionan una solución óptima de cualquier problema pero a pequeña escala, por eso 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 dinámica, programación lineal entera mixta y algoritmos de Branch & Bound.

El primer trabajo en el desarrolló un método para determinar la operación eficiente de redes de gas natural realizado por Wong & Larson, 1968. En dicho trabajo usaron herramientas matemáticas de programación dinámica para resolver problemas de optimización en redes sencillas de gas natural. Señalan que las principales ventajas de la programación dinámica es que garantizan el cálculo de un óptimo global y que la no linealidad del problema puede ser fácilmente manejado. Por otro lado, indican que una desventaja que se tiene al usar programación dinámica en este tipo de problemas es su

limitación a redes con estructuras sencillas, por ejemplo redes tipo línea serie y tipo árbol, además que los cálculos se incrementan exponencialmente según la dimensión del problema. De igual manera, manifiestan que la aplicabilidad de la programación dinámica en problemas con topologías simples obedece al hecho que es posible predeterminar los flujos de antemano, lo cual reduce el problema a encontrar los valores óptimos de presiones; y que en redes tipo malla, esta propiedad no se aplica, por lo que deben considerarse también a las presiones y flujos en el proceso de optimización.

30 años después Tucciarelli y Termini en 1998 consideraron la simulación de la ley de descomposición del cloro en los sistemas de agua potable. En particular, se prestó atención a la identificación de la diversidad espacial de los parámetros de calidad del agua. El criterio para agrupar los parámetros de calidad del agua se basó en una lógica difusa. Se aplicó una técnica numérica integrada, incluidos los códigos interconectados (comercial y / o elaborados de manera adecuada), para determinar la diversidad espacial del cloro residual en una subred en la ciudad de Palermo (Italia). El procedimiento incluyó el código EPANET para la estimación de los caudales, la presión del agua y el transporte de sustancias en el agua de la red (módulos hidrodinámicos y de calidad del agua) calibrar sus parámetros desconocidos. Los autores mencionan que la ventaja del procedimiento propuesto en comparación con otros propuestos en la literatura es que le permite al usuario identificar automáticamente los niveles de cloro en cualquier zona de la red sin ningún problema (Tucciarelli & Termini, 1998).

Sin embargo, Aybar, 2013 su metodología empleada consistió en desarrollar modelos matemáticos integrados de los sistemas de transporte y distribución de gas seco, a través del uso de ecuaciones fundamentales que gobiernan el funcionamiento de los fluidos compresibles; dichos modelos incluyeron variables de tipo operativo y económico. Uno de estos modelos permite la

manipulación de las variables independientes para calcular valores de consumos de combustibles totales, mientras que otro calcula directamente los valores óptimos de presión y flujo. Mediante el modelo elaborado, demostró que a través del control adecuado de parámetros operativos (presiones y volúmenes demandados) se optimiza el funcionamiento de la red principal de transporte y de distribución de gas natural seco del proyecto

Por medio del programa elaborado en el software GAMS, simuló el funcionamiento de la red principal de la instancia de prueba real. El algoritmo desarrollado ha sido aplicado a una amplia gama de escenarios de flujos, obteniendo resultados de presiones en el orden de las reales que están disponibles en la data de los concesionarios de transporte y distribución, y que se sitúan dentro de los dominios de operación de las estaciones de compresión (Aybar, 2013).

El problema de programación lineal entero mixto fue utilizado por (Dai & Li, 2014) los cuales presentaron un enfoque de reformulación para identificar las posiciones óptimas de las válvulas reguladoras de presión en un sistema de distribución de agua. El problema de programación lineal entera mixto se reformula como programación matemática con restricciones complementarias que puede resolverse de manera eficiente por los algoritmos de programación no lineal. El problema de Programación lineal entera mixta se adapta a una Programación no lineal, utilizando un método de castigo que se resuelve por una secuencia de problemas de Programación No Lineal.

En el mismo año se presenta (Puleo *et al.,* 2014) una metodología basada en la programación lineal para determinar el programa de bombeo óptimo en 24 horas, considerando como variables de decisión las tasas de flujo continuo de la bomba que posteriormente se transforman en un programa discreto. La metodología se aplicó en un estudio de caso derivado

de la red de referencia de una cuidad. Para evaluar la confiabilidad de programación lineal, se realizó una comparación con las soluciones generadas por un algoritmo de búsqueda de dimensiones dinámicas discretas. Se demostró que el costo asociado con el resultado derivado de la solución inicial de programación lineal fue menor que el obtenido con soluciones iniciales sin la programación lineal aun con semillas diferentes. En este trabajo, el problema de programación de la bomba se realizó a través de un método basado en programación lineal. La metodología de la programación lineal demostró, por lo tanto, la posibilidad de ser adoptada para determinar rápidamente una aproximación, aunque es aceptable, solución estás optimizada.

Otro autor en el mismo año (Ostfeld, Oliker, & Salomons, 2014) presenta un nuevo enfoque para la identificación de fuentes de contaminación en sistemas de distribución de agua a través de un modelo de árboles acoplados: algoritmo de programación lineal. Los modelo de árboles son una extensión de los árboles de regresión (árboles de regresión: modelos basados en árboles utilizados para resolver problemas de predicción en los que la variable de respuesta es un valor numérico) en el sentido de que asocian las hojas con modelos lineales multivariados. Los árboles modelo reemplazan a EPANET a través del aprendizaje (es decir, entrenamiento y validación cruzada), después de lo cual una formulación de programación lineal utiliza la estructura de clasificación de reglas lineales del árbol modelo para resolver el problema inverso de la identificación de la fuente de contaminación. El uso de árboles modelo representa el modelado hacia adelante (es decir, desde la raíz hasta las hojas). La implementación de la programación lineal en la estructura de árbol lineal permite el modelado hacia atrás (inverso) (es decir, de las hojas a la raíz) donde las características de las inyecciones de contaminación son las incógnitas del problema. La metodología propuesta proporciona una estimación del tiempo,

la ubicación y la concentración de las fuentes de inyección de contaminación. El modelo aplicó a dos casos de prueba teóricos.

En el 2015 Termini & Viviani lograron una reducción en la concentración de cloro, que se utiliza como desinfectante químico para el agua en los sistemas de distribución de agua potable, puede considerarse un índice del deterioro progresivo de la calidad del agua. En este trabajo, se prestó atención a la distribución espacial del cloro residual en los sistemas de distribución de agua potable. El criterio para agrupar los parámetros de calidad del agua normalmente utilizados es altamente subjetivo y, a menudo, se basa en datos que no están correctamente identificados. En este documento, se aplicó un análisis de clúster basado en lógica difusa. La ventaja del procedimiento propuesto es que permite a un usuario identificar (de forma automática y clara) la zonificación de la red y calibrar fácilmente los parámetros desconocidos. Se ha utilizado un análisis de la correlación entre los sitios de muestreo para el cloro residual como para evaluar la aplicabilidad del procedimiento (Termini & Viviani, 2015).

En muchas redes modernas de agua, una tendencia urgente es medir la presión en varios puntos de la red por razones operativas (Berglund *et al.,* 2017), esto debido a que las fugas generalmente induce en la presión establecida, estas mediciones de rutina pueden usarse para desarrollar enfoques de detección de fuga. Esta investigación empleó métodos de aproximación lineal sucesivos, basados en programación lineal y programación lineal de enteros mixtos, en un marco de simulación-optimización para explorar una metodología alternativa de detección de fugas para redes de distribución de agua urbana basada en mediciones de presión. Los métodos intentan minimizar las diferencias absolutas entre los valores de presión observados y simulados en los sensores para determinar una combinación lineal de fugas que se aproxime más al patrón de presión observado. Se usaron modelos casos de prueba teóricos ya establecidos y

casos de prueba real como pequeñas redes publicadas a una red de 27,000 nodos para una utilidad de los EE. UU. Los resultados muestran que los métodos funcionan bien cuando se dispone de datos de presión general y modelos hidráulicos que representan condiciones operativas reales. Los métodos desarrollados en este trabajo no pretenden reemplazar los métodos tradicionales de detección de fugas; más bien, están destinados a trabajar en conjunto con los métodos disponibles para aislar con mayor precisión y eficiencia las ubicaciones de las fugas y reducir la pérdida de agua.

Sin embargo Wang *et al.,* 2017 proponen el control predictivo del modelo económico no lineal y utilizan el modelo hidráulico no lineal de la red de distribución de agua, además proponen la de programación de bombeo a través del software de EPANET. Wang *et al.,* 2017 Al igual que en otros trabajos, el principal objetivo operacional de las redes es la minimización de los costos asociados con el bombeo y el tratamiento del agua, al tiempo que garantiza el suministro de agua con los flujos y presiones necesarios en todos los nodos de control / demanda en la red. Los autores mencionan que usualmente se buscan otros objetivos operativos relacionados con la seguridad y confiabilidad como es determinar la secuencia óptima de llenado / vaciado de los tanques teniendo en cuenta que el precio de la electricidad varía entre el día y la noche y que la demanda también sigue un patrón repetitivo de 24 horas. Finalmente, se los autores muestran los resultados de la simulación en circuito cerrado de la aplicación de la estrategia de control propuesta a la red de agua de una red de prueba real.

2.4. Solución al problema de Redes de distribución de fluidos con heurísticas.

A diferencia de los métodos exactos los métodos heurísticos encuentran resultados óptimos en tiempo razonables. Por esta razón son

muy usadas en problemas de tipo NP, sin embargo debido a su sencillez al aplicarlas se usan también para problemas de tipo P. A continuación se presentan algunos trabajos relacionados con heurísticas aplicadas al problema de redes de distribución de fluidos.

Por ejemplo (Borraz, 2004), para encontrar soluciones de buena calidad al problema de optimización de redes de gas natural, el autor propone un método de búsqueda más efectivo basado en técnicas no tradicionales de programación no lineal, que el caso de la técnica de programación dinámica no secuencial o la técnica de reducción, Igualmente, propone la implementación de una heurística de búsqueda denominada tabú, como promesa efectiva para obtener soluciones aproximadas de buena calidad de transmisión de gas natural, integrando todas estas técnicas dentro de una metodología de solución como parte de un esquema refinado de optimización, la cual se divide básicamente en cuatro fases: fase de pre-procesamiento, fase de asignación de flujos, fase de solución óptima para las variables de presión y fase de ejecución del procedimiento heurístico, esta última fase está desarrollada para reflejar la optimización tanto en las variables del flujo másico como en las variables de presión, ya que se aplica una heurística de búsqueda tabú que emplea una técnica de programación dinámica no secuencial y un esquema de memoria corta.

En cambio en el trabajo de Oteiza *et al.,* 2015 compararon dos metaheurísticas en un conjunto de redes de una cuidad, los autores usaron el recocido simulado y los algoritmos genéticos, los resultados al ser comparados demuestra algoritmo genético mejor desempeño que el algoritmo simulado, además, abordaron el problema del diseño de tuberías para transportar gas natural líquido, a través de un país sometido a una economía inestable. Oteiza *et al.,* 2018 presentan una técnica de optimización hiperheurística para reducir los tiempos computacionales para encontrar una solución del diseño de redes de tuberías. La estrategia

propuesta es un enfoque de equipo A, que comprende la ejecución guiada de tres meta-heurísticas: un algoritmo genético, un recocido simulado y una optimización de colonias de hormigas. Además, se definió un mecanismo de aprendizaje especializado para el intercambio de información con el fin de acelerar el proceso de búsqueda. Para evaluar el diseño algorítmico los autores emplearon instancias de prueba reales para evaluar el impacto de la función de aprendizaje. Las instancias que ocuparon los autores corresponden a infraestructuras marinas reales que se ubican en la plataforma marina argentina. Concluyeron que la hiperheurística demostró ser competitiva, ya que es capaz de explorar un amplio espacio de búsqueda de forma rápida y ofrecer soluciones satisfactorias.

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 y las instancias de prueba teóricas en donde se realizaron las pruebas fueron las de Alperovits y Balerma.

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 perdidas de fugas de agua en sistemas de distribución de agua (Reis *et al.,* 1997; Araujo *et al.,* 2006; Cattafi *et al.,* 2011; Dai, 2014; Ali, 2015; Pecci *et al.,* 2015).

Diversos autores (Ali, 2015; Baños *et al.,* 2010) han trabajo en algoritmos evolutivos para el diseño de redes de distribución de agua, considerando obtener el menor costo y la mejor configuración de los diámetros de las tuberías. Y otros han presentado este mismo algoritmo

para 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 (Liberatore & Sechi, 2009; Nicolini, 2009; Liberatore, 2009; Hurtado-Guzmán, 2009).

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.

Algunos lo hicieron con instancias de prueba reales (Saldarriaga & Salcedo, 2015) la red de la subsección 8-04 de Bogotá, Colombia. (Kang & Lansey, 2010) algunos sectores hidráulicos del poniente del Distrito Federal, México; (Saldarriaga & Salcedo, 2015) ubicada en Bogotá, Colombia; Darvini & Soldini, 2015 en Chiaravalle, Italia, todos ellos con instancia medianas o pequeñas. Diferentes trabajos han tratado el rendimiento de estas técnicas en redes teórica de tamaño pequeño y mediano, pero pocos de ellos han trabajado con redes teórica de gran escala superior a 443 nodos y 454 tuberías (Martinez-Bahena, 2016).

En el trabajo de Lei *et al.,* 2011 se propuso 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.

Y para otro fluido diferente al agua, Kang & Lansey, 2010 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 usando válvulas mejorar la calidad del agua. El problema de diseño

de la red de distribución de agua tiene que ver principalmente con encontrar los tamaños de tubería óptimos que brinden el mejor servicio a un costo mínimo.

Años después De Paola *et al.,* 2016 estudiaron un modelo para el ajuste de válvulas en redes de distribución de agua, con el objetivo de reducir el nivel de fugas. El enfoque se basa en el algoritmo de optimización de búsqueda de armonía. Este algoritmo imita un proceso de improvisación de jazz capaz de encontrar las mejores soluciones, en este caso correspondiente a la configuración de la válvula en una red. El modelo también interactúa con la versión mejorada de un simulador hidráulico popular, EPANET 2.0, para verificar las restricciones hidráulicas y evaluar el rendimiento de las soluciones. Se introducen restricciones en la función objetivo y el modelo se aplica a dos estudios de prueba, los resultados obtenidos en términos de reducciones de presión son comparables con los de los algoritmos metaheurísticos competitivos (por ejemplo, algoritmos genéticos). Los resultados demuestran que el algoritmo búsqueda de armonía sirve para la gestión y optimización de la red de agua.

En cambio Edwin *et al.,* 2017 desarrollaron un algoritmo genético a la aplicación de redes de distribución de agua en la red de Hanoi, con una frente de pareto. Para comprobar la eficacia del algoritmo propuesto para cumplir con las exigencias económicas, y bajo los límites de presión y velocidad según la normativa peruana El empleo de algoritmos genéticos en la solución de las redes de distribución de agua ha permitido lograr obtener un conjunto de soluciones, del cual el diseñador finalmente deberá decidir según su criterio la solución a adoptar, realizándose la elección dentro de un grupo de redes de distribución de agua optimizadas en confiabilidad y costo encontraron que los resultados aceptables mediante la aplicación del MAGMO.

También Kowalska *et al.,* 2018 realizaron simulaciones de cambios de concentración de cloro en una parte seleccionada del sistema de distribución de agua real. El modelo hidráulico calibrado, que consta de 1092 nodos y 1332 tuberías, Las concentraciones de cloro estuvieron entre 0.01 a 5 mg/l en los puntos característicos de la red de agua elegidos se estimaron por campo Kowalska *et al.,* 2018. Oliveira *et al.,* 2018 utilizan la topología de la Red que consta de 15 nodos, 23 tuberías y 3 fuentes de suministro. La generación de datos para este estudio, se realiza por un período de 8760 horas (equivalente a un período de un año). EPANET se utilizan para diseñar ataques químicos hipotéticos. Ambos están integrados en la programación. Usaron redes neuronales para predecir el comportamiento de dispersión de este contaminante. Las simulaciones fueron en largo período del comportamiento hidráulico y calidad del agua. El fluido utilizado fue cloro se agregó en tres depósitos cada 12 horas de simulación, con una concentración de 1,5 *mg/l*, el contaminante es agregado constantemente en diferentes nodos a una concentración de 12.4 *mg/l*.

En otro trabajo, el papel se centró en la programación óptima de una estación de bombeo de drenaje, cumpliendo con las variaciones en la velocidad de rotación de la bomba y un patrón recurrente para la descarga de entrada. El papel estuvo estructurado en varios pasos consecutivos. En el primer paso, se describió la configuración experimental y se presentaron los resultados de las pruebas de calibración en diferentes máquinas de bombeo para obtener ecuaciones que vinculan variables significativas (descarga, carga, potencia, eficiencia). Luego, esas ecuaciones se utilizaron para construir un modelo de optimización de enteros mixtos capaz de encontrar la solución de programación que minimice la energía de bombeo requerida. El modelo se resuelve con un sistema de drenaje urbano en Nápoles (Italia) y los resultados de la optimización se analizan para proporcionar información sobre el rendimiento computacional del algoritmo y sobre la influencia de las

características de la máquina de bombeo en el ahorro de eficiencia general. Con referencia a los escenarios simulados, se puede ahorró un valor promedio de 32% de energía con un control optimizado. Su valor real depende de las características hidráulicas del sistema (Fecarotta *et al.,* 2018).

(Lee *et al.,* 2018) En este estudio, un algoritmo de optimización metaheurística inspirado en un procedimiento de corrección de la visión se aplica a problemas de ingeniería civil. El algoritmo de corrección de la visión (VCA) tiene la capacidad de resolver varios problemas relacionados con las funciones matemáticas de referencia y la ingeniería civil. Los procesos de corrección de la visión tienen tres pasos principales: corrección miope / hipermetropía, ajuste de brillo / cumplimiento de la compresión y corrección astigmática. Este procedimiento es esencial para aumentar la facilidad de uso de las gafas y obtener una visión de alta calidad en humanos. A diferencia de los algoritmos meta-heurísticos convencionales, VCA ajusta automáticamente la probabilidad de búsqueda global / local y la dirección de búsqueda global en función de los resultados de optimización acumulados. En VCA, todas las variables de decisión tienen sus propias probabilidades de búsqueda y requieren procesos diferentes según se requiera una búsqueda global o una búsqueda local. El algoritmo propuesto se aplica a problemas de optimización representativos, y los resultados se comparan con los de los algoritmos existentes. En problemas de ingeniería civil, incluido el problema de diseño de la red de distribución de agua, VCA muestra resultados respetables en comparación con los algoritmos existentes. En todos los problemas de referencia y problemas de ingeniería civil, VCA muestra buenos resultados y mostró la aplicabilidad a otros problemas de ingeniería civil.

Muchos investigadores han desarrollado diferentes enfoques para el diseño óptimo de redes de tuberías de suministro de agua, pero

Chandramouli en el 2019 desarrolló un caso de diseño detallado que utiliza el software de PANET con algoritmos genéticos en el MATLAB para un diseño óptimo basado en confiabilidad de redes de tuberías de suministro de agua, menciona que a pesar de que mucho científicos han trabajado en este tema hay pocas redes de referencia de alta calidad disponibles para la prueba de algoritmos, lo que, a su vez, dificulta las pruebas profundas de algoritmos, el análisis de sensibilidad y la comparación de las técnicas desarrolladas debido a esto desarrollo una herramienta para generar casos de prueba virtuales. La herramienta, llamada HydroGen, puede generar diferentes casos de prueba de diferente tamaño y características variables en formato EPANET o GraphML. Su algoritmo utiliza un conjunto de casos reales para generar una extensa biblioteca de redes de pruebas en las que se pueden probar métodos metaheurísticos para la optimización del diseño de redes. Como trabajo futuro proponen la generación a otros formatos de salida otros tipos de redes de distribución tales como redes de distribución de gas, redes de distribución de petróleo o redes eléctricas (Chandramouli, 2019).

En cuanto a las hiperheurísticas, McClymont *et al.,* 2014 en su trabajo investigan un nuevo enfoque hiperheurístico que utiliza la programación genética para desarrollar operadores de mutación para algoritmos evolutivos especializados en una formulación bi-objetiva del problema. Una vez que generaron, los operadores evolucionados se utilizan hasta el infinito en cualquier algoritmo evalutivo en cualquier red lo que mejora el rendimiento. Se demuestra un nuevo método multi-objetivo que desarrolla un conjunto de operadores de mutación para un entrenamiento. Los mejores operadores se evalúan en detalle aplicándolos a tres redes de prueba de complejidad variable. Se realiza un experimento en el que se evolucionan 83 operadores. Los 10 mejores son examinados en detalle. Un operador, GP1, ha demostrado ser especialmente efectivo e incorpora un interesante

aprendizaje específico del dominio (suavizado de tuberías), mientras que GP5 demuestra la capacidad del método para encontrar operadores conocidos y bien utilizados, como un gaussiano.

Zulkifli y Nurani Usan EPANET para la solución de las ecuaciones gobernantes de una red de tuberías, aplicado a una hiperheurística con heurísticas bio inspiradas. Demuestra que los nuevos algoritmos desarrollan soluciones de alta calidad en problemas de optimización de red de distribución de agua de referencia de manera eficiente y pueden proporcionar información importante sobre el espacio de búsqueda de problemas (Zulkifli *et al.,* 2018, Nurani *et al.,* 2015).

Estos dos últimos autores aplicaron una hiperheurística con diferentes heurísticas de bajo nivel basadas en un algoritmo evolutivo, a un sistema de distribución de agua potable (cambiando el diámetro de las tuberías), y usó el software de EPANET para la solución de las ecuaciones gobernantes de una red de tuberías.

El trabajo de Kheiri *et al.,* 2015 está basado en una hiperheurística pero no aplica bombas ni válvulas a la red en cambio Ewald *et al.,* 2008 aplica bombeo a través de toda la red, pero a través de un algoritmo genético para activar las bombas y analizar el desplazamiento del cloro desde los tanques hasta los nodos.

Sin embargo en cuanto a otro combustible, Liu *et al.,* 2008 mencionan que basados en la calidad de la energía, analizan la pérdida y la eficiencia de la energía de todos los componentes del sistema de transporte del oleoducto de crudo. La expresión de la pérdida de energía la determinaron y el modelo matemático del sistema de transporte por tuberías lo establecieron con el objetivo de minimizar la pérdida de energía total del sistema, que combina los dos algoritmos de optimización del algoritmo genético monobjetivo y

multiobjetivo. Los autores Utilizaron un oleoducto externo en el noreste de China como como instancia de prueba real, la heurística la utilizaron para realizar una optimización jerárquica para la disposición de los equipos y los parámetros de operación de cada estación en el sistema de transporte de la tubería. Los resultados encontrados muestran que el costo total de transporte de petróleo y la pérdida total de energía del sistema de transporte por tuberías, después de la optimización mejoraron en comparación con los no optimizados.

Y con respecto a las bombas, Ewald *et al.,* 2008 aplicaron un algoritmo a la asignación de las estaciones de bombeo en un sistema de distribución de agua potable. El algoritmo fue ejecutado en una Grid para un caso de estudio y los resultados se compararon con una versión no distribuida del algoritmo. En los resultados encontraron que el cloro se desplazó de forma adecuada haciendo la activación de las bombas de forma alternada.

A diferencia de estos últimos trabajos esta investigación se va a realizar una hiperheurística para minimizar los costos diseño para una red de fluidos, modificando diámetros, bombas y válvulas.

2.5. Conclusiones del capítulo

Haciendo un análisis al estado del arte, se observa que hasta la fecha no se encuentra una hiperheurística aplicada a problemas de redes de distribución de fluidos, para una instancia superior a 443 nodos y 454 tuberías además de bombas y válvulas. Por lo tanto, en este trabajo se propone obtener la factibilidad y optimización de una red de distribución de fluidos, para ofrecer una solución al problema que se trata en este trabajo, se aplicará una hiperheurística porque representa una técnica potente y robusta

que permite la generalidad de aplicarse a diferentes problemas, independientemente que sean problemas P o NP.

Capítulo 3 Modelo matemático

3.1. Introducción.

En este apartado se presenta el modelo de optimización así como el modelo de satisfacción de restricciones, donde el primero permite evaluar los costos de una solución que la hiperheurística necesita para optimizar. El Modelo de optimización propuesto contempla cada una de las restricciones requeridas para definir las características de la tubería del problema. En cuanto al modelo de satisfacción de restricciones, contiene la formulación del problema de análisis en régimen permanente de sistemas hidráulicos complejos como lo son las redes de distribución de fluidos. Este tipo de modelos permite predecir la respuesta del sistema (caudales internos que circulan a través de sus elementos así como las presiones en los nodos). Ambos modelos son necesarios para que la heurística trabaje, el primero evalúa los costos y el segundo evalúa la factibilidad. A continuación se describen ambos modelos.

3.2. Modelo de optimización

En este trabajo se utilizó el siguiente modelo de optimización, el cual está compuesto por una función objetivo y un conjunto de restricciones, las cuales permiten el adecuado funcionamiento en un sistema de distribución de fluidos. La función objetivo consiste en obtener el costo mínimo que implica agregar nuevos elementos al sistema, como son válvulas reductoras de presión, bombas y tuberías. La función está sujeta a un conjunto de restricciones que permiten validar las soluciones.

$$\min C(d,V,Pu) = \sum_{i=1}^{nd}\sum_{j=1}^{nP} C_{d_i} L_{ij} x_{ij} + \sum_{i=1}^{nV}\sum_{i=1}^{nP} C_{V_i} x_{ij} + \sum_{i=1}^{nPu}\sum_{j=1}^{nP} C_{pu_i} x_{ij} \qquad (1)$$

Sujeto a:

$$\sum_{i=1}^{nd} x_{ij} = 1, \sum_{i=1}^{nV} x_{ij} = 1, \sum_{i=1}^{nPu} x_{ij} = 1 \qquad \forall j = 1,2,..,nP; \qquad (2)$$

$$d_{\min} \le d_i \le d_{\max} \qquad (3)$$

$$V_{\min} \le V_i \le V_{\max} \qquad (4)$$

$$Pu_{\min} \le Pu_i \le Pu_{\max} \qquad (5)$$

$$x_{ij} \in \{0,1\} \qquad (6)$$

Dónde:

x = Es variable de ponderación.

L_i =Longitud de cada segmento de tubería.

nP = Número de tubería de diámetro.

nPu =Número de bombas.

nV = Número de válvulas.

C_{d_i} =Costo de las tuberías.

C_{V_i} =Costo de las válvulas.

C_{pu_i} =Costo de las bombas.

La ecuación *(1)* representa la función objetivo la cual minimiza el costo total de los cambios realizados a la red. Ésta está dividida en tres partes, la primera es la suma de los costos de cada segmento de tubería

multiplicada por su costo, la segunda es la suma del costo de todas las bombas agregadas a cada segmento y finalmente la suma de las válvulas agregadas a cada segmento de tuberías de la red.

Para cada segmento de tubería solo se utiliza un diámetro, una válvula y una bomba. Entonces la restricción *(2)* asegura que la suma de cada elemento (diámetro, válvula y bomba) de cada tubería sea 1, además de que este pertenezca a la lista comercial disponible; en el caso de la restricción *(3)* indica que se tiene que seleccionar dentro del intervalo de un diámetro mínimo y máximo de esa lista comercial, las restricciones *(4)* y *(5)* $V_{\min} \leq V_i \leq V_{\max}$

(4 Indican los mismo que la anterior pero para las válvulas y las bombas comerciales.

La restricción *(5)* es una variable binaria, donde si es igual a 1 indica que se encuentra el elemento y en caso contrario es igual a 0.

3.3. Modelo de satisfacción de restricciones.

El modelo matemático de un sistema de red de tuberías consta de un nodo de conexión (nodo de caudal), es un punto en el que dos o más líneas convergen, también por donde el caudal total es igual a la suma de los caudales que entran menos los que salen. Como en un problema de análisis el consumo, *S*, es un dato del problema, la presión en el citado nodo será la incógnita a determinar. Un nodo de altura piezométrica fija es un punto en el que la presión se mantiene constante. Tal ocurre en una conexión a un depósito de regulación o a una zona de presión invariante. Entonces, se establece las siguientes relaciones:

$$\sum_i Q_{in} - \sum_i Q_{out} = Q_e \qquad (7)$$

$$\sum_c h_f = \sum_c E_p \qquad (8)$$

$$\mathrm{H_{min}} \leq \mathrm{H_i} \leq \mathrm{H_{max}} \qquad (9)$$

La ecuación *(7)* representa la conservación de masa, donde la suma del flujo que entra y sale en un nodo debe ser igual a cero. La restricción indica que el flujo que entra en un nodo, Q_{in}, a, Q_{out}, es equivalente a la demanda Q_e. La ecuación *(8)* representa la conservación de la energía. La cual indica que la suma de las pérdidas de energía por fricción h_f en cualquier tubería, (c) debe ser igual a cero o a la energía potencial, (E_p) la cual es suministrada por una bomba. La restricción *(9)* se refiere a los requisitos de presión mínima y máxima que satisfacen a los usuarios del agua, garantizando al mismo tiempo Operación apropiada de la red. La restricción requiere que la presión en un nodo *Hi* sea mayor que la presión mínima requerida $H_{\min} = 30$ *mca* (30 *metros columna de agua*) y más baja que la presión máxima $H_{\max} = 1750$ *mca* (1750 *metros columna de agua*) (PEMEX, 2006).

El software empleado en este trabajo tiene implementado el método del gradiente. El método del gradiente ha demostrado robustez y adquirido una notable aceptación entre la comunidad científica, tanto que en la actualidad es, probablemente, el método de solución más empleado en el análisis de redes. Adicionalmente, es el algoritmo más popular de los paquetes de cálculo, como el software EPANET (Rossman, 2000) el cual será usado en este trabajo. De acuerdo a las ecuaciones *(7)*, *(8)* y *(9)* planteadas en el apartado anterior dichas ecuaciones son válidas para cualquier fluido siempre y cuando cumpla con las siguientes

consideraciones: estado permanente, fluido newtoniano, isotérmico, desarrollado e incompresible, esta última consideración no implica que la densidad sea constante sino que la densidad esté sólo en función de la temperatura y de la concentración, para flujos de líquidos, la compresibilidad puede despreciarse y en el caso de flujo de gases, sólo si el número de Match está por debajo de 0.3 (White, 2009). En el presente trabajo, las velocidades del aire son menores a ese número, de tal manera, que puede ser considerado el fluido como incompresible para nuestro análisis.

3.3.1. Componentes del sistema físico.

Los principales elementos que forman parte de una red hidráulica y que tienen relevancia en el problema de análisis estático son: tuberías, depósitos, válvulas y bombas (Saldarriaga, 2010). Todos estos elementos son en este tipo de análisis considerados estáticos y, por tanto, su comportamiento se describe a través de relaciones algebraicas. Las tuberías y válvulas son elementos resistentes (disipan energía) y las bombas elementos transformadores cuya misión es proporcionar energía de presión adicional al fluido (en función del caudal trasegado). Por otra parte, los depósitos constituyen también fuentes de alimentación de presión al sistema (elementos fuente de presión ideal). Por este hecho, en el análisis estático, los depósitos son considerados nodos del sistema.

Los consumos se consideran en los nodos del modelo y el comportamiento hidráulico global puede ser descrito mediante una ecuación constitutiva algebraica, o sea, por una relación entre el caudal circulante y la diferencia de alturas piezométricas entre los puntos citados extremos.

Una vez agrupados todos los elementos del sistema en las correspondientes líneas, los extremos de cada línea serán nodos del sistema. De este modo un sistema de distribución, cualquiera que sea su

nivel de complejidad, puede esquemáticamente ser representado, a efectos de análisis hidráulico, por un conjunto de “nodos” y “líneas” ver *Fig. 1*. El fluido fluye a lo largo de las líneas y entra o sale del sistema en los nodos (a través de los depósitos o como fuentes externas de caudal, $S=SQ$). Este esquema, que constituirá la base topológica del modelo, deberá representar con suficiente aproximación el conjunto de todas las líneas y sus conexiones.

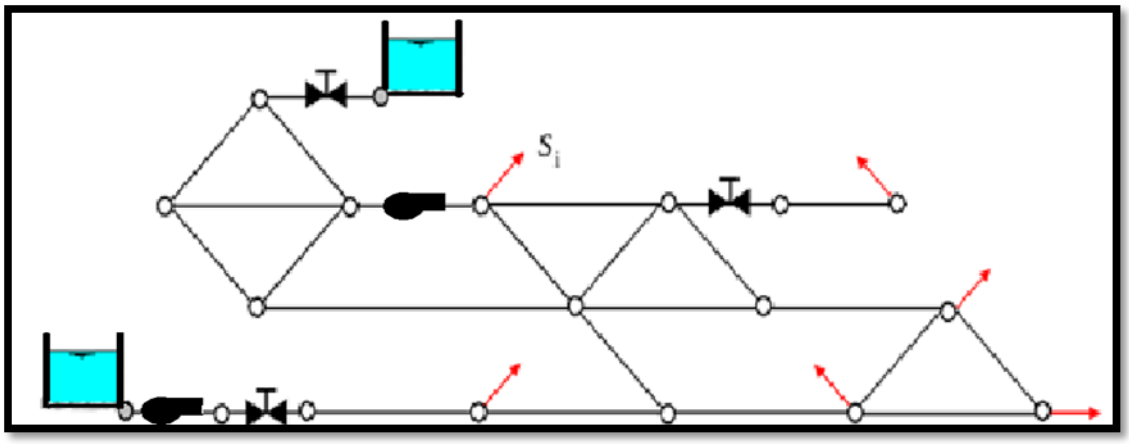


Fig. 1 Esquema físico de un sistema de distribución de un fluido

El tratamiento matemático que en el modelo se le da a un nodo, puede ser de conexión (o nodo de caudal) o de altura piezométrica fija (o nodo de presión). En lo que sigue se describe el modelo matemático del sistema en este estudio.

3.3.2. Ecuaciones gobernantes para un sistema de distribución de un fluido.

Un sistema de tuberías consta de un nodo de conexión (nodo de caudal), es un punto en el que dos o más líneas confluyen o también por donde un caudal puede entrar o salir del sistema. Como en un problema de

análisis el consumo, *S*, es un dato del problema, la presión en el citado nodo será la incógnita a determinar. Un nodo de altura piezométrica fija es un punto en el que la presión se mantiene constante. Tal ocurre en una conexión a un depósito de regulación o a una zona de presión invariante. En éstos, la incógnita es por regla general el caudal externo que aportan o consumen del sistema.

$$M = L - N + 1 \qquad (10)$$

Identificadas las L líneas y los N nodos del sistema ($N=NJ+NF$, siendo NJ el número de nodos de conexión y NF el número de nodos de altura piezométrica fija), el número M de mallas elementales cumple la relación, a partir de la teoría de grafos, bien conocida: y que resulta válida para cualquier tipo de red. Por ejemplo, para el sistema presentado en la *Fig. 1* se tiene N=20 y L=26, por lo que el número de mallas elementales del sistema es: $M = L - N + 1 = 26 - 20 + 1 = 7$.

Independientemente de la forma de interconexión del sistema y del tipo y características propias de los elementos que lo configuran, el estado estacionario responde a dos principios básicos que dan pie a otras tantas ecuaciones: la de continuidad *(7)* y la de energía *(8) si d*e este modo para cada uno de los N nodos "i " del sistema se deberá verificar la ecuación de continuidad que se expresa de acuerdo:

$$\sum_{j=1}^{n} Q_{ij} = S_i \qquad i = 1,2,...,N \qquad (11)$$

Donde el índice *j* referencia todos los nodos conectados directamente al nodo i en tanto que N corresponde al número total de nodos del sistema. A la expresión anterior hay que asignarle un criterio de signos ver *Fig. 2*. Consideraremos un caudal interno, Q_{ij}, positivo cuando el fluido circula del

nodo *i* al nodo *j* (caudal divergente del nodo) y negativo en el caso contrario (caudal que converge en el nodo). Para los caudales externos Si se adopta el criterio opuesto de manera que los consumos son siempre negativos.

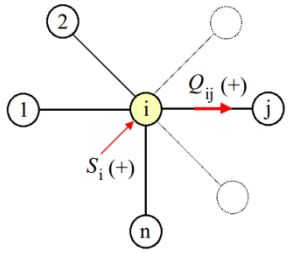


Fig. 2 Criterio de signos para la ecuación de continuidad en un nodo genérico i.

Si N es el número de nodos del sistema, podremos construir un sistema de *N* ecuaciones del tipo *(11)*, conjunto al que es posible añadir una ecuación más, la que tiene en cuenta el balance global de caudales para toda la red, que supone exigir el cumplimiento de que "la suma de todos los consumos ha de ser igual a la de aportes" [Saldarriaga, 2010].

Sin embargo, esta expresión se puede obtener asimismo sumando las *N* ecuaciones del sistema. En consecuencia, las ecuaciones que imponen la conservación de masa (continuidad) proporcionan *N* relaciones independientes (de las $N+1$ posibles).

La ecuación de Bernoulli es una forma de expresar el principio de conservación de la energía *(8)* aplicada al flujo de líquido a través de una tubería. En cada punto a lo largo de la tubería de la energía total del líquido se calcula tomando en consideración la energía de líquido debido a la presión, la velocidad, y la elevación combinado con las pérdidas de energía de entrada, de salida de energía, y de energía (Menon, 2005)[Menom, 2005]. La energía total del líquido contenido en la tubería en cualquier punto es una constante.

Esto también se conoce como el principio de conservación de la energía. En términos genéricos, la ecuación de energía o ecuación de Bernoulli es aplicable entre dos secciones rectas cualesquiera del sistema, en particular, entre los extremos de cada línea del modelo o a un conjunto de ellas integradas en una malla. Establece que la diferencia de energía (alturas piezométricas) H_i - H_j es igual a las pérdidas por rozamiento y pérdidas menores más la energía añadida al flujo a través de bombas. Lógicamente H_{ij} representa la pérdida de carga total y H_{bomba} el aporte total de energía en el trayecto *i-j* por medio de una bomba, la cual se puede representar con la función característica de la misma.

$$H_i + H_{bomba} = H_j + \Delta H_{ij} \qquad (12)$$

Ahora bien, tanto para los elementos bombas como para los elementos resistentes se pueden establecer unas leyes de comportamiento propias que relacionan el caudal circulante con, respectivamente, la altura manométrica y la pérdida de carga – ecuación característica o constitutiva del elemento -. La combinación de esta ecuación con permite sustituir H_{ij} o H_{bomba} por una función del caudal.

$$H_i + H_f = f\ (Q_{ij}) \qquad (13)$$

Si la línea únicamente dispone de elementos resistentes (tubería, válvulas, y otros.) es posible escribir como:

$$H_i - H_j = \Delta H_{ij} = R_{ij}\, Q_{ij} \left| Q_{ij} \right| \qquad (14)$$

En donde el término R_{ij} representa aquí un coeficiente de resistencia hidráulica global de la línea *i-j*, dado por *(7)* para un elemento de tubería de acuerdo a la ecuación de Darcy - Weisbach *(8)* y por para dado por un

elemento válvula o cualquier otra pérdida de carga singular.

$$\frac{8 f_{ij} L_{ij}}{g \pi^2 D_{ij}^5} \qquad (15)$$

$$\frac{8 k_{ij}}{g \pi^2 D_{ij}^4} \qquad (16)$$

Si la línea corresponde a un elemento bomba y/o válvulas especiales la ecuación *(6)* tiene que ser generalizada, puesto que la pérdida de carga que estos elementos provocan depende no solamente del caudal que los atraviesa, sino de otras variables como, por ejemplo, las propias alturas piezométricas en sus extremos. De tal manera que se puede escribir:

$$H_i - H_j = -H_{bomba} = -\left(A + BQ + CQ^2\right) \qquad (17)$$

En la formulación matemática del problema de análisis estático para redes se establece un sistema de ecuaciones de forma que el número de incógnitas iguale al número de ecuaciones independientes. Se han propuesto distintas formulaciones para la resolución del problema, que fundamentalmente difieren entre sí en el tratamiento del sistema de ecuaciones. Las dificultades típicas provienen de la no linealidad y del gran número de ecuaciones que, en general, hay que resolver. Para simplificar la exposición se admite que el sistema se compone exclusivamente de líneas que se modelan mediante *(8)*. Estas ecuaciones se combinan con las ecuaciones de continuidad *(7)* para formar el conjunto de *N+L* ecuaciones que sigue: de la ecuación (11) Para $N = NJ + NF$ ecuaciones y para la ecuación *(14), p*ara L ecuaciones

Si contamos con un sistema de este tipo en el que dejamos como incógnitas los L caudales internos que circulan por las líneas, las *NJ* alturas

piezométricas en los nudos de conexión (las NJ demandas serán dato) y los NF caudales externos aplicados en los nudos de altura conocida, el número total de incógnitas (N+L) es igual al número de ecuaciones independientes y la solución existe y es única. No obstante, en general, no es necesario resolver de manera simultánea la totalidad de las ecuaciones anteriormente planteadas. Con respecto a la unicidad de la solución, conviene subrayar que para que el problema tenga efectivamente una única solución es necesario que al menos exista un nudo de altura piezométrica conocida (o que esta sea una función conocida del caudal externo).

De otro modo sería posible conocer la diferencia de alturas piezométricas entre cada par de nudos del sistema pero no su altura piezométrica. O dicho de otro modo, existirían infinitos valores de alturas piezométricas en los nudos que cumplirían con las condiciones del problema.

De acuerdo con el juego de datos/incógnitas considerado, en cada nudo hay dos variables (H_i,S_i), de las que una será dato y la otra incógnita. Así, las NJ ecuaciones de continuidad del sistema *(11) que parte de la ecuación (7)* forman un sistema con L incógnitas (los caudales internos, Q_{ij}), puesto que el conjunto de variables a resolver sólo contiene caudales. Por otra parte, por cada nudo adicional de altura piezométrica conocida se obtiene una nueva ecuación de continuidad, pero también una nueva incógnita: el caudal de nudo Si, por lo que el número de incógnitas caudal del conjunto de las *N* ecuaciones de continuidad pasará a ser L+NF.

Este hecho permite en el caso de redes ramificadas con un único nudo de altura piezométrica conocido, como se verá a continuación, desacoplar las *NF* ecuaciones de continuidad del sistema *(10)*. Para una red ramificada se cumple $L = N - 1$. Si, además, el sistema ramificado contiene un único nudo de altura piezométrica conocida, las $NJ = N$-1 ecuaciones de continuidad correspondientes a los nudos de conexión (y que constituyen un sistema de

ecuaciones lineales), pueden ser resueltas independientemente del sistema de ecuaciones energéticas, puesto que ellas solas permiten determinar los caudales circulantes por cada una de las $L=N-1$ líneas del sistema a partir de los datos de consumo en los nudos de la misma. A su vez, el caudal externo, Si, aportado por el depósito puede ser determinado directamente mediante la ecuación de continuidad o, alternativamente, a partir del balance global de consumos y aportes *(7)*.

Así, desde un punto de vista del cálculo hidráulico, se puede describir una red ramificada como aquella en que la determinación de los caudales que circulan por las líneas se puede realizar sin conocer las características de las propias líneas. Basta con conocer los consumos de la red y la conectividad del sistema. Dado que las características hidráulicas de las tuberías y bombas son conocidas, sustituyendo los caudales obtenidos anteriormente en el sistema de ecuaciones de energía, constituido por $L=N-1$ ecuaciones del tipo *(8)*, se obtienen finalmente, empezando por la altura especificada en el nudo de altura piezométrica conocida, las alturas piezométricas en los $NJ = N-1$ nudos de conexión del sistema.

Por contraposición, la distribución de caudales en las líneas de una red mallada sí depende de las características hidráulicas de las mismas. Por otra parte, desde el punto de vista topológico, un sistema con varios puntos de alimentación y sin mallas, dada su estructura arbórea, puede considerarse ramificado, si bien hidráulicamente (desde el punto de vista del cálculo) es un sistema mallado. En definitiva, siempre que un sistema contenga mallas o, alternativamente, siendo ramificado disponga de más de un nudo de altura piezométrica conocida, las NJ ecuaciones de continuidad *(7)* combinadas con las L ecuaciones de línea forman, como veremos a continuación, un sistema acoplado de ecuaciones no lineales que permiten determinar los L caudales internos de las líneas así como las NJ alturas piezométricas en los nudos de conexión. El cálculo de los caudales externos aplicados a cada uno de los

nudos de altura conocida, se efectuará una vez determinados los caudales de línea, aplicando las NF ecuaciones de continuidad.

En sistemas mallados el sistema de ecuaciones de continuidad lo componen N ecuaciones, mientras incluye como incógnitas $L+NF$ caudales de línea ($L>N-1$). Resulta, pues, evidente que aquel sistema no basta para determinar las incógnitas Q_{ij}. En el caso particular de un sistema ramificado con varios nudos de altura piezométrica conocida se cumplirá que $L=N-1$, pero por cada nudo de altura conocida aparece una nueva incógnita: el caudal aportado o consumido en el nudo S_i. El número total de caudales incógnitas en las N ecuaciones de continuidad es, pues, $L+NF=N-1+NF>N$, (dado que $NF>1$), por lo que el sistema formado por las ecuaciones de continuidad no es suficiente.

Sin embargo, en ambos casos, las NJ ecuaciones de continuidad y las L ecuaciones de energía de *(11)* constituyen un sistema de $NJ+L$ ecuaciones algebraicas que permite la determinación de los caudales internos Q_{ij} de las líneas y de las alturas Hi en los nudos de conexión. Tal sistema no puede ser resuelto mediante un método directo, puesto que las ecuaciones de energía son no lineales.

Además de contar con las L ecuaciones de línea y N ecuaciones de nudo formuladas a través del sistema, se pueden plantear M ecuaciones de malla. Estas ecuaciones resultan de aplicar el principio de “conservación” de la energía mecánica *(8)*, a un circuito cerrado. Por ello, la suma algebraica de las pérdidas de carga y de la energía proporcionada al fluido a lo largo de un circuito cerrado debe ser nula.

Ahora bien, como las $L+N+M$ ecuaciones anteriores no son independientes, para resolver el problema de análisis se han venido proponiendo diversas alternativas que difieren en la formulación del sistema

de ecuaciones. Todo ello en aras a reducir su tamaño y dotarlos de mejor convergencia. Nos referimos a los ya clásicos planteamientos de la formulación por nudos (ecuaciones en H), formulación por líneas (ecuaciones en Q) y formulación por mallas (ecuaciones en ΔQ).

Cada una de las alternativas presenta ventajas e inconvenientes. Así, por ejemplo, si bien la formulación por mallas es la que da lugar a un sistema con menor número de ecuaciones (M ecuaciones), exige determinar un conjunto inicial de caudales que satisfagan la continuidad en todos los nudos. Por otra parte, la formulación por mallas (y también por líneas) requiere información adicional en la definición de mallas independientes así como en la formulación de las ecuaciones de malla asociadas.

Tras lo expuesto, y aunque se necesita resolver un conjunto de ecuaciones mayor, la formulación planteada para aplicar el algoritmo del gradiente de (Todini & Pilati, 1987) ha adquirido gran aceptación entre la comunidad científica. En ello tiene mucho que ver el que se genera una matriz de coeficientes que además de ser simétrica tiene alta dispersión, es decir, con una mayoría de entradas nulas, hecho que permite la utilización de técnicas eficientes de reordenamiento, adecuadas para matrices dispersas y que simplifican significativamente el sistema a resolver pues permiten almacenar y operar sólo con los elementos de valor no nulo de la matriz de coeficientes (ITA, 2015).

3.3.3. Método del gradiente.

El software empleado en este trabajo tiene implementado el método del gradiente. El método del gradiente fue desarrollado por los profesores E. Todini y E.P. O'Connell en la universidad de Newcastle upon Tyne y por R. Salgado, en su tesis doctoral en 1982-1983. Todini y Pilati en 1987 planearon la forma definitiva del método (Todini & Pilati, 1987; Saldarriaga, 2010), en el

cual las ecuaciones de energía individuales para cada tubo se combinan con las ecuaciones de masa individuales en cada unión con el fin de obtener un solución simultanea tanto de los caudales en las tuberías como de las alturas piezométricas en los nodos.

El método linealiza las ecuaciones de energía utilizando una expansión de serie de Taylor, este método a diferencia de otros métodos, las ecuaciones se resuelven utilizando un esquema imaginativo que se basa en la inversión de la matriz de coeficientes originales. Este método es de los más utilizados en los programas comerciales y de distribución gratuita en la red.

El método del gradiente es para el cálculo de redes de distribución de aguas está basado en el hecho de que al tener un flujo permanente se garantiza que se cumplan las ecuaciones de conservación de la masa en cada uno de los nodos de la red y la ecuación de conservación de la energía en cada uno de los circuitos de ésta. Las ecuaciones *(7) y (8)* son las ecuaciones de conservación que tienen que cumplirse. Además debe haber relación no lineal entre las pérdidas por fricción y el caudal para cada uno de los tubos que conforman la red.

$$f = \frac{0.25}{\left[\log_{10}\left(\frac{\varepsilon_r}{3.7} + \frac{5.74}{Re^{0.9}}\right)\right]^2} \qquad (15)$$

En el caso de la ecuación *(7)* se utiliza para obtener las pérdidas de carga en la tubería, la cual se puede calcular con varias ecuaciones mostradas en la sección siguiente. La ecuación *(15)* se utiliza para calcular $\sum_c h_f$ de la ecuación *(8)*.

El proceso de solución se puede resumir en los siguientes pasos:

- Se supone unos caudales iniciales en cada uno de los tubos de la red (no necesariamente balanceados, lo cual implica un ahorro de tiempo).
- Se resuelve el sistema utilizando un método estándar para la solución de las ecuaciones lineales simultáneas.
- Se determina el caudal de la iteración $i+1$.
- Con ese valor del caudal se sustituye en las ecuaciones del sistema, para encontrar la altura piezométrica de $i+1$.

El proceso se repite hasta que en dos iteraciones sucesivas se cumpla de la altura piezométrica de la iteración actual tiene un error relativo cercano a cero, este en la literatura indica que si es menor a $1x10^{-6}$, se puede tomar como el valor buscado

3.3.4. Ecuaciones para cálculo de pérdidas de carga en una tubería de EPANET.

El método del gradiente antes descrito es usado por el software de EPANET para dar solución al modelo de satisfacción de restricciones. EPANET es un programa informático desarrollado por la agencia de Protección ambiental de los Estados Unidos (EPA, *Enviroment Proteccion Agency*), para el cálculo de redes hidráulicas, muy sencillo y con una interfaz gráfica para la elaboración y visualización de los elementos de la red. Con esto permite el estudio y análisis del comportamiento de redes de tuberías hidráulicas a presión (EPANET, 2016; EPANET, 1 de octubre 2018).

La primera versión de EPANET fue lanzada hacia el año 1993, La versión original del programa fue desarrollada en inglés por la EPA y ha sido traducida al español por varias instituciones. En España y Latinoamérica una de las de mayor difusión es la desarrollada por la Universidad Politécnica de Valencia.

Es un software libre, que se puede descargar directamente de la página web de la EPA en su versión original en inglés o la versión en castellano del Grupo Multidisciplinar de Modelación de Fluidos GMMF UPV (EPANET, 2016).

Como se mencionó, su ventaja es la sencillez en su uso y su libre acceso, lo cual ha contribuido a su gran popularidad, ya que con él se puede hacer la mayor parte de los cálculos necesarios de un análisis hidráulico. Algunas de sus prestaciones son dimensionado de tuberías, determinar las mejoras y/o ampliaciones que necesita una red, analizar ubicación de elementos como válvulas, bombas y depósitos, entre otras prestaciones. Ya que el programa está compuesto por un módulo de análisis hidráulico que permite simular el comportamiento dinámico de una red de distribución de agua potable. Hace posible incorporar a la simulación tuberías, bombas de velocidad fija y velocidad variable, válvulas de estrangulación, válvulas reductoras y sostenedoras de presión, tranques de cabeza constante o variable y sistemas de control y operación temporales o según nivel y presión.

EPANET permite hacer un análisis hidráulico basándose en el método del gradiente antes descrito y ocupa las ecuaciones *(7), (8) y (9)* en cuanto a la ecuación *(15)* se utiliza para calcular las pérdidas de la tubería, pero esta ecuación presenta variantes ya que existen diferentes métodos para calcularla, a continuación se presentan los métodos más comunes los cuales son usados por EPANET.

- **Hazen-William**s:

La fórmula propuesta forma parte de una serie de expresiones empíricas, de la forma:

$$\frac{h_f}{L} = \frac{R}{D^m} Q^n \qquad (18)$$

$$R = \frac{10.675}{C^m} \qquad (19)$$

Donde:

n= 1.852.

m=4.8704.

L=Longitud de tubería, en *m*.

D=Diámetro interior de la tubería, en *m*.

Q=Caudal circulante, en m^3/s.

C= Coeficiente que depende de la rugosidad y estado de la tubería.

- **Chazy-Manning:**

  La fórmula de Chazy-Manning, se define como:

$$\frac{h_f}{L} = \frac{10.294\mathrm{n}^2}{D^{5.33}} Q^2 \qquad (20)$$

Donde:

n= Coeficiente de rugosidad de Manning.

L=Longitud de tubería, en *m*.

D=Diámetro interior de la tubería, en *m*.

Q=Caudal circulante, en m^3/s

- **Darcy-Weisbach:**

  La ecuación esta expresada en función del caudal y es:

$$\frac{H^f}{L} = \frac{8f}{\pi^2 D^5 g} Q^2 \qquad (21)$$

Donde:

L=Longitud de tubería, en *m*.
D=Diámetro interior de la tubería, en *m*.
f=factor de fricción, adimensional.
Q=Caudal circulante, en m^3/s

Para el cálculo de *f* la expresión matemática de Swamee-Jane (a diferencia de la de Colebrook-White) no precisa iteraciones para calcularla, entonces se expresa de la siguiente:

$$f = \frac{0.25}{\left[log_{10}\left(\frac{\varepsilon_r}{3.7} + \frac{5.74}{Re^{0.9}}\right)\right]^2} \qquad (15)$$

La ecuación anterior se plantea en una sección anterior ya el método del gradiente la utiliza para el cálculo de las pérdidas de carga, sin embargo ésta puede variar dependiendo el método a utilizar.

Para:

$$10^{-6} \leq \varepsilon_r \leq 10^{-2} \qquad (22)$$

El rango de Reynolds es para:

$$4000 \leq Re \leq 10^8 \qquad (23)$$

Siendo la rugosidad relativa:

$$\varepsilon_r = \frac{\varepsilon}{D} \qquad (24)$$

Donde:

ε= es el coeficiente de rugosidad de la tubería.

Y para dado el caso que flujo sea laminar:

$$f = \frac{64}{Re} \qquad (25)$$

Sin embargo solo una ecuación para el cálculo de la pérdida de carga se puede usar y esto es porque permite el cálculo más preciso que las demás, y es la ecuación de: **Darcy-Weisbach,** esto es debido a que se usa para diferentes fluidos y no solo para agua como es el caso de las anteriores ya que son ecuaciones experimentales.

Capítulo 4. Metodología

4.1. Introducción.

En este capítulo se explica la estrategia que se ha utilizado en esta tesis para dar solución al problema de optimización propuesto. Para empezar se enuncian los conceptos de la hiperheurística después se detalla la forma en la se aplicaron las heurísticas de bajo nivel así como la función de aprendizaje. De igual forma se detalla cómo se evalúa la factibilidad es decir se ve si se cumplen el modelo de satisfacción de restricciones para las

presiones en cada tubería y nodo. También se explican las instancias utilizadas, por último se detalla el modo de operar la sintonización de las instancias.

4.3. Metodología de implementación de la hiperheurística.

Para lograr el objetivo de este trabajo se hace la siguiente metodología de forma general para lograrlo.

1. Se aplican estructuras de vecindad para lograr la factibilidad de la solución inicial.
2. Se modifica el software de EPANET para poder evaluar la factibilidad.
3. Entra a la hiperheurística y la función de aprendizaje decide que heurística aplicar.
4. La función de aprendizaje también decide cuando aplicar cruzamiento si los costos de la solución no mejora.
5. Se repiten pasos 2 y 3 hasta logar la solución optimizada y factible.

A continuación se detalla la metodología de este trabajo.

4.3.1. Estructuras de vecindad.

En este trabajo se realizaron 27 diferentes estructuras de vecindad las cuales todas son de inserción es decir se elige una tubería, válvula o bomba y se hacen cambios ya sea por otro valor de diámetro, válvula o bomba. En la *Fig. 3* se muestran las diferentes formas de combinar las intersecciones quitar, poner y/o cambiar, el 1, 2 y 3 indican el número de cambios que se harán a la red.

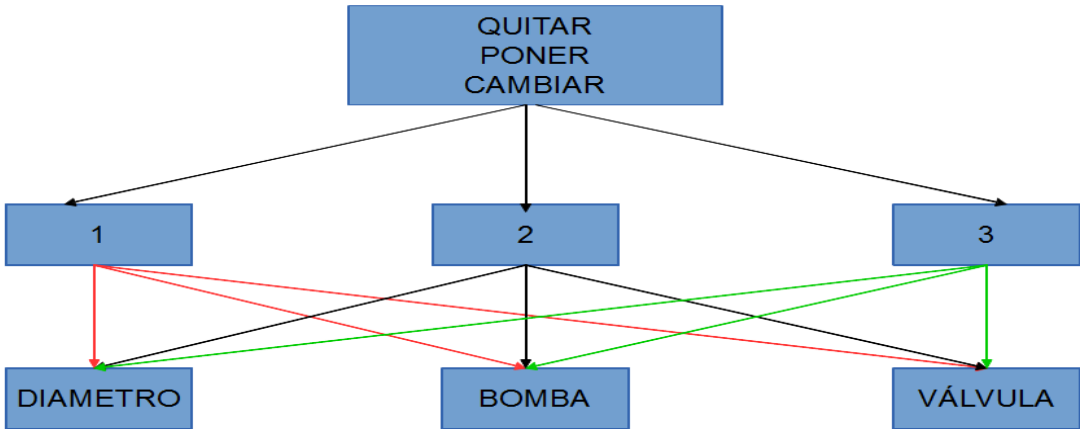


Fig. 3 Estructuras de vecindad con 1,2 o 3 movimientos.

En el caso de quitar no es posible para el diámetro, porque si esto sucede se estaría eliminando la tubería, se estaría eliminando una sección de la red, para las bombas y válvulas lo que se hace es cambiarla por otra bomba de mayor o menor caballaje y las válvulas se cambia el valor del tarado. Las líneas de unión de la *Fig. 3* marcadas con color rojo, verde y negro son 9 en total, pero si combinamos se puede hacer más de un cambio, podrías cambiar un diámetro y bomba o quitar una bomba y poner una válvula, en la se muestra el total de combinaciones que es posible hacer, esto permite la exploración del espacio de soluciones.

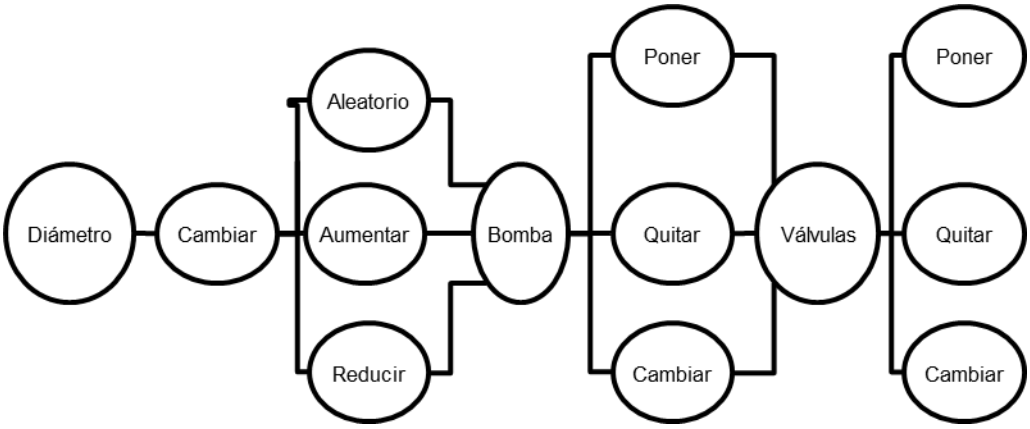


Fig. 4 número total de inserciones

En la *Fig. 4* se muestra gráficamente como se realizaron los cambios, son 3 componentes: diámetro, bomba y/o válvula, en el diámetro: Aleatorio (puede ser que aumente o disminuya), aumenta y/o reduce. En la *Fig. 4* hasta aquí muestra sin embargo no hay que olvidar que son 3 movimientos (1,2 o 3 cambios), es decir 3 x 3 x 3 esto da un total de 27 de estructuras de vecindad.

4.3.2. Consideraciones para utilizar EPANET.

El método del gradiente ha adquirido una notable aceptación entre la comunidad científica, tanto que en la actualidad es, probablemente, el método de resolución más empleado en el análisis de redes. Y es el más utilizado en los paquetes de cálculo, el EPANET (Rossman, 2000). De acuerdo a las ecuaciones planteadas en el apartado anterior dichas ecuaciones son válidas para cualquier fluido siempre y cuando cumpla con las siguientes restricciones:

- Estado permanente: en este estudio sólo interesa conocer el estado final del sistema y no su evolución en el tiempo.
- Fluido Newtoniano: es un fluido con viscosidad en que los esfuerzos son directamente proporcionales al gradiente de velocidad.
- Flujo incompresible: esto no implica que la densidad sea constante, hace que la densidad esté sólo en función de la temperatura y de la concentración, para flujos de líquidos, la compresibilidad puede despreciarse y en el caso de flujo de gases, sólo si el número de Match está por debajo de 0.3 (White, 2004), y en este trabajo las velocidades del aire son bajas, de tal manera, que estos flujos pueden ser considerados incompresibles.
- Isotérmico: es un fluido a temperatura constante.
- Desarrollado: hidráulicamente, aquel que tiene el perfil

de velocidad constante a lo largo de la longitud de un conducto, más allá de lo que se conoce como región de entrada.

La metodología de solución de las ecuaciones del capítulo anterior es el método del gradiente, sin embargo el software solo permite análisis de redes hidráulicas, ya que las propiedades que usa el software es para el agua a 20 ^{o}C (Rossman, 2000), sin embrago presenta la flexibilidad de introducir algunas propiedades de fluidos diferentes como es el la viscosidad cinemática y el peso específico a la temperatura que se desee la simulación (ITA, 2015). Para evaluar si el comportamiendo de dos fluidos es el mismo, se grafican el resultado de la evaluacion de las ecuaciones gobernantes de una red de tuberías con el método del gradiente, observe *Fig. 5*. Por lo tanto se concluye que debido a que ambos fluidos son Newtonianos, su comportamiento es igual por lo que es correcto el planteamiento de la solución, en el software de EPANET.

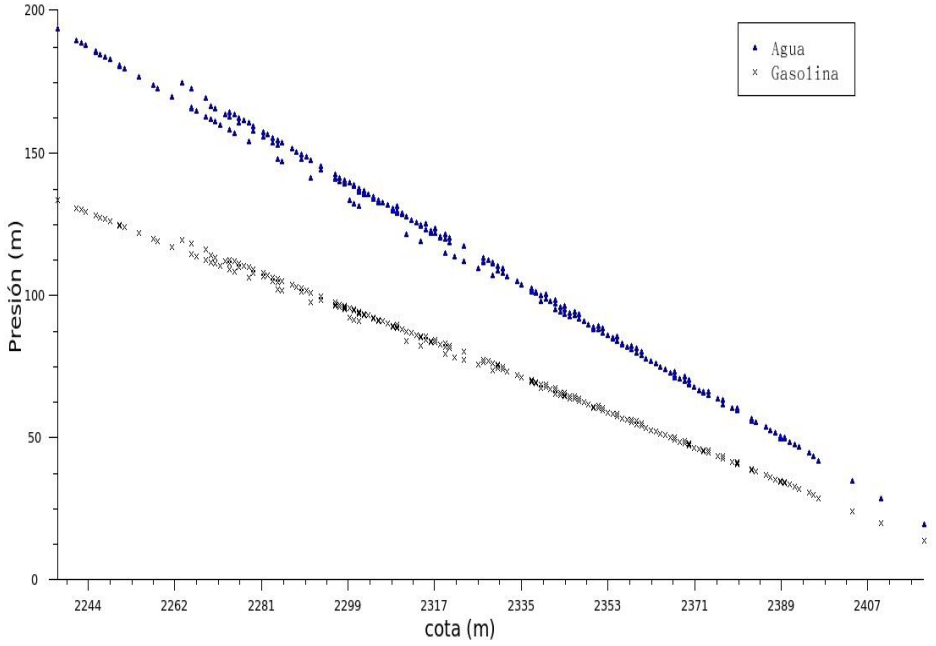


Fig. 5 Comportamiento de dos fluidos newtonianos con diferentes densidades.

En el presente trabajo fue necesario adaptar EPANET para se evaluaran las ecuaciones del fluido a utilizar.

Los resultados muestran que EPANET es un software que al modificar su código, permite simular cualquier fluido líquido newtoniano. Por lo tanto este software sirve para cualquier fluido newtoniano, y debido a que la gasolina pertenece a este grupo, el software de EPANET se puede usar para la gasolina.

4.3.3. Cálculo de las propiedades del fluido.

En la *Tabla 1* se muestran los valores de la red de distribución como son velocidad, viscosidad y densidad del fluido que en este caso es gasolina, y estos valores sirven para modificar las ecuaciones mostrada en la sección 3.3, las cuales están en el paquete comercial EPANET, como ya se demostró en dicha sección, las ecuaciones *(26), (27), (28)* y *(29)* muestran los valores de la gasolina de acuerdo a la *Tabla 1.*

Tabla 1 Los valores del fluido.

Propiedad	Valor
Velocidad	5 *m/s*
Viscosidad	$1.07x10^{5}$ m^2/s
Densidad	870 *kg/m3*

Cálculo de la viscosidad relativa con respecto al agua a 20 *ºC.*

$$\nu_{relativa} = \frac{\nu_{gasolina}}{\nu_{agua}} \qquad (26)$$

Dónde:

$$\nu_{gasolina} = 1{,}07x10^{5}\, m^2/s \qquad (27)$$

$$\nu_{agua} = 1{,}0038x10^{6}\, m^2/s \qquad (28)$$

Al sustituir las ecuaciones *(27)* y *(28)*, en la ecuación *(29)*.

$$\nu_{relativa} = \frac{1{,}07x10^{5}}{1{,}0038x10^{6}} = 10.6594939231 \qquad (29)$$

De igual forma para el peso específico

$$\gamma_{relativa} = \frac{1.32}{1.9345} = 0.68234686 \qquad (30)$$

4.3.4. Ecuaciones modificadas en EPANET.

De acuerdo a la sección anterior, se necesita hacer una modificación al Software de EPANET. Este programa al ser descargado de *(EPANET, 1 de octubre 2018)* se tienen dos carpetas una con el nombre de src la cual contiene los archivos mencionados en esta sección.

Archivo: TYPES.h

Se busca la macro y se cambia al siguiente valor:

#define VISCOS 1.1e-5

O Si se desea cambiar directamente el valor de acuerdo a las ecuaciones *(32)* y *(33)* para estas propiedades se abre el archivo:

Archivo VARS.h

Se buscan las siguientes variables de tipo *float* y se cambia al valor corespondiente:

Extern float Viscos /*viscosidad cinemática relativa*/

Extern float SpGrav /*Peso específico relativo */

4.3. Pasos a seguir para crear la librería estática de EPANET.

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*

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 hiperheurístico, para evitar la configuración de variables de entorno.

IV. Incluir la librería "toolkit.h", dentro del código fuente del algoritmo genético.

El Algoritmo del *anexo 1* lee los datos de la red por medio de EPANET, creando un archivo con la extensión ".inp". Después se lee el archivo mediante la función *ENopen (),* se verifica si el archivo existe, si es así, se procede a cargar los datos en el simulador EPANET con la función *ENopenH ()*, después se obtiene el número total de enlaces y nodos de la red.

4.4. Solución inicial factible.

La *Fig. 6* muestra la metodología para que una solución sea factible, en la cual se recibe una solución, pero no se sabe si lo es, para comprobar la factibilidad se evalúa con EPANET, ya que éste permite evaluar una solución al revisar la presión y caudal en cada nodo. Si no es factible, se hace factible al modificar la red esto se hace al aplicar perturbaciones *Fig. 4* de manera guiada es decir, que solo se aplican cambios en donde las condiciones de presión no se cumplen. Si la presión excede los 1754 *mca*, se pone una válvula, si las presiones son inferiores a 40 *mca*, se coloca una bomba un

nodo antes del que no cumple con esta condición. Cada vez que se realiza un cambio se tiene que evaluar con EPANET, esto es para asegurarse que se cumplan con las presiones. Una vez que se cumplen las presiones en todos los puntos, es decir que ya es factible pero no está optimizada para esto se inicia la hiperheurística.

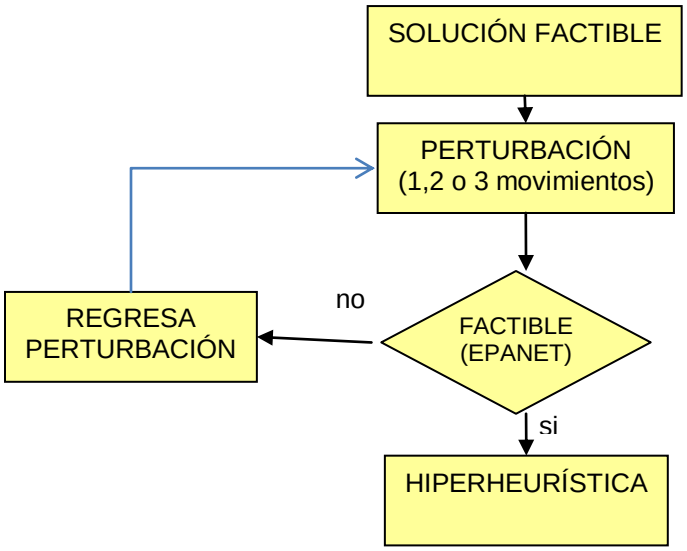


Fig. 6 Diagrama de flujo de la factibilidad

4.5. Función de aprendizaje.

La función de aprendizaje es la que tiene el control de las heurísticas que se van a aplicar, se aplican en cada punto de decisión (Burke *et al.,* 2003), ver *Fig. 7*.

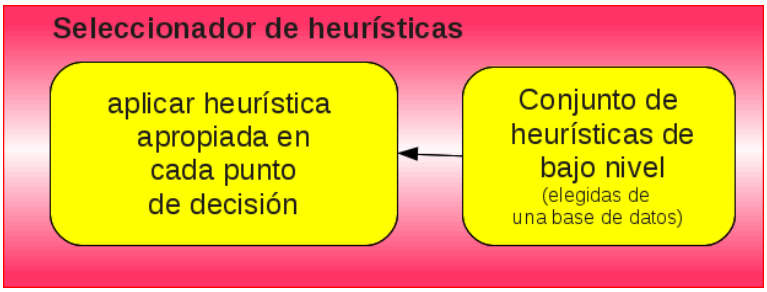


Fig. 7 Seleccionador de heurísticas.

Al principio de la hiperheurística todas las heurísticas se encuentran en una lista donde cada una de ellas tiene la misma oportunidad de ser seleccionada como la búsqueda local clásica, la búsqueda tabú y la búsqueda local iterada excepto el cruzamiento este se aplica después de varios intentos en no mejorar la solución con búsquedas locales, en la *Fig. 8* se puede notar la variedad de heurísticas de bajo nivel que maneja la hiperheurística de este trabajo.

Las heurísticas como son de bajo nivel permiten ya sea la explotación o la exploración, el cual con base en la experiencia permite encontrar soluciones óptimas, sin embargo se tienen que tener heurísticas que permitan la aceptación o el rechazo de soluciones (*Fig. 8*), cabe mencionar que cada vez que genere una solución se tiene que evaluar si es factible o no, mediante el módulo de EPANET.

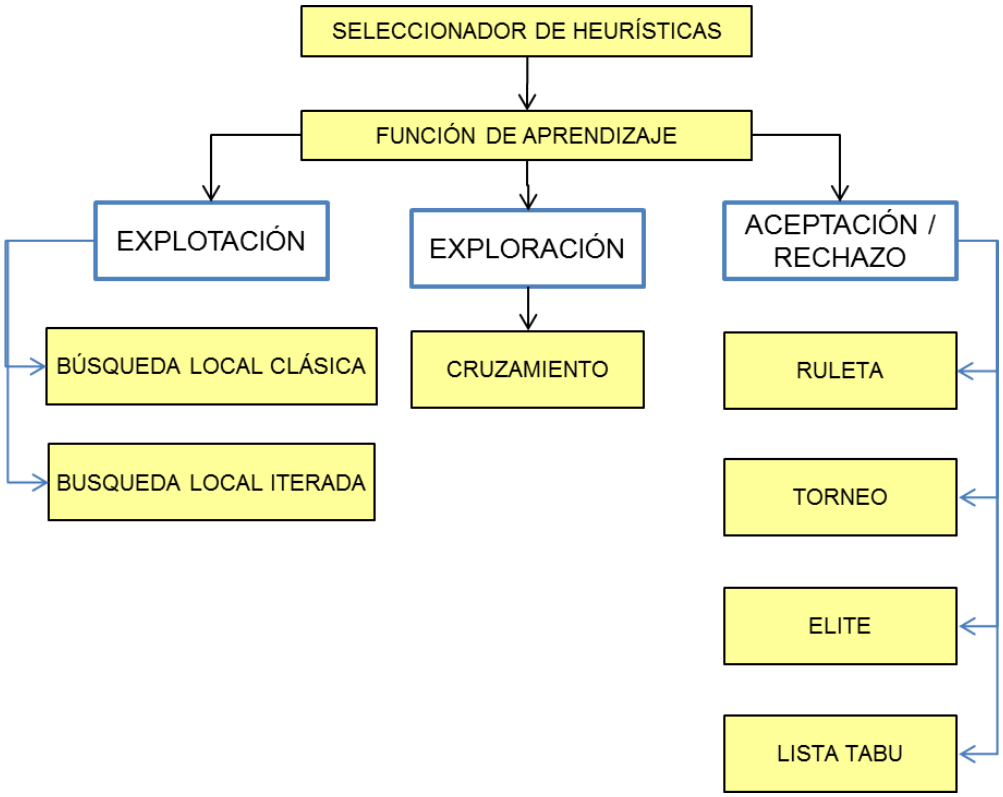


Fig. 8 Representación gráfica de la heurística.

La búsqueda local clásica, la búsqueda tabú y la búsqueda local iterada se van ponderando y organizando en una lista tabú, la cual entre más resultados satisfactorios se obtengan al finalizar la heurística esta se acomoda de la mejor a la peor, y cada vez que la función de aprendizaje se necesita aplicar a alguna se utiliza de forma aleatoria ruleta, torneo o elite, para que ésta elija cual se va a ejecutar. Hay que recordar que cada una de las búsquedas están con las diferentes estructuras de vecindad eso hace que son 27 búsquedas locales, 27 búsquedas tabú y 27 búsquedas locales iteradas dando un total de 81 heurísticas de bajo nivel.

En elite se elige la que mejor dé resultados, en el caso de torneo se toman dos y estas compiten pero la mejor gana, en el caso se ruleta como

cada una tiene un peso la probabilidad de elegir a la peor disminuye, sin embargo puede ser que si se elija.

Para ejecutar el cruzamiento después de varios intentos sin mejorar se aplica. Entonces la función de aprendizaje tienes las siguientes funciones:

- Cuántas veces se cruza.
- Cada cuanto lo hace.
- Qué heurística se aplica

¿Qué heurística se aplica? la función decide que heurística se aplica y todas aquellas que no generan soluciones factibles, se quita de las lista de las posibles a elegir. Esto con ayuda del factor de reducción el cual ya se ha demostrado en otros trabajos que si se elige un camino malo, las solución que de ahí se obtengan serán igual de malas (Kirkpatrick, et. al, 1983) al igual que pasa con el factor de enfriamiento del Recocido Simulado.

4.6. Cruzamiento.

Se elige dos mejores soluciones en donde se encuentran las mejores configuraciones de diámetros, las bombas o las válvulas, para hacer el cruzamiento, el cual se ponen en un vector para cada solución, éstos contienen la información de las posiciones así como su valor, y se divide en dos partes, y se toma la primera mitad del vector con la segunda parte del otro vector y viceversa.

Primera solución

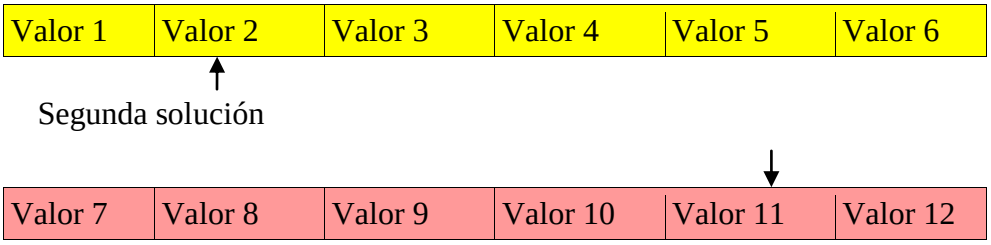


Fig. 9 puntos de cruce.

Aplicando el cruzamiento en los puntos en donde se encuentran las fechas (*Fig. 9*) para generar dos nuevas soluciones.

Solución nueva 1.

Solución nueva 2.

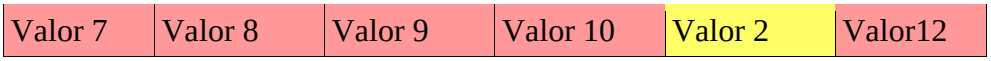


Fig. 10 Dos nuevas soluciones.

Con esto se generan dos nuevas soluciones que se tienen que evaluar si son factibles, si las dos son factibles de forma aleatoria de elije una, si una es factible esa entra a ser optimizada y en todo caso si ninguna de las son factibles se forma aleatoria se elige una y se hace factible.

Entonces cuando ya se tiene una factible se va a explorar nuevamente con alguna búsqueda local que la función de aprendizaje eligió.

4.7. Hiperheurísticas.

En la Fig. 6 muestra la metodología de solución general, en la cual se recibe una solución de un estudio de viabilidad, pero no se sabe si es factible, para comprobar la factibilidad se resuelven la ecuación *(7) y (8)* usando el método del gradiente hidráulico, el cual al tener un flujo permanente se garantiza que se cumplan las ecuaciones de conservación de masa en cada uno de los nodos de la red y la de energía en cada uno de los circuitos de ésta, esto implica que se puede usar para cualquier fluido Newtoniano.

Caso contrario entra al seleccionador de hiperheurísticas (*Fig. 11*), el cual permite elegir de un conjunto de heurísticas de bajo nivel, donde la vecindad va a estar conformada con la modificación de las siguientes variables diámetro de tuberías, bombas y válvulas, donde el objetivo es minimizar el número de cambios.

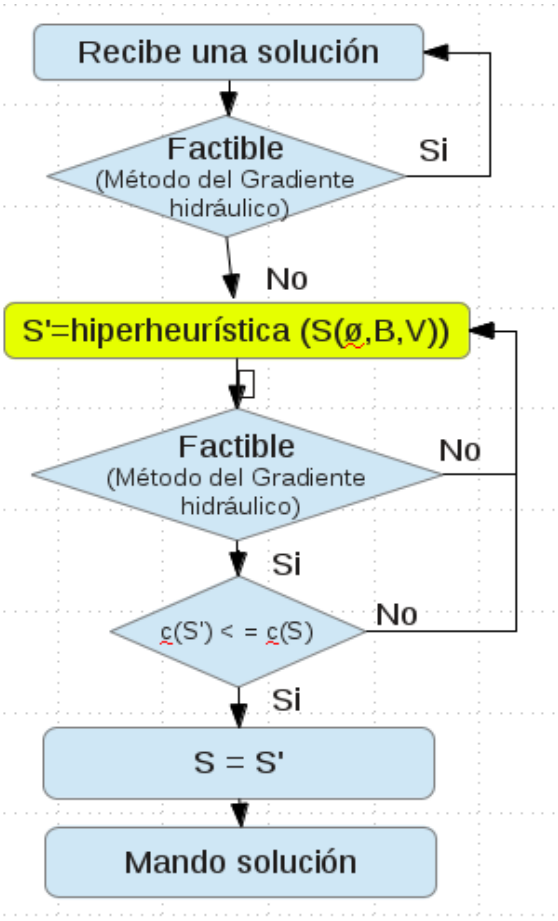


Fig. 11 Diagrama de flujo de la metodología de algoritmo para la validación de una solución propuesta

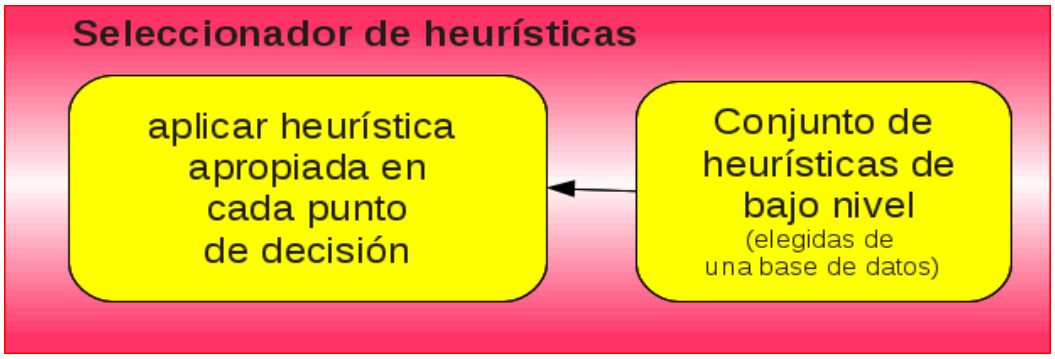


Fig. 12 Seleccionador de heurísticas.

Las heurísticas como son de bajo nivel permiten ya sea la explotación, que permite indagar en las soluciones vecinas, o la exploración que se lleva a cabo cuando una heurística analiza diversos puntos en el espacio de soluciones distanciados unos de otros; la heurística trabaja con base en la experiencia y permite encontrar soluciones cercanas al óptimo. Para evitar caer en óptimos locales y experimentar una convergencia rápida en un procedimiento heurístico se tiene que implementar procedimientos que permitan aceptación o rechazo de soluciones malas (*Fig. 10*), cabe mencionar que cada vez que se genera una solución se tiene que evaluar si es factible o no, esta evaluación de factibilidad se realiza al resolver la matriz resultante de las ecuaciones *(7) y (8)*, aplicando el método del gradiente hidráulico ver *Fig. 11*. La sección de aprendizaje es donde la heurística es capaz de identificar que heurística aplicar en cada punto de decisión.

Cuando una solución es factible se regresa para su viabilidad, de lo contrario entra al seleccionador de hiperheurísticas, el cual permite elegir de un conjunto de heurísticas de bajo nivel, donde la vecindad va a estar conformada con la modificación de las siguientes variables diámetro de tuberías, bombas y válvulas, donde el objetivo es minimizar el número de cambios.

Capítulo 5 Resultados

5.1. Introducción.

En este capítulo se presentan los resultados obtenidos de la hiperheurística aplicada a las instancias real y teórica en forma secuencial. 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 Cuexcomate. Se presenta el análisis estadístico para el algoritmo en forma secuencial.

5.2. Descripción del equipo utilizado.

Para las instancias de prueba teórica y real se utilizó la infraestructura del clúster Cuexcomate ubicado en la Universidad Autónoma del Estado de Morelos (UAEM), campus Chamilpa, Cuernavaca, Morelos. Las características del clúster Cuexcomate se presentan en la *Tabla 2*.

Tabla 2 Configuración del cluster Cuexcomate.

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	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 modulos RAM de 4GB 1333MHZ DDR3. Total de 24 GB RAM.

Total 12 TB HD	• 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 modulos 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 modulos 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

5.3. Descripción de la redes de distribución de fluidos.

Para las pruebas se trabajaron con dos instancias de prueba teórica e instancias de prueba real las cuales constan de 1000 nodos y 666 nodos respectivamente.

- **Instancia de prueba teórica.**

La primera que es la instancia de prueba teórica, se creó al conectar dos puntos arbitrarios. Las tuberías están conectadas por nodos, que hacen

un total de 1000, un embalse que representa el pozo de extracción o las baterías de recolección según sea el caso y 1000 tuberías, en la *Fig. 13* se muestra gráficamente la trayectoria.

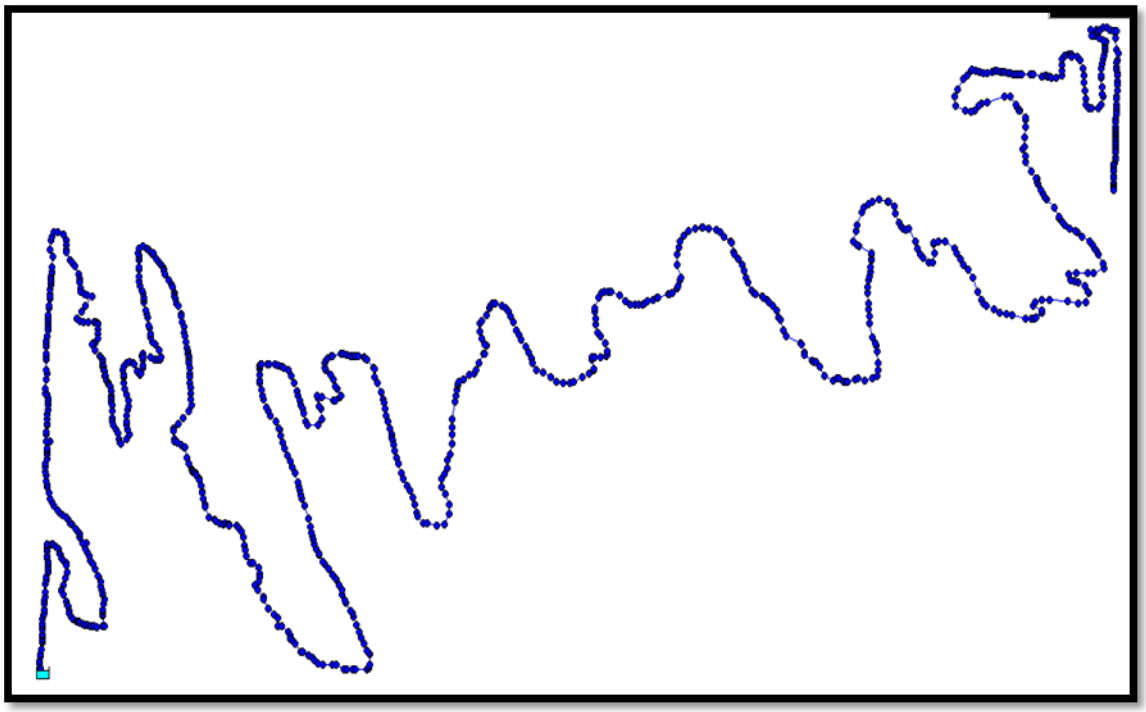


Fig. 13 Gráfica de la instancia de prueba teórica

Los datos de la red se muestran en la *Tabla 3*, donde se ve que la trayectoria es de más de 17 *km*, de las tuberías que enlazan los nodos la más pequeña es de 0.20 *m* y la más grade 117.1 *m*, todas y cada una de las coordenadas fueron tomadas de forma arbitraria solo con el objetivo de que se siguiera un trayectoria para unir dos puntos unidos por nodos los cuales deberían de ser 1001, ya que el primero siempre se toma como embalse y no se cuenta como nodo de conexión.

En cuanto a las alturas se tomó dentro de un rango de 0 a 200 m, esto debido a las condiciones de relieve del estado de Veracruz, México, tiene esas características. La elevación más pronunciada de la costa de Veracruz es de 200 m y el nivel del mar se toma como 0 m *[mexico.pueblosamerica, 2019]*, ver *Tabla 3*.

Cada punto representa un nodo que es la unión de dos tuberías y cada nodo no tiene demanda es decir que no habrá fluido que se extraiga de ahí, este nodo se representa con las tres coordenadas latitud, longitud y altitud.

Tabla 3 datos de la instancia de prueba teórica

Distancia total **(*m*)**	**Tubería más grande** **(*m*)**	**Tubería más pequeña** **(*m*)**
17 623.48	117.01	0.02

- **Instancia de prueba real.**

La segunda es la instancia de prueba real, esta instancia fue tomada de un trabajo previo la cual se evalúa la viabilidad, en este trabajo se indica que antes de aprobar la construcción de la nueva tubería se deben calcular las necesidades de recursos tanto humanos como materiales, así como la disponibilidad de financiación [Cabrera *et al.* 2009].

En esta tesis se evalúa también la factibilidad, es decir, se evalúan las ecuaciones gobernantes y se determinan si se cumplen las restricciones de caudal y presiones en cada tubería y nodo.

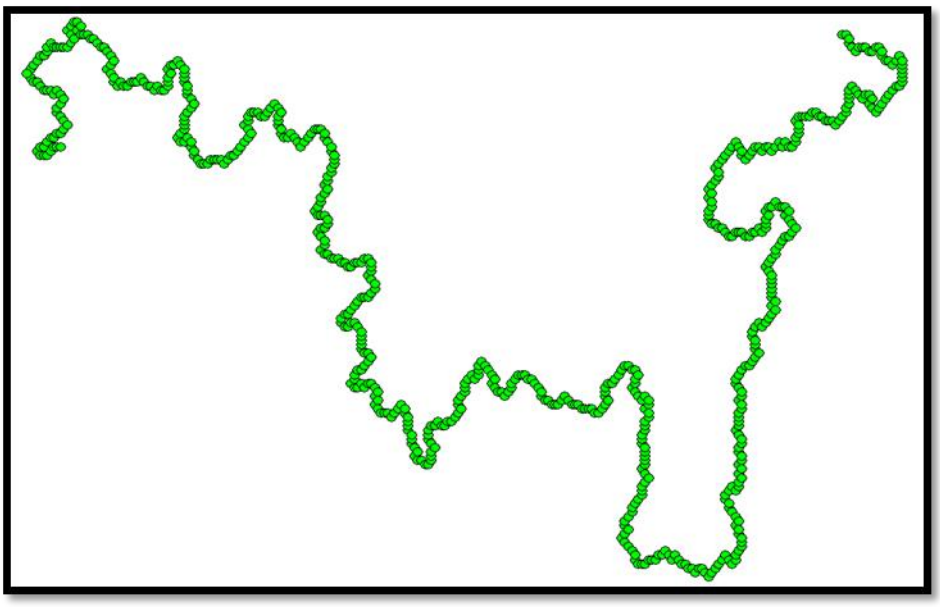


Fig. 14 Trayectoria de la instancia de prueba real.

Esta trayectoria une dos puntos ubicados en Veracruz, México, los cuales son: el poblado de Santa Rita y la colonia de Paya Linda del Puerto de Veracruz. La localidad de Santa Rita es el inicio de trayectoria y cuenta con 1056 habitantes (mexico.pueblosamerica, 2019) y está ubicada a 40 *metros de altitud*.

La colonia de Playa Linda del Puerto de Veracruz y el poblado de Santa Rita tienen coordenadas 19° 9’ 53.942” *N*, 96° 15’ 50.398” *O* y 19° 12’ 28.199” *N*, 96° 10’ 23.521” *O*, respectivamente, La *Fig. 15* y *Fig. 16* muestran las coordenadas en la página coordenadas-gps el cual sirve para ubicar cualquier lugar en la tierra (google, 2019).

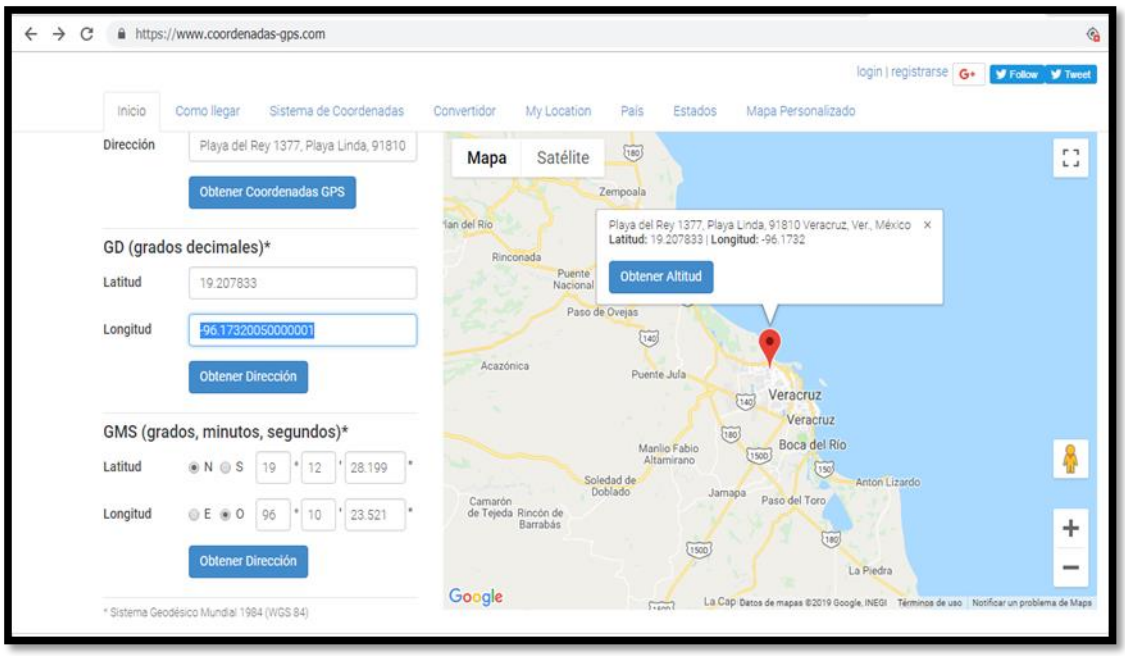


Fig. 15 inicio de la instancia.

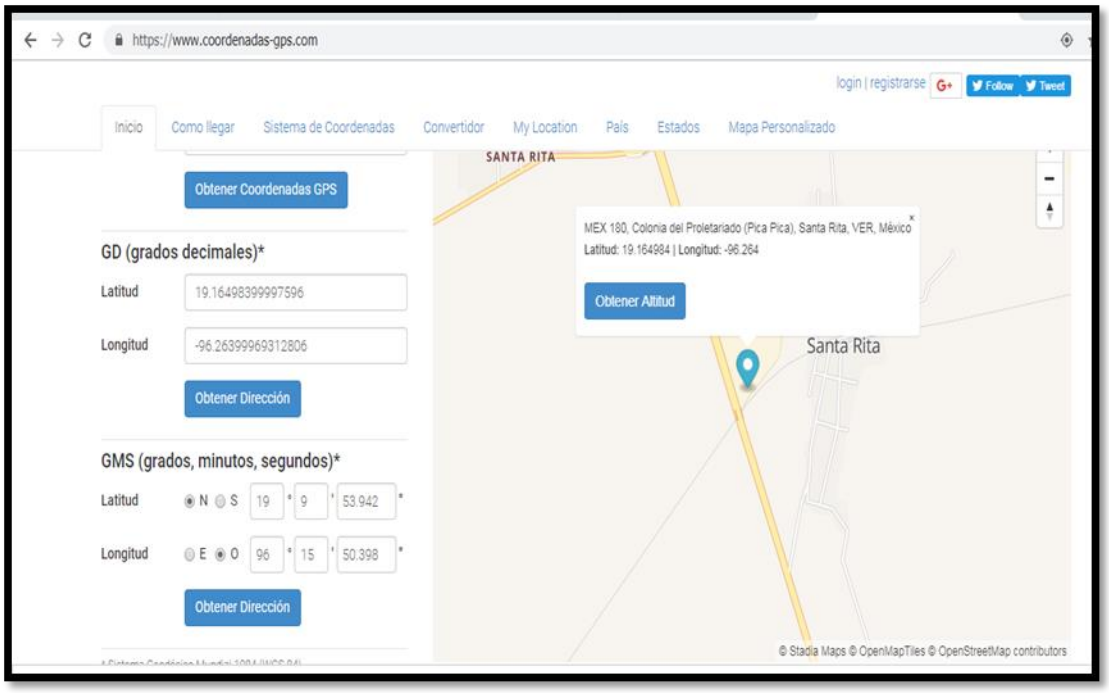


Fig. 16 Fin de la instancia.

En la *Fig. 17* y *Fig. 19* se muestran los dos puntos de inicio *a)* y final *b)* así como la trayectoria que parte desde playa linda y llega al poblado de Santa Rita. En la *Fig. 18* se observa el mapa del relieve así como los poblados de la zona, y en la *Fig. 19* se puede observar cómo la trayectoria evita los mantos acuíferos, poblados, carreteras federales, pantanos, entre otras.

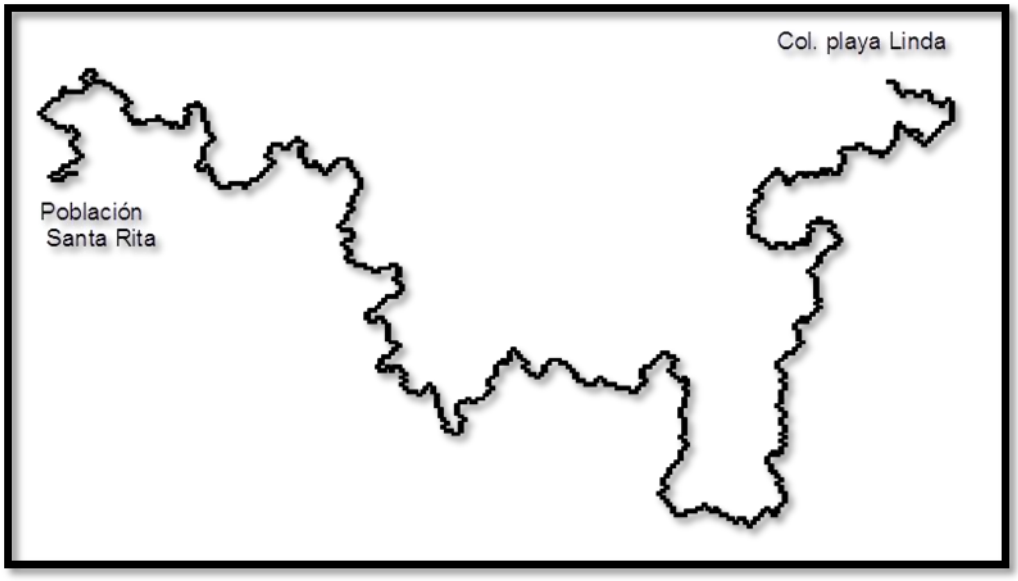


Fig. 17 Trayectoria que uno dos poblaciones.

El mapa de la zona se puede ver en la siguiente figura donde se observa dos marcas las cuales apuntan los lugares descritos anteriormente, La marca en el poblado se ve marcado en otro espacio esto es porque al ubicar el lugar manda al centro del poblado sin embargo la trayectoria llega hasta un costado de este poblado, como lo indica la *Fig. 19*.

Fig. 18 Ubicación de los dos lugares de la trayectoria.

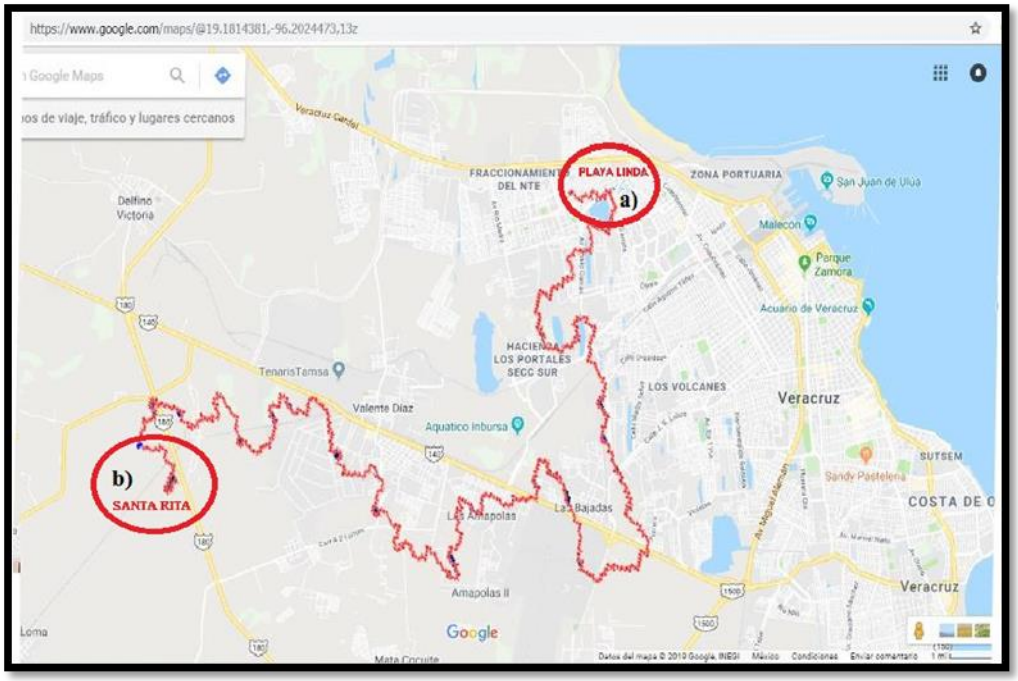


Fig. 19 Ubicación de la trayectoria.

Los datos de la instancia de prueba real se muestran en la *Tabla 4*, la cual indica que la tubería más grande es de 4 801.1 *m* y la más pequeña es de 50 *m*, a comparación de la tubería más grande de la instancia de prueba teórica que es de 117.1 *m*, y la más pequeña 0.20 *m,* y esto se ve reflejado en el tamaño de las trayectorias, mientras que en la distancia total de una va de 317 848.64 *m* a 17 623.48 *m* de la otra, ver *Tabla 4*.

Sin embargo en cuanto a las alturas la instancia de prueba real tiene una altitud máxima es de 117 *m*, en este punto la instancia de prueba teórica es mayor con 200 *m*.

Tabla 4 Datos de la instancia.

Distancia total (*m*)	**Tubería más grande (*m*)**	**Tubería más pequeña (*m*)**
317 848,64	4801.1	50

Los requerimientos de la red de distribución se muestran en la *Tabla 5* y *Tabla 6*, los cuales fueron tomados de las condiciones de operación de la industria petrolera de México. Esta instancia fue tomada de los dos puntos antes mencionados, sin embargo se pueden cambiar e inclusive puede ser el inicio tanto el pozo de salida, baterías de recolección o la estación de recolección, para este caso en particular el inicio es el patín de recolección, el cual debe tener una presión de salida de 703 *mca*. Estos cambios dependerán de los requerimientos es decir que la presente metodología puede servir para otras instancias.

Tabla 5 Valores de las presiones en cada uno de los puntos de inyección de fluido.

Lugar	Presión (*Kg/cm2*)	Presión (*mca*)
Valor de presión a la salida del pozo:	17.570	175.7
Valor de presión a la salida del patín de recolección:	36.200	362.0
Valor de presión a la salida de la estación de recolección:	70.300	703.0
Presión otorgada al colocar una bomba:	50	500

Tabla 6 presión de las diferentes tuberías.

Diámetro de la tubería	Presión (*Kg/cm2*)	Presión (*mca*)
Menor a 6”	con un valor límite inferior a 21	210
Superior a 4”	con un valor límite inferior a 5	50
6” (se utiliza esta tubería para cualquier presión superior a 21 kg/cm^2 (210 *mca*))	21.0 - 40	210-400

Para esta instancia en particular el valor de la presión a la salida del pozo, es a 175.70 *mca* como se puede mostrar en la *Fig. 20*. Se simula con un embalse a una cota de 175,70 *m* de altura con respecto al nodo inicial, sin importar el tamaño de la tubería y la altura que este tenga, es decir la presión del nodo inicial es la presión del nodo origen más su altura a la que este se encuentre (*Fig. 20*). Con esto se puede deducir que al subir el embalse hasta la cantidad en *mca* es la presión de salida del pozo inyectado al sistema, que en nuestro caso sería la presión de salida del patín de recolección, este movimiento se hace para darle las condiciones iniciales correctas al sistema.

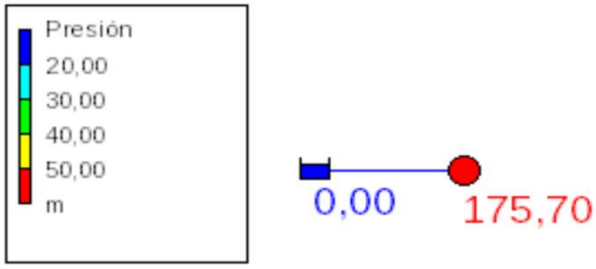


Fig. 20 Representación gráfica de la presión de entrada del sistema.

Para el cálculo del combustible suministrado en una estación que en este caso va a ser el último nodo demanda, se sabe que es de 50 L por minuto a lo que equivale a 0,8333 L por segundo por 8 bombas 6,6664 L/s, entonces en el nodo final se tiene una demanda de 6,6664 L/s (PEMEX, 2006).

5.4. Diámetros Comerciales.

Los diámetros comerciales usados en este trabajo se tomaron de (Tubacero, 2015) y (Cotosa, 2015) Ver Apéndice Apara el caso de las bombas y válvulas. A continuación en la *Tabla 7* y en la *Tabla 8* se muestran los valores de los diámetros y potencias de bombas utilizados en este trabajo, respectivamente.

Tabla 7 Diámetros comerciales

No	Diámetro (*mm*)	No	Diámetro (*mm*)
1	25.4	**21**	279.4
2	38.1	**22**	292.1
3	50.8	**23**	304.8
4	63.5	**24**	317.5
5	76.2	**25**	330.2

6	88.9	**26**	342.9
7	101.6	**27**	355.6
8	114.3	**28**	368.3
9	127	**29**	381
10	139.7	**30**	393.7
11	152.4	**31**	406.4
12	165.1	**32**	419.1
13	177.8	**33**	431.8
14	190.5	**34**	444.5
15	203.2	**35**	457.2
16	215.9	**36**	469.9
17	228.6	**37**	482.6
18	241.3	**38**	495.3
19	254	**39**	508
20	266.7		

Tabla 8 Potencias nominales

No	**Potencia (*KW*)**
1	0.093
2	0.186
3	0.249
4	0.559
5	0.746
6	1.119
7	1.491
8	1.864
9	2.237
10	2.983
11	4.101
12	5.593
13	7.457
14	9.321
15	11.186
16	12.677
17	14.914

5.5. Factibilidad.

Las presiones máximas que debe soportar la tubería es de 17.2 kPa, lo que equivale a 1,754 mca y las presiones mínimas son de 0.3 Pa que equivale a 30 mca *(PEMEX, 2006)*. La *Fig. 21* muestra la presión en cada uno de los nodos y se ve que está dentro de estos valores establecidos, la línea marcada en rojo inferior indica 30 y la superior 1600 mca , cabe recalcar que la presión máxima permitida para el diseño es de 1750 mca, la cual está por debajo de la obtenida, donde el eje X representa el número de nodo y el eje Y la presión del nodo.

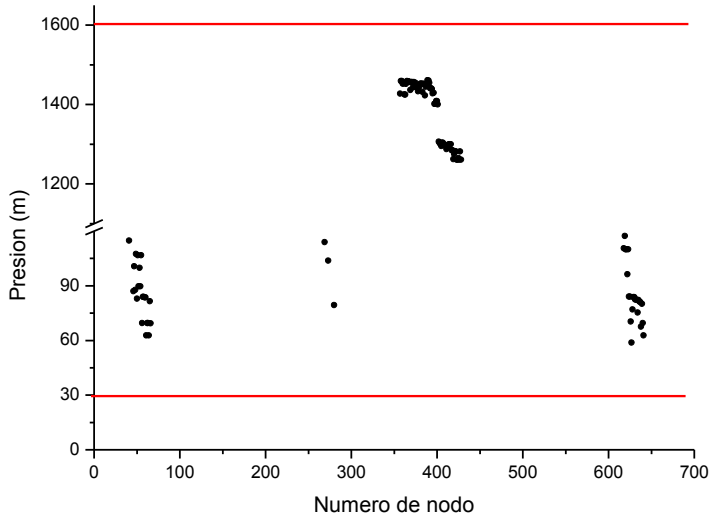


Fig. 21 Factibilidad para 666 nodos.

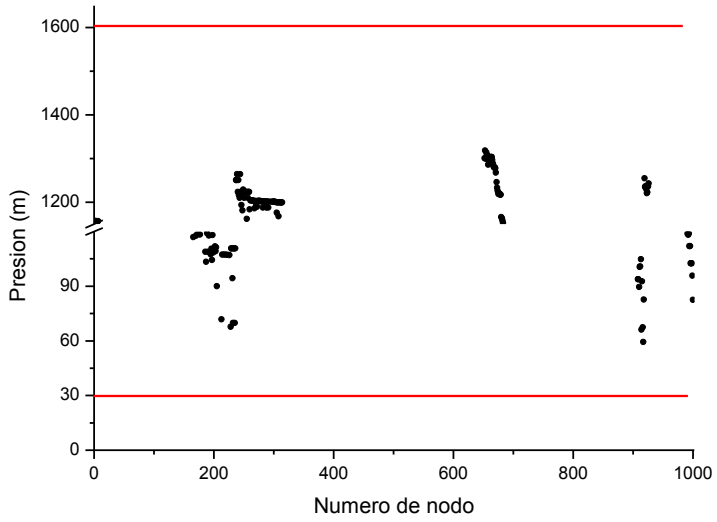


Fig. 22 Factibilidad para 1000 nodos

Fig. 22 y *Fig. 22* se muestra la factibilidad para 666 y 1000 nodos respectivamente de la solución inicial pero no está optimizada, es decir no ha entrado a la hiperheurística para su optimización, una vez que entra la función de aprendizaje se empiezan a aplicar las búsquedas locales, iteradas y búsqueda tabú, y como resultado la *Tabla 9* muestra las estructuras de vecindad que se terminaron usando.

Es decir de 81 heurísticas (81 estructuras de vecindad), se redujo a 51 (17 heurísticas por 3 cambios) debido a que cuando entra la heurística de bajo nivel si no genera soluciones factibles o no mejora durante varias ejecuciones, esta se elimina y finalmente la función de aprendizaje se queda con las mejores heurísticas (*Tabla 9*). Todas las búsquedas donde la estructura de vecindad involucraba válvulas fueron eliminadas, ya que las presiones son altas no necesitan ser consideradas, sin embargo la hiperheurística es capaz de ponerla si en algún momento las condiciones cambiaran es decir que las presiones a las cuales se sometieran en los tubos fueran menores o que el relieve fuera más pronunciado.

Tabla 9 Estructuras de vecindad finales

Número	Intersección	Componente
1	Reducir	Diámetro
2	Cambiar-aleatorio	Diámetro
3	Cambiar	Bomba
4	Quitar	Bomba
5	Quitar-Poner	Bomba
6	Quitar-Cambiar	Bomba
7	Poner-Cambiar	Bomba
8	Poner-Cambiar-Quitar	Bomba
9	Quitar-Reducir	Bomba-Diámetro
10	Quitar-Aleatorio	Bomba-Diámetro
11	Quitar-Aumentar	Bomba-Diámetro
12	Poner-Reducir	Bomba-Diámetro
13	Poner-Aleatorio	Bomba-Diámetro
14	Cambiar-Aumentar	Bomba-Diámetro
15	Cambiar-reducir	Bomba-Diámetro
16	Cambiar-Aleatorio	Bomba-Diámetro
17	Poner-Cambiar-Quitar-Aumentar-Reducir	Bomba-Diámetro

También se nota que el aumento del diámetro se excluyó, pero no el cambio de diámetro aleatorio, es decir que puede ser que aumente o disminuya el diámetro durante la ejecución de la búsqueda local, esto se debe a que al aumentar el diámetro la presión disminuye entonces se evita poner alguna válvula y viceversa si al reducir el diámetro disminuye el costo, pero la presión aumenta así que se tienen que poner válvula y esto impacta el costo. Así que al combinar el aumento y disminución de los diámetros de las tuberías y poner o no válvulas, si mejora el costo de la configuración.

5.6. Análisis de sensibilidad de los parámetros de la instancia de prueba teórica.

Es necesario el análisis de sensibilidad para tener el control de los parámetros importantes de nuestro algoritmo ya que 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 éste tenga el mejor funcionamiento posible en cuanto a la eficiencia y eficacia.

La forma en que se determinaron los términos a sensibilizar fue realizando un análisis tanto del problema como del algoritmo, a manera de identificar los parámetros críticos que influyeron de cierta manera en la calidad de las soluciones.

De acuerdo a la bibliografía la hiperheurística debido a la función de aprendizaje los parámetros de sensibilidad se reducen [Edmund Burke *et al.,* 2003], en este caso los parámetros que se sintonizaron fueron los siguientes:

- Tiempo de ejecución de las búsquedas locales.
- Tiempo total de la hiperheurística.

Estos parámetros fueron seleccionados debido a que de acuerdo a la literatura son los parámetros críticos del algoritmo y todo el procedimiento se desarrolló en forma secuencial.

Se realizaron 30 ejecuciones en cada prueba y se dejó fija la búsqueda local clásica con una estructura de vecindad. Se realizaron cada una de las estructuras de vecindad mostradas en la *Tabla 9*.

Tabla 10 Tiempo de iteraciones de la estructura de vecindad.

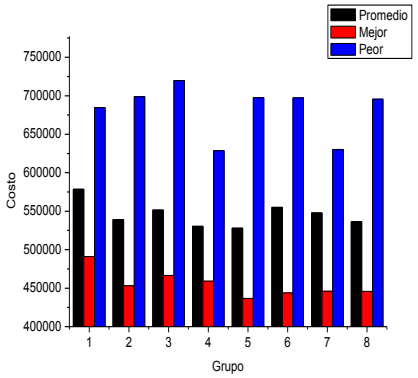
Grupo	No de iteración	Promedio t(*min*)	Mejor t(*min*)	Peor t(*min*)
1	10	1.56	0.08	21.45
2	500	4.16	0.32	48.47
3	100	1.15	0.07	11.55
4	1000	5.67	0.97	5.27
5	1500	6.58	2.13	73.12
6	2000	17.62	2.48	8.417
7	5000	53.93	5.43	44.1
8	10000	93.39	13.15	97.92

Tabla 11 Costo de las iteraciones de la estructura de vecindad.

Grupo	No de iteración	Costo promedio	Mejor Costo	Peor Costo
1	10	578445.36	491035.38	684719.06
2	500	538971.89	453239.84	698707.25
3	100	551382.49	466375.06	719702.44
4	1000	530393.51	459118.59	628564.50
5	1500	527862.81	436575.31	697514.94
6	2000	554979.66	443852.25	697358.50
7	5000	547871.41	445918.22	630067.81
8	10000	536479.49	445784.78	695640.63

El resultado mostró ser el peor caso la estructura de vecindad: poner, cambiar, quitar bombas y aumentar, reducir diámetros, ya que ésta requería más tiempo de cómputo para su convergencia. En la *Tabla 11* se muestran los resultados de la sintonización del parámetro del tiempo de ejecución de la búsqueda local con la estructura de vecindad antes descrita, donde se puede apreciar que se agruparon los datos, cada grupo se ejecutó 30 veces con un número de iteraciones de 10, 100, 500, 1000, 1500, 2000, 5000 y 10 000, donde se encontró que entre más grande es el número de iteraciones mayor es el tiempo de ejecución (ver *Tabla 10*). Sin embargo, el número de iteraciones es pequeño en comparación al número de movimientos posibles que la estructura de vecindad puede hacer para acotar el problema debido al tiempo que este podría tardar en su ejecución. Se tomaron 5 minutos para su

ejecución esto es porque en el grupo 4 de 5.67 *min* subió a 6.58 *min* para el tiempo promedio, pero en cuanto al peor caso se observa que incrementó de 5.27 a 73.12 *min, las Fig. 23* y *Fig. 23* muestran las gráficas del comportamiento de los grupos contra costo donde se ve que no disminuye significativamente el costo conforme el tiempo crece.



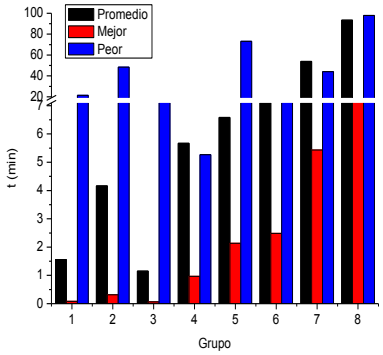


Fig. 23 Costo Vs iteraciones

Fig. 24 Tiempo Vs Iteraciones

Entonces finalmente se ajusta el tiempo del algoritmo a 40 *h* de ejecución y 5 *min* cada búsqueda local, tabú o iterada y cada hora se aplica cruzamiento. Para la instancia de prueba real de Veracruz se ajusta a 10 *h* y de igual forma que la anterior 5 *min* cada búsqueda local, tabú o iterada y cada hora se aplica cruzamiento.

5.7. Convergencia del algoritmo.

Para la sintonización de los parámetros se ejecutó 30 veces el algoritmo, se fijaron la búsqueda local clásica, la estructura de vecindad donde se pone, cambia, quita bombas y aumenta, reduce diámetros, se eligió esta estructura de vecindad porque es la que más permutaciones tiene de

todas las estructuras de vecindad, por lo tanto es la que más tiempo de ejecución necesita.

Una vez que se sintonizaron los parámetros en un tiempo de 40 5 *h* (2 400 *min*), se observó la convergencia del algoritmo como se muestra en la *Fig. 25* y *Fig. 26*. En la *Fig. 25* primera se observa que después de 2 336 *min*, se comporta de forma asintótica en el peor caso, a diferencia del mejor caso que a 179 *min* y el mejor caso a 1 137 *min* el comportamiento ya es asintótico. Con esto se determinó que a mayor tiempo de ejecución el costo de la solución seguirá bajando sin embargo el costo a pagar es el tiempo de ejecución, por lo tanto las pruebas se acortaran a 10 *hrs* (600 *min*). Como conclusión el algoritmo necesita más tiempo de ejecución o paralelizar para mejor la calidad de la solución pero ya no entra en el objetivo de este trabajo.

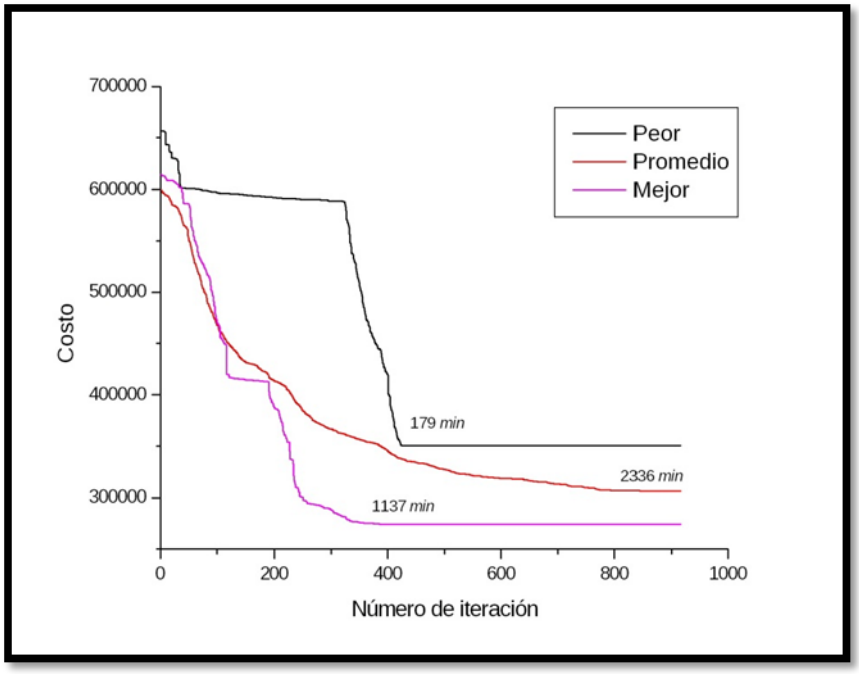


Fig. 25 Convergencia para 1000 nodos

En el caso de la instancia de prueba real de Veracruz, se ejecutaron 30 pruebas en un tiempo de 10 *h* (600 *min*) cada prueba y el resultado se ve

en la *Fig. 26*. En el peor caso la convergencia llegó a 299 *min* y lo mejor es que el promedio de las 30 ejecuciones llegó a 358.8 *min*, y a partir de ahí se ve que no cambia el valor caso contrario a la instancia de prueba teórica, esto se debe a que el número de nodos es menor y eso reduce el tiempo de cómputo ya que por cada nodo se tiene que resolver la matriz de ecuaciones gobernantes del sistema que hace el paquete comercial EPANET, pero de igual forma por cada ejecución se tiene que evaluar la factibilidad. Y en cada movimiento posiblemente no se generen factibles, lo que hará será descartar esa matriz y encontrar otra.

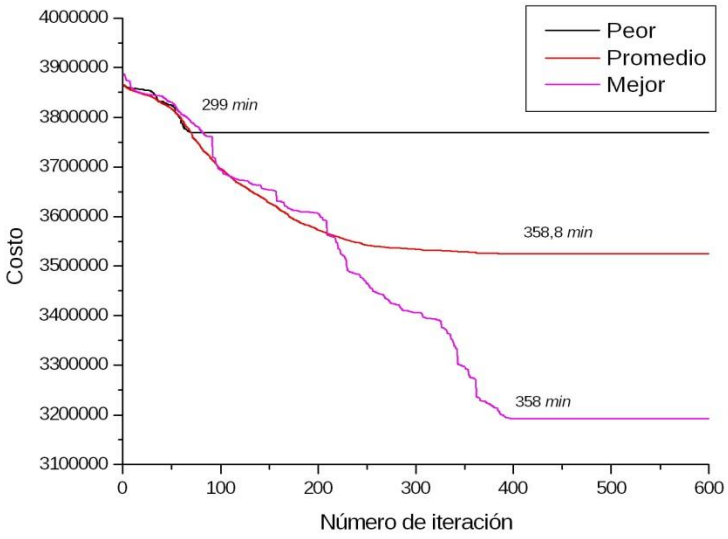


Fig. 26 Convergencia de 666 nodos

5.8. Análisis estadístico de la instancia de prueba teórica.

La *Fig. 27* muestra los resultados de una instancia de 1000 nodos en la cual, se activaron 15 bombas, sin embargo a simple vista no se logra

apreciar en que parte se encuentran ubicadas y eso es debido a que como es una instancia grande las bombas se montan así en la *Fig. 28* se ve en la parte inferior izquierda un acercamiento en el cual muestra que son dos bombas las que se encuentran cerca del embalse, esto es debido que al inicio necesita presión para poder llevar el fluido por las tuberías con la presión deseada. Como esta red no tiene presión inicial se necesita algún medio mecánico para poder elevar la presión. Lo que se observó en los resultados es que en ocasiones resultaba más barato colocar dos bombas de menor caballaje que una que es de costo más elevado.

En cuanto a la *Fig. 28* el acercamiento indica que están dos bombas conectadas en serie pero no están unidas por un nodo esto es debido a las restricciones de propio software el cual tiene como restricción que no se pueden conectar dos bombas al mismo nodo es decir el paralelo, así que las ubica en serie y ahí si dos bombas compartirían un nodo donde uno sería el nodo final y para la otra bomba sería el nodo inicial.

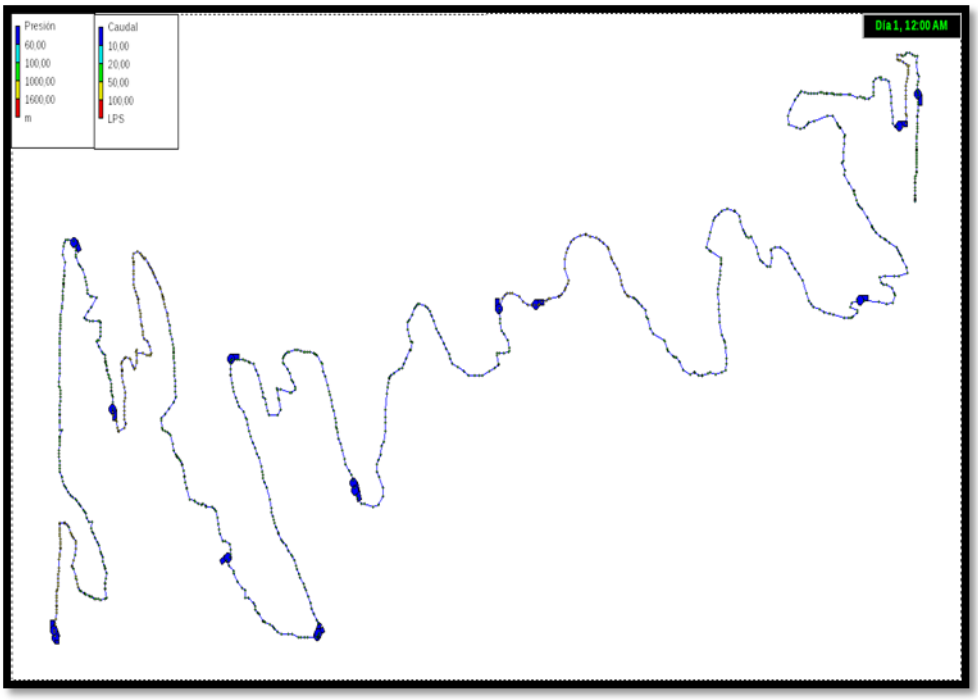


Fig. 27 Red optimizada.

Se observa que la trayectoria de la red no tiene una forma lineal porque la instancia de prueba teórica está evitando los poblados, carreteras, mantos acuíferos, y otros. Pero a pesar de esto la hiperheurística es capaz de encontrar una trayectoria factible y optimizarla.

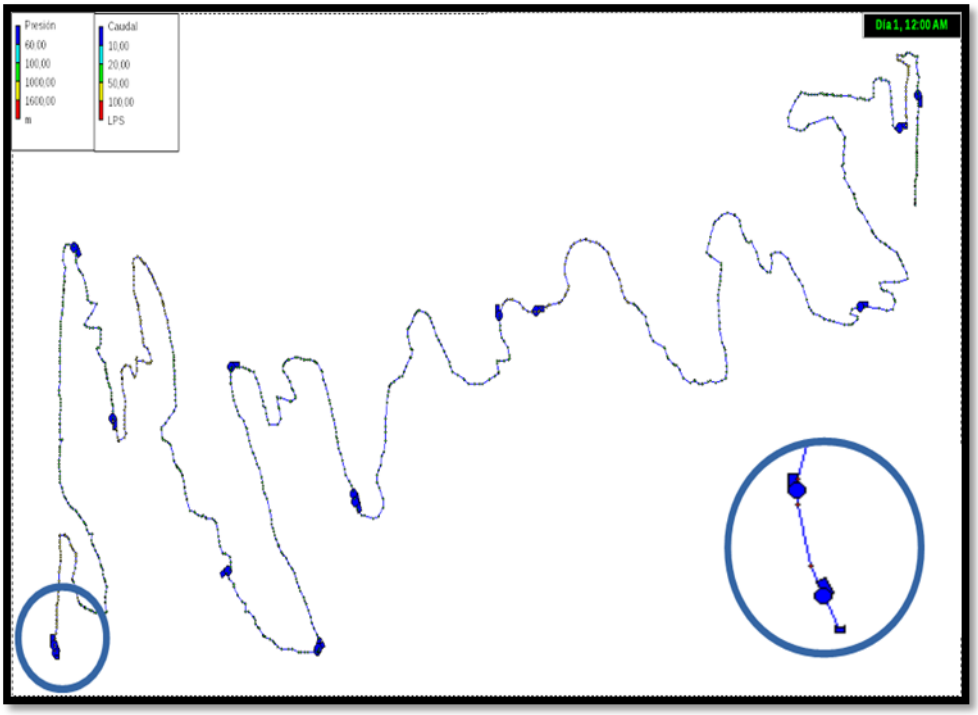


Fig. 28 Bombas de la red.

La solución optimizada tiene que ser factible y para demostrarlo se puede ver en la *Fig. 30* y *Fig. 30* que todos los nodos están dentro de los rangos de 30-1750 *mca* si hubiera algún nodo marcado en color rojo, estaría fuera de rango. En caso de la gráfica se hizo un corte en el eje "*y",* el cual permite observar los extremos, todo los que no se muestra debido al corte se encuentra en los márgenes, en los extremos se alcanza a notar que la presión no alcanza ni los 1500 *mca*, y en el límite inferior no llega ni a los 40 *mca*, es decir que la heurística está entregando soluciones factibles y

optimizadas. La *Fig. 30* muestra las presiones de las dos instancias donde se muestra que ninguna de las dos excede las presiones.

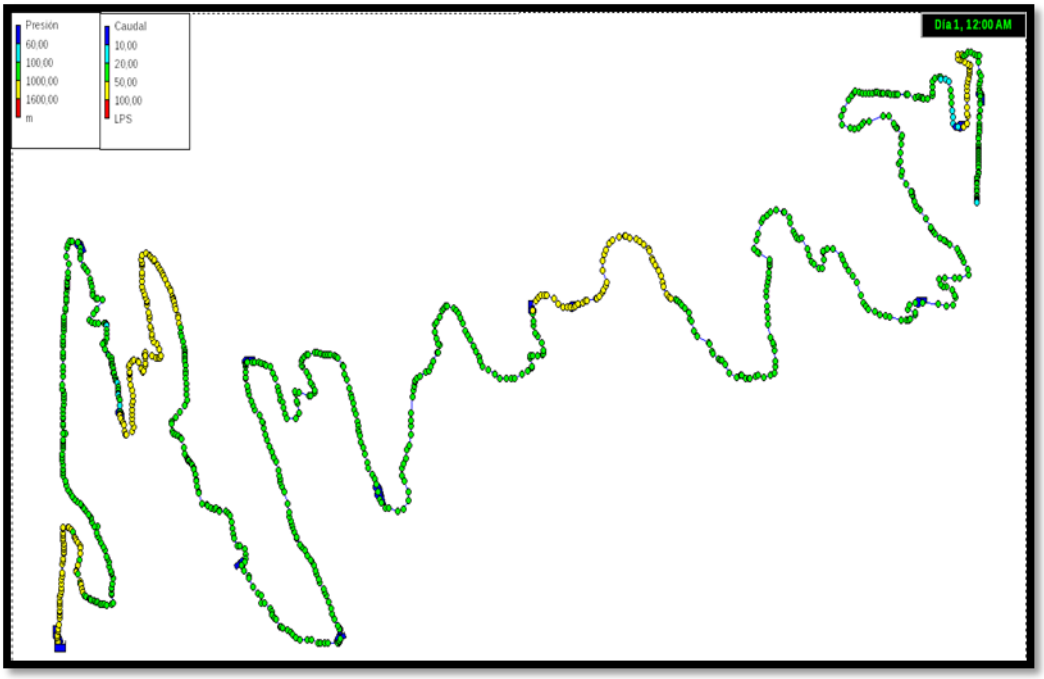


Fig. 29 Presión en los nodos

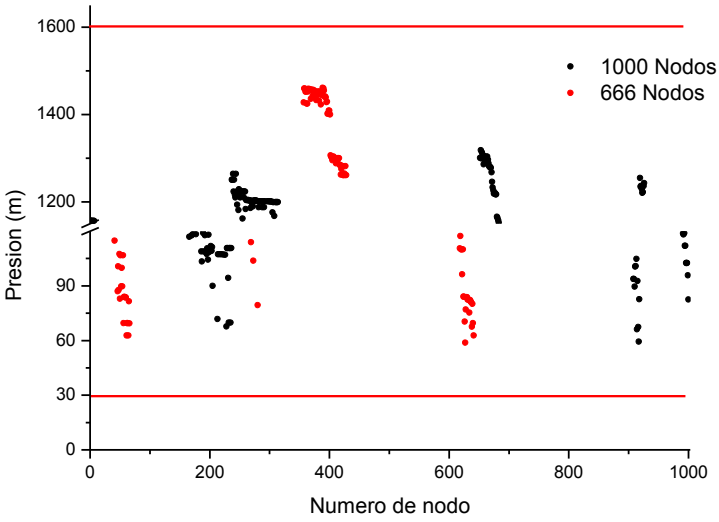


Fig. 30 Presión en los nodos.

Al finalizar las 30 ejecuciones se encontró lo siguiente: el mejor costo fue de $274 197.6, se agregaron 84 bombas en puntos estratégicos para hacer factible la solución inicial y al optimizarla terminó con 15 bombas, en cuanto a los diámetros el menor fue de 25.4 mm y el mayor de 508 mm y al calcular el promedio de los diámetros marcó: 254.4 mm con una desviación estándar: 145.7 esto indica que el sesgo de los números estuvo cargado hacia los diámetros de menor tamaño, esto es un buen indicio que la heurística estuvo jugando con los valores y si optimizó de una forma correcta, ya que a menor diámetro menor costo. Considerando que la longitud total de trayectoria es 17 623.5 m es decir casi de 18 km, esto es una distancia considerable, la hiperheurística encontró un buen camino para reducir los costos teniendo en contra lo grande de la trayectoria.

Cabe recalcar que la altura a la que están las tuberías es entre 0-200m, para los 666 nodos. En la *Fig. 31*. Se muestra el histograma de frecuencias donde se ve el comportamiento de la hiperheurística, la frecuencia de los datos es mayor para $300 000, o sea que de las 30 ejecuciones 9 tuvieron un costo de entre $300 000, pero solo en una ocasión fue menor a $280 000, es decir que una ejecución logró bajar el costo por debajo de ese costo. Sin embargo el sesgo se encuentra cargado a la izquierda lo cual indica que encontró costos más bajos que altos. Esto es un buen índice de que hiperheurística está trabajando.

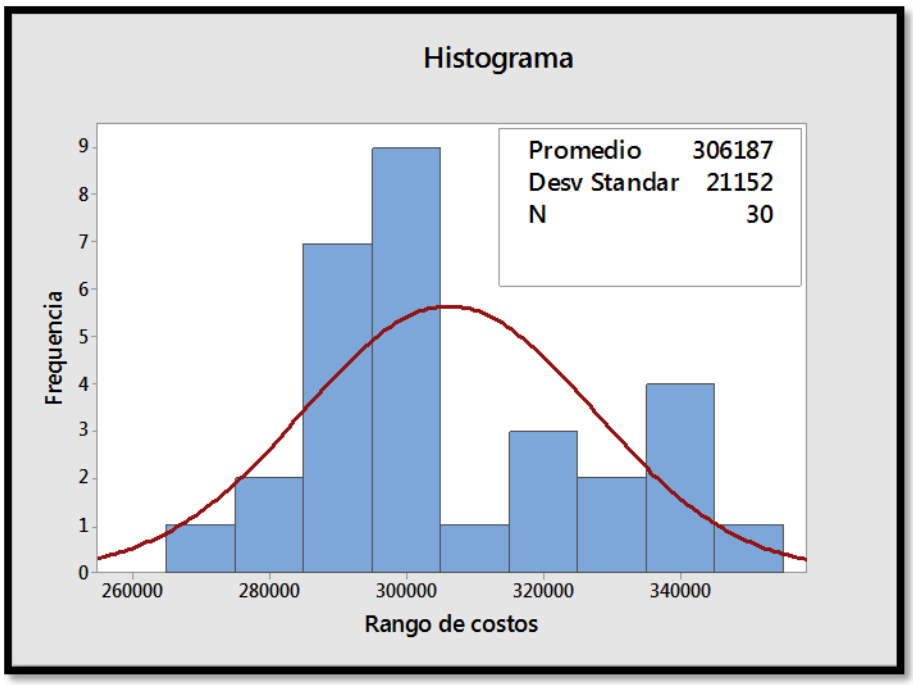


Fig. 31 Histograma de frecuencias.

En la *Tabla 12* se presenta la mejor, la peor, el promedio de las soluciones 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 grande pero si se compara con respecto a los costos es pequeña, por lo tanto la media es un valor confiable de la solución que el algoritmo produce.

Tabla 12 Resultados de los costos de la hiperheurística

Medidas de tendencia central y dispersión	Costo ($)
No pruebas	30
Promedio	352 495.8
Desviación	142 431
Estándar	
Mejor	274 197.59
Mediana	298 930.58
Peor	350 160.25

5.9. Análisis estadísticos de la instancia de prueba real.

La trayectoria mostrada en la *Fig. 32* une dos puntos el poblado de Santa Rita y la Colonia Playa Linda del estado de Veracruz, como ya se detalló en la sección 5.2, esta red es prueba y al ser optimizada se proponen 27 bombas, a pesar de ser más pequeña que la anterior se necesitaron más bombas esto se debe a la forma que tiene una distribución no uniforme y menos distribuida que la instancia de prueba teórica. Además de que las curvas son más cerradas, esto genera que la presión al terminar una curva sea más pequeña que una curva menos pronunciada.

De igual forma en el círculo marcado en la parte superior izquierda muestra cómo se montan dos bombas pero al contrario de la anterior esta no comparte un nodo sino que se encuentra una bomba en una tubería y después de ella otra tubería y ésta está conectada a un dos el cual contiene otra bomba.

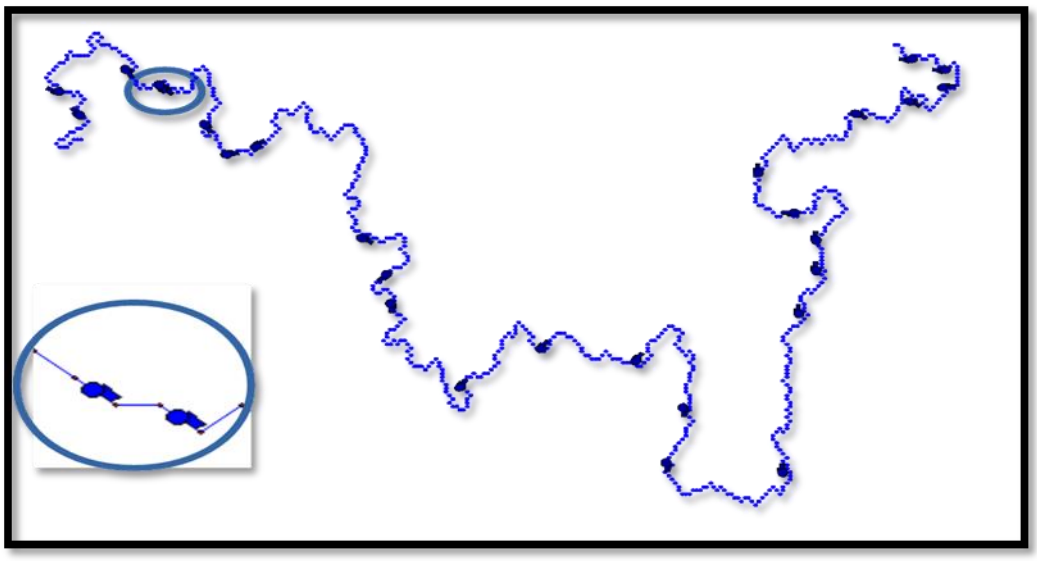


Fig. 32 Bombas de la red

Sin embargo de nada sirve si se tiene una red optimizada si no hay factibilidad en ella es así como en la *Fig. 33* se ven claramente las bombas y los nodos y ninguno de ellos está en color rojo, el cual indicaría que las presiones no están en el rango que se pide, todos los valores se encuentran por debajo de los 1600 *mca* y por arriba de los 30 *mca*.

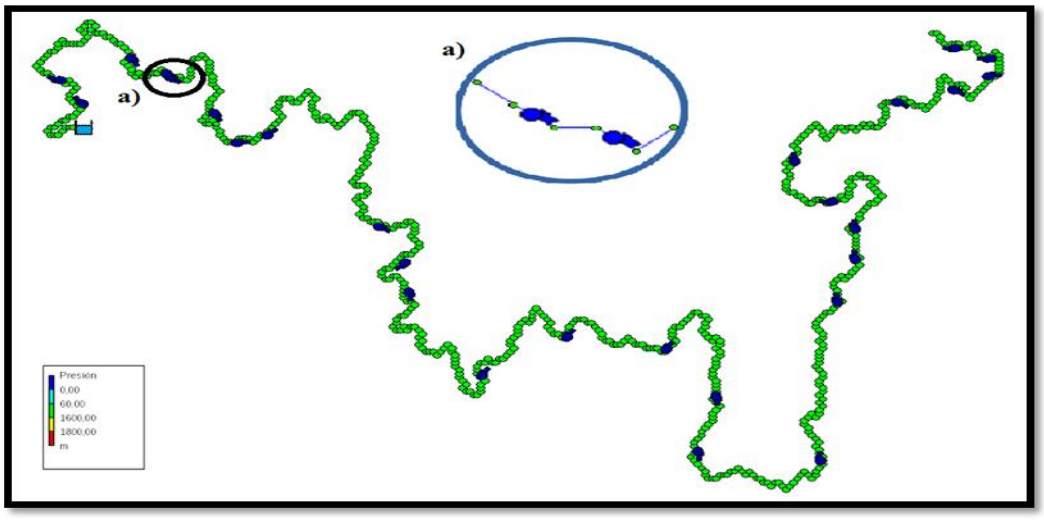


Fig. 33 Presión de los nodos.

La instancia con una longitud total de 317 848.6 m, al finalizar las 30 ejecuciones se encontró que el menor costo fue de: $3 192 687.3, si comparamos con la instancia de prueba teórica veremos que los costos de esta están muy por arriba de hecho hay una diferencia de $2 6 000 000, pero la instancia de prueba es mucho más grande a pesar de tener menos nodos. En total se encontraron 27 bombas, con diámetros de 25.4 mm a 508 mm donde el promedio de ellos fue de 245.2 mm con una desviación estándar de 137.81 menor a la instancia de prueba teórica, es decir su comportamiento del algoritmo fue mejor en éste.

Además que la instancia de prueba teórica las alturas variaron de entre 0 y 200 m en ésta las alturas de los nodos estuvieron entre 0-117 m.

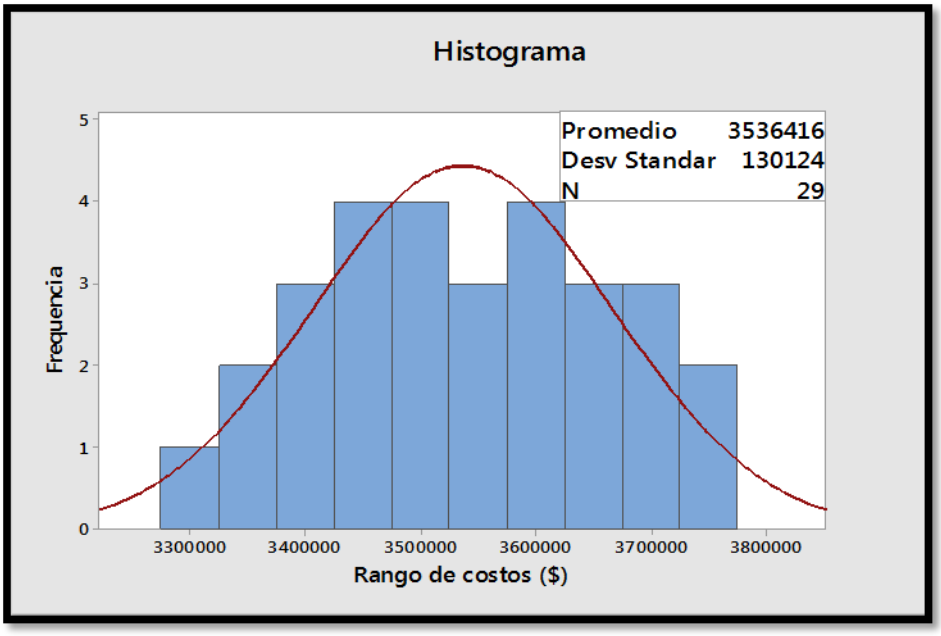


Fig. 34 Diagrama de frecuencias.

En la *Fig. 34* se muestra el comportamiento en el histograma de frecuencias el cual se ve un comportamiento diferente al caso anterior, ya que tuvo mayor presencia entre los rangos de $3 520 00 y $3 680 000 y solo

una vez tuvo el costo por debajo de los $3 200 000. Esto se debe a que la instancia de prueba a pesar de tener menos nodos es mucho más larga y de curvas más pronunciadas, lo que genera menos soluciones factibles así que para hacerla factible se necesitan aditamentos más caros. Pero a pesar de esto la campana de Gauss, se encuentra en el centro lo que indica que los costos estuvieron cargados en el centro, es decir que los costos encontrados estuvieron cercanos al promedio de todos los costos.

En la *Tabla 13* al igual que el caso anterior se presenta la mejor, la peor y el promedio y desviación estándar del caso de prueba real, se ve, 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 más grande que la anterior pero si se compara con respecto a los costos es pequeña, por lo tanto la media es un valor confiable de la solución que el algoritmo produce.

En la *Tabla 12* al igual que el caso anterior se presenta la mejor, la peor, el promedio, la desviación estándar y la mediana. Se observa que en la moda más del 50% de los costos son mayores a $3 517 070, es decir que repite con mayor frecuencia para el 50% de la distribución de datos. Con respecto a la desviación estándar, esta es grande pero si se compara con respecto a los costos es pequeña, por lo tanto la media es un valor confiable de la solución que el algoritmo produce.

Tabla 13 Resultados de los costos de la hiperheurística

Medidas de tendencia central y dispersión	**Costo ($)**

No pruebas	30
Promedio	3 524 960
Desviación Estándar	142 431.30
Mejor	3 192 690
Mediana	3 517 070
Peor	3 769 720

Capítulo 6 Conclusiones y trabajos futuros

6.1. Conclusiones.

En esta investigación se determina la factibilidad y optimización de una red de distribución de hidrocarburos (caso teórico y caso de prueba) para el estado de Veracruz, México. Con base a los resultados se puede concluir lo siguiente:

- La metodología empleada permitió obtener soluciones factibles, al cumplir con el modelo de satisfacción de restricciones y permitió la reducción de costo de los cambios realizados en una red de distribución factible, al involucrar un modelo de optimización con una hiperheurística.
- Con la estrategia realizada de cambios de diámetros y agregando bombas, se satisface al 100% las presiones mínimas y máximas en cada uno de los nodos.
- Al inicio de la hiperheurística la función de aprendizaje trabajó con 27 estructuras de vecindad, de las cuales quedaron 17, debido a que generaban soluciones infactibles o no mejoraron durante la ejecución de las búsquedas locales, por tanto la función de aprendizaje las eliminó.
- Se aprecia que entre mayor número de nodos se tengan más grande es el tiempo de cómputo que se requiere para la evaluación de

factibilidad, esto se debe a que en cada nodo se tienen que evaluar las ecuaciones de continuidad y energía.

- La convergencia del algoritmo para ambas instancias fue asintótico sin embargo faltó tiempo de ejecución para que las búsquedas locales convergieran, ya que se ejecutaron 30 pruebas de 40 horas y al incrementar el tiempo de las heurísticas de bajo nivel, este tiempo fue mayor, por eso se recomienda la paralelización de los procesos y mejorar la convergencia.
- La hiperheurística fue capaz de optimizar el número de bombas, esto es de reducir de 84 bombas iniciales a 15 bombas finales para el caso de la instancia de prueba teórica y la instancia de prueba real a 27 bombas finales de 105 iniciales.

6.2. Trabajos futuros.

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.
- Paralelizar el algoritmo presentado en esta tesis ya que se solo se hace en forma secuencial. Esto para distribuir los procesos y ejecutar la búsqueda más tiempo del que se realizó y así mejorar la exploración en el espacio de soluciones.
- Hacer uso de la infraestructura Grid Morelos, utilizando varios clusters para la distribución del algoritmo para mejorar el tiempo de ejecución tanto para la instancia de prueba teórica como para la instancia de prueba real de Veracruz.

- Realizar pruebas experimentales haciendo uso del ancho de banda Infiniband con una velocidad de trasmisión de 10 $Gbps$.
- Resolver la instancia de prueba teórica y la instancia de prueba real de Veracruz mediante otra metodología, como es algoritmo genético, recocido simulado, colonia de hormigas, y otros. para ver la eficacia de la hiperheurística.
- Probar el algoritmo para otro hidrocarburo.

Bibliografía

Aho, A., Sethi, R., & Ullman, J. D. (1998). *Compiladores Principios técnicas y herramientas.* Adiidson Wesley longman de México, S. A., de CV.

Alba, E. (2005). *Parallel Metaheuristics. A New Class of Algorithms* (Vols. ISBN-10 0-471-67806-6). Canada.

Alba , E., Luque, G., & Nemaschnow, S. (2013) . *20(1)*(1–48. http://doi.org/10.1111/j.1475-3995.2012.00862.x).

Alancay, N., Villagra, S. M., & Villagra, N. (2016). Trajectory and Population Metaheuristics applied to Combinatorial Optimization Problems. *Revista de Informes Científicos y Técnicos de la Universidad Nacional de la Patagonia Austral. © 2009 Todos los Derechos Reservados.*, DOI: http://dx.doi.org/10.22305/ict-unpa.v8i1.157.

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), 04014048. https://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. https://doi.org/10.1029/WR013i006p00885

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. https://doi.org/10.1007/s11269-006-4635-3

ASME. (s.f.). *the state of mechanical engineering: Today and beyond.* ASME Research Study: 2011.

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

Ávila-Melgar E. Y., C.-C. M.-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.*

Aybar, V. A. (2013). *Control de parámetros operativos para obtimizar el funcionamiento de la red principal de transporte de y de distribución de gas natural seco del proyecto Camisea I*. Universidad Nacional de Ingeniería.

Bai, R., & Kendall, G. (2005). An Investigation of Automated Planograms Using a Simulated Annealing Based Hyper-Heuristic. En *Metaheuristics: Progress as Real Problem Solvers.* Springer.

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. https://doi.org/10.1016/j.asoc.2009.07.010

Berglund, A., Areti, V. S., Brill, D., & Mahinthakumar, G. (Kumar). (2017). Successive Linear Approximation Methods for Leak Detection in Water Distribution Systems. *Journal of Water Resources Planning and Management*, *143*(8), 04017042. https://doi.org/10.1061/(ASCE)WR.1943-5452.0000784

Borraz, C. (2004). Una metodología de solución basada en programación dinámica no secuencial y búsqueda tabú para la operación eficiente de sistemas de transporte de gas natural en estado estable. Universidad Autónoma de Nueva León.

Burke, E., Hart, E., Kendall, G., Newall, J., Ross, P., & Schulenburg, S. (2003). *Hyper-heuristics: An emerging direction in modern search technology.* In F. Glover and G. Kochenberger.

Burke, E., Kendall, G., Silva, D. L., O'Brien, R., & Soubeiga, E. (2005). An Ant Algorithm Hyperheuristic for the Project Presentation Scheduling Problem. In *2005 IEEE Congress on Evolutionary Computation* (Vol. 3, pp. 2263–2270). IEEE. https://doi.org/10.1109/CEC.2005.1554976

Burke, E. K. (2009). A Classification of Hyper-heuristic Approaches A Classification of Hyper-heuristic Approaches. *University of Nottingham Jubilee Campus Computer Science Technical Report No . NOTTCS-TR-SUB-0906241359-0664*, 1–54. http://www.cs.nott.ac.uk/~gxo/papers/hhsurvey.pdf.

Cabrera, E., Garcia Sierra, J., & Martínez, F. (1996). *Ingeniería hidráulica aplicada a los sistemas de distribución de agua.* Universidad Politécnica de Valencia.

Cabrera, E., García-Serra, J., Martinez, F., Cabrera, E., & Espert, V. (2009). *Ingeniería hidráulica aplicada a los sistemas de distribución de agua.* Universidad Politécnica de Valencia.

Cattafi, M. G. (2011). Optimal Placement of Water Distribution Network With CLP(FD). *Theory and Practice of Logic Programming*, 45(1), 41-51. http://doi.org/10.1007/BF00940812.

Castillo, J. A. (2008). *Optimization de recargas de combustible nuclear usando la técnica de buúsqueda Tabu.* México, D.F.: UNAM.

Cerny, V. (1985). Thermodynamical Approach to the Traveling Salesman Problem: An Efficient Simulation Algorithm. *Journal of Optimatization Theory and Applications.*, 45(1), 41-51. http://doi.org/10.1007/BF00940812.

Chang, J. B. (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.

Chakhlevitch, K., & Cowling, P. (2008). Hyperheuristics: Recent Developments. (3-29).

Chandramouli, S. (2019). MATLAB Code for Linking Genetic Algorithm and EPANET for Reliability Based Optimal Design of a Water Distribution Network. In *Water Resources and Environmental Engineering I* (pp. 183–194). Singapore: Springer Singapore. https://doi.org/10.1007/978-981-13-2044-6_16

Cook, W. J., Cunningham, W. H., Pulleyblank, W. R., & Schrijver, A. (1977). Heuristics for Integer Programming Using Surrogate Constraints. *Decision Sciences*, 156-166.

Cook, W. (1995). *DIMACS, Series in discretes mathematics and theoretical computer science*. (A. mathematical Society, Ed.).

Cotosa. (2015). Cotosa. Retrieved from https://www.cotosa.com.mx/

Cowling, P.; Kendall, G.; Soubeiga, E. (2000). A hyperheuristic approach for scheduling a sales summit. In Selected Papers of the Third International Conference on the Practice And Theory of Automated Timetabling. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 176–190.

Cruz-Chavez, M. A., & Diaz Parra, O. (2009a). Un Mecanismo de Vecindad con Búsqueda Local y Algoritmo Genético para el Problema de Transporte con Ventanas de Tiempo. (ISSN: 2007-3283).

Cruz-Chavez, M. A., Zavala, J. C., Mariano, C. E., Juárez, F., & Ávila, E. (2009b). Calendarización en Redes de Distribución de Agua. (52-66.http://www.gridmorelos.uaem.mx/~mcruz/paper8.pdf).

Cruz-Chávez, M. A., Zavala-Diaz, J. C., Eduardo, C., Romero, M., Juárez-Pérez, F., & Avila, E. (2009). Calendarización en Redes de Distribución de Agua. *CICos* 52–66.

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. https://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. https://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. https://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.

De Paola, F., Galdiero, E., & Giugni, M. (2016). A jazz-based approach for optimal setting of pressure reducing valves in water distribution networks. *Engineering Optimization*, *48* (5), 727 – 739. https://doi.org/10.1080/0305215X.2015.1042476

Den Besten, M., Stützle, T., & Dorigo, M. (2001). Design of Iterated Local Search Algorithms (pp. 441–451). https://doi.org/10.1007/3-540-45365-2_46

Djebedjian, B., Shahin, I., & Mohamed E.N. (2008). Gas distribution network optimization by genetic algorithm. *Ninth International Congress of Fluid Dynamics & Propulsion*, 1–19.

Edwin, P. V., Angely, V. C., Franz, C. P., Jesus, M. M., Eduardo, C. V., & Luis, A. R. (2017). Diseño Óptimo de Redes de Distribución de Agua Usando Un Software Basado En Microalgoritmos Genéticos Multiobjetivos. *Ribagua*, *4*(1), 6–23. https://doi.org/10.1080/23863781.2017.1317087

EPANET. (2016). United States Environmental Protection Agency. Retrieved October 1, 2016, from http://www.epa.gov/nrmrl/wswrd/dw/Epanet.html. United States Environmental Protection Agency.

Ewald, G., Kurek, W., & Brdys, M. A. (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, Part C (Applications and Reviews)*, *38*(4), 497–509. https://doi.org/10.1109/TSMCC.2008.923864

Fecarotta, O., Carravetta, A., Morani, M., & Padulano, R. (2018). Optimal Pump Scheduling for Urban Drainage under Variable Flow Conditions. *Resources*, *7*(4), 73. https://doi.org/10.3390/resources7040073

García Villoria, A., Salhi, S., Corominas, A., & Pastor, R. (2011). Hyper-heuristic approaches for the response time variability problem. *European Journal of Operational Research*, *211*(1), 160–169. https://doi.org/10.1016/j.ejor.2010.12.005

Garey, M. R., & Johnson, D. (1979). "*Computers and Intractability" a Guide to the Theory of NP-completeness.* New York NY.: Freeman.

Gendreau, M. (2003). *an introduccion to Tabú Search. Handbook of metaheuristic*. https://doi.org/http://doi.org/10.1007/0-306-48056-5).

Guia de Manejo Ambiental Para Estaciones de Servicio de Combustible. (septiembre de 1999). Recuperado el 24 de enero de 2019, de SCRIBD: https://es.scribd.com/document/89413483/Guia-de-Manejo-Ambiental-Para-Estaciones-de-Servicio-de-Combustible

Glover, F. (1986). Future Paths for Integer Programming and Links to Artifical Intelligence. *Computers and Operations Research*(https://doi.org/10.1016/0305-0548(86)90048-1), 533-549.

Glover, F. (1989). Tabu Search: Part I. *ORSA Journal on Computing*, 1-190.

Glover, F. (1990). Tabu Search: Part II. *ORSA Journal on Computing*, 1-4.

Glover, F. (1998). A Template for Scatter Search and Path Relinking," in Artificial Evolution. *Lecture Notes in Computer Science 1363, J.-K. Hao, E. Lutton, E. Ronald, M. Schoenauer and D. Snyers (Eds.), Springer-Verlag*, 13-54.

Glover, F., & Kochenberger, G. (2003). Handbook of Metaheuristics. *Springer*, https://doi.org/10.1007/b101874.

Glover, F., Laguna, M., & Martí, R. (1999). *to appear in Theory and Applications of Evolutionary Computation: Recent Trends.* A. Ghosh and S. Tsutsui (Eds.), Springer-Verlag.

Glover, F., Laguna, M., & Martí, R. (2000). Fundamentals of Scatter Search and Path Relinking. *Control and Cybernetics*, 653-684.

Glover, F., Laguna, M., Taillard, E., & de Werra, D. (1993). A user's guide to tabu search. *Annals of Operations Research*, 3-28.

Glover F., & Kochenberger, G. (2003). Handbook of Metaheuristics. *Kluwer Academic Publishers*, https://doi.org/10.1007/b101874.

Goldberg, D., & Holland, J. (1989). Classifier Systems and Genetic Algorithms. *Artificial Intelligence*, 235-282.

google. (2019). Coordenadas-gps. Retrieved from https://www.coordenadas-gps.com/

Guistolisi, O., & Todini, E. (2009). Pipe hydraulic resistance correction in WDN analysis. (39-52).

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

Hoos, H. H., & Shutzle, T. (1975). Stochastic Local Search: Foundations and Applications. Stochastic Local Search Foundations and Applications. *3777*(126–169. http://dl.acm.org/citation.cfm?id=983505).

Horst, R., Pardalos, P., & Thoai, N. (1995). *Introducction to global optimization.* Holanda: kluwer dordrecht.

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

Hurtado-Guzmàn, V. H. (2009.). Algoritmos Genèticos y Epanet 2.0 para la Localizacion Optima de Valvulas Reductoras de Presion en Redes dde Distribucion de Agua Potable. *Universidad Autonoma de Mexico.*, http://www.ptolomeo.unam.mx:8080/xmlui/bitstream/handle/132.248.52.100/1153/Tesis.pdf?sequence=1.

Horts, R., & Pardalos, P. M. (2013). *Handbook of Global Optimization* (Springer S).

ITA. (2015). Cursos online de EPANET Una formación a tu medida. Retrieved from https://cursosagua.net/modelacion/epanet/index-es.php

Jacobson, S. H., & Yücesan, E. (2004). Analyzing the Performance of Generalized Hill Climbing Algorithms. *Journal of Heuristics*, *10*(4), 387–405. https://doi.org/10.1023/B:HEUR.0000034712.48917.a9

Joyanes, A. L., & Zahonero, M. (2004). *Algoritmos y estructuras de datos. Una perspectiva en C.* España: Mc Graw Hill.

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*, *136*(4), 463–473. https://doi.org/10.1061/(ASCE)WR.1943-5452.0000056

Kheiri, A., Keedwell, E., Gibson, M. J., & Savic, D. (2015). Sequence Analysis-based Hyper-heuristics for Water Distribution Network Optimisation. *Procedia Engineering*, *119*, 1269–1277. https://doi.org/10.1016/j.proeng.2015.08.993

Kirk, E., & Othmer, F. (2002). *Enciclopedia de tecnología química / [bajo la dirección de] Kirk-Othmer*. (E. Limusa, Ed.).

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

Kolonko M., T. T. (1997). Convergente of Simulated Annealing with Feedback Temperatura Schedule. *Problem in The Engineeering and Informational Sciences*, 11, 279-304.

Kowalska, B., Holota, E., & Kowalski, D. (2018). Simulation of chlorine concentration changesin real water supply network using EPANET 2.0 y waterGEMS software packages. In *Urban Water Systems & Floods* (Vol. 184, pp. 39–48). https://doi.org/10.2495/FRIAR18004

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. https://doi.org/10.1016/j.jenvman.2012.11.030

Labrada, Y., Enríquez, J., & García, Y. (2014). Un algoritmo de búsqueda local iterada como solución al problema de la mochila. *Programación Matemática y Software*, 57-64.

Laguna, M., Barnes, J. W., & Glover, F. (1993). Intelligent scheduling with tabu search: An application to jobs with linear delay penalties and sequence-dependent setup costs and times. *Applied Intelligence*. https://doi.org/10.1007/BF00871895

Laguna, M., Barnes, J. W., & Glover, F. W. (1991). Tabu Search Methods for a Single Machine Scheduling Problem. *Journal of Intelligent Manufacturing*, 63-74.

Laurenco , H. R., Martin, O., & Shutzle, T. (2002). *Handbook of Metaheuristic, chapter Iterated local search* (Vols. 321-353). Kluwer Academic Publishers.

Lee, E. H., Lee, H. M., Yoo, D. G., & Kim, J. H. (2018). Application of a Meta-heuristic Optimization Algorithm Motivated by a Vision Correction Procedure for Civil Engineering Problems. *KSCE Journal of Civil Engineering*, *22*(7), 2623–2636. https://doi.org/10.1007/s12205-017-0021-3

Lei, T. G. (2011). Desing of Water Distribution Network via ANt Colony Optimization. *In The 2nd Iternational Conference on Intelligent Control and Processing.*, (pp. 366-370). IEEE Computer Society.

Liberatore, S. &. (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. D. (2006). 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.

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. https://doi.org/10.1016/S1004-9541(08)60026-9

Lourenço, H. R., Martin, O. C., & Stützle, T. (2003). Chapter 11, ITERATED LOCAL SEARCH. *Springer Book*.

Martinez-Bahena, B. (2011). *Solución al problema del árbol de expanción mínima aplicando Recocido Simulado con Busqueda Tabu.* cuernavaca: CIICAp.

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. https://doi.org/10.1109/CERMA.2012.38

Martínez-Bahena, B. (2016). *Algoritmo genético distribuido, para optimizar la red hidraulica del Fraccionamiento Real de Montecasino del minucipio de Huitzilac.* cuernavaca: UAEM.

Martínez-Bahena, B. C.-C.-M.-R.-L. (2018). Using a genetic algorithm with a mathematical programming solver to optimize a realwater distribution system. *Water (Switzerland)*, 10(10). https://doi.org/10.3390/w10101318.

Martinez-Oropeza, A. (2010). *Solución al problema de máquinas en paralelo no relacionadas mediante un algoritmo de colonia de hormigas*. Universidad autónoma del estado de Morelos.

Martínez-Rangel, M. (2008). Algoritmo de Recocido simulado paralelizado, aplicado al problema de asignación de recursos en un taller de manufactura flexible sujeto a disposiciones de tiempo. https://doi.org/10.1161/01.RES.0000056770.30922.E6.

McClymont, K., Keedwell, E. C., Savić, D., & Randall-Smith, M. (2014). Automated construction of evolutionary algorithm operators for the bi-objective water distribution network design problem using a genetic programming based hyper-heuristic approach. *Journal of Hydroinformatics*, *16*(2), 302–318. https://doi.org/10.2166/hydro.2013.226

Melián, B., & Pérez, J. A. (2003). Metaheurísticas: una visión global. *Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial*, 7-28 ISSN: 1137-3601.

Menon, E. S. (2005). *Piping Calculations Manual*. Sydney, Toronto, Mc graw Hill.

Mexico.pueblosamerica. (2019). Pueblos de América. Retrieved from https://mexico.pueblosamerica.com/i/santa-rita-195/

Mostafaei, H., Castro, P., & Ghaffari-Hadigheh, A. (2015). A Novel Monolithic MILP Framework for Lot-Sizing and Scheduling of Multiproduct Treelike Pipeline Networks. *industrial & Engineering Chemistry Research*, 9202–9221, DOI: 10.1021/acs.iecr.5b01440.

Mott, R. .. (2006). *Mecánica de fluidos* (sexta ed.). Pearson Prince Hall.

Nicolini, M. &. (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/10.1016/(ASCE)0733-9496(2009)135:(178).

Nurani Zulkiflia, S., Abdul Rahima, H., & Jye Lau, W. (2015). Detection of contaminants in water supply: A review on state-of-the-art monitoring technologies and their applications. (1269 – 1277).

Oliveira, E. C. M., Melo Brentan, B., Danta, R. F., dos Santos Macedo, L., Luvizotto Junior, E., & Ribeiro, L. C. L. J. (2018). Detection of chemical intrusion compounds in water distribution networks by quality sensors data. In *1stInternational WDSA / CCWI 2018* (pp. 23–25). Ontario, Canada.

Ortiz Huerta, A. (2010). *Análisis numérico de transferencia de calor y masa.* Temixco: CENIDET.

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. https://doi.org/10.1061/(ASCE)WR.1943-5452.0000407

Oteiza, P. P., De Meio Reggiani, M. C., Rodriguez, D. A., Viego, V., & Brignole, N. B. (2015). Metaheuristic Techniques for the Optimal Design of NGL Pipelining. In K. V Gernaey, J. K. Huusom, & R. Gani (Eds.), *12th International Symposium on Process Systems Engineering and 25th European Symposium on Computer Aided Process Engineering,* Vol. 37, pp. 785–790. Elsevier. https://doi.org/10.1016/B978-0-444-63578-5.50126-2

Oteiza, P. P., Rodríguez, D. A., & Brignole, N. B. (2018). Parallel Hyperheuristic Algorithm for the Design of Pipeline Networks. *Industrial & Engineering Chemistry Research*, *57*(42), 14307–14314. https://doi.org/10.1021/acs.iecr.8b02818

Papadimitriou, C., & Steiglitz, K. (2013). *Combinatorial Optimization: Algorithms and Complexity.* Dover Books on Computer Science.

Pecci, F., Abraham, E., & Stoianov, I. (2015). Mathematical Programming Methods for Pressure Management in Water Distribution Systems. *Procedia Engineering*, *119*, 937–946. https://doi.org/10.1016/j.proeng.2015.08.974

PEMEX. (2006). Especificaciones Técnicas 2006. Retrieved from http://www.pemex.com/franquicia/incorporacion-operacion/Documents/EspTec2006Conduccion.pdf

Petroleos Mexicanos. (2015). Obtenido de www.institutodelpetroleo.mx

Puleo, V., Morley, M., Freni, G., & Savić, D. (2014). Multi-stage Linear Programming Optimization for Pump Scheduling. *Procedia Engineering*, *70*, 1378–1385. https://doi.org/10.1016/j.proeng.2014.02.152

Poling, B., Prausnitz, J., & O´connell, J. (2001). *The properties of gases and liquids.* Mc Graw Hill.

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*. https://doi.org/10.1007/s11269-007-9230-8

Reis, L. F. R., Porto, R. M., & Chaudhry, F. H. (1997). Optimal Location of Control Valves in Pipe Networks by Genetic Algorithm. *Journal of Water Resources Planning and Management*, *123*(6), 317–326. https://doi.org/10.1061/(ASCE)0733-9496(1997)123:6(317)

Ríos M., R. Z., & Borraz Sánchez, C. (2009). Mejorando el transporte de gas natural mediante un método híbrido de búsqueda tabú y programación dinámica. *Ingenierías*, 54-65.

Rochat, Y., & Taillard, D. (1995). Probabilistic diversification and intensification in local search for vehicle routing. *Journal of heuristics*, 147-167.

Rossman, L. (1999). *Computer Models/EPANET in L. Mays.* New York: Water Distribution Systems Handbook.

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

Rossman, L. (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.

Saldarriaga, J. (2010). *Hidraulica de tuberías.* Mc Graw Hill.

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. https://doi.org/10.1016/j.proeng.2015.08.986

Savic, D. A. (1997). Genetic Algorithms for Least-Cost Design of Water. *Journal of Water Resources Planning and Management*, 67–77. https://doi.org/10.1061/(ASCE)0733-9496(1997)123

Shamir, U. (1974). Optimal Design and Operation of Water Distribution Systems. *Water Resources Research*, *10*(1), 27–36. https://doi.org/10.1029/WR010i001p00027

Shu, S. &. (2010). Calibrating Water Distribution System Model Automatically by Genetic Algorithms. *International Conference on Intelligent Computing and Integrated Systems.*, ICISS2010, 16-19. http://doi.org/10.1109/ICISS.2010.5654995.

Stützle, T. (2019). *Local search algorithms for combinatorial problems : analysis, improvements, and new applications*. Alemania.

Soubeiga, E. (2003). *Development and application of hyperheuristics to personnel scheduling.* University of Nottingham.

Talbi, E.-G. (2002). *A Taxonomy of Hybrid Metaheuristics*. *Journal of Heuristics* (Vol. 8). https://doi.org/10.1023/A:1016540724870

Talbi, E.-G. (2009). *Metaheuristics*. Hoboken, NJ, USA: John Wiley & Sons, Inc. https://doi.org/10.1002/9780470496916

Termini, D., & Viviani, G. (2015). Spatial diversity of chlorine residual in a drinking water distribution system: application of an integrated fuzzy logic technique. *Journal of Hydroinformatics*, *17*(2), 293–306. https://doi.org/10.2166/hydro.2014.092

Todini, E., & Pilati, S. (1987). A gradient algorithm for the analysis of pipe networks. In *Computer Applications in Water Supply. Volume 1 System Analysis and Simulation*.

Tubacero. (2015). No Title. Retrieved from http://www.tubacero.es/resources/upload/link/tubacero_catalogo_general.pdf

Tucciarelli, T., & Termini, D. (1998). Optimal valves regulation for calibration of pipe networks models. *In HYDROINFORMATICS '98, 1*, 1029–1036.

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. https://doi.org/10.1061/(ASCE)0733-9496(2010)136:2(279)

Villela Tinoco, J. C., & Coello Coello, C. A. (2013). hypDE: A Hyper-Heuristic Based on Differential Evolution for Solving Constrained Optimization Problems, Springer, 267–282. https://doi.org/10.1007/978-3-642-31519-0_17

Valiente, A. (2002). *Problemas de flujo de fluidos* (segunda edicion ed.). LIMUSA.

Van dijk, M. V. (2008). Optimising Water Distribution Systems Using a Weighted Penalty in a Genetic Algorithm. *Water SA.*, 34(5), pp. 537-548.

Vázquez, J. A., & Petrovic, s. (2010). A new dispatching rule based genetic algorithm for the multi-objective job shop problem. *Springer Science+Business Media*, pp. 771-793.

Vicentini, F., & Puddu, S. (2003). *Algoritmos heurísticos y el problema job shop scheduling.* Buenos Aires: Universidad de Buenos Aires.

Wang, Y., Puig, V., & Cembrano, G. (2017). Non-linear economic model predictive control of water distribution networks. *Journal of Process Control*, *56*, pp. 23–34. https://doi.org/10.1016/j.jprocont.2017.05.004

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), pp. 629–636. https://doi.org/10.1061/(ASCE)WR.1943-5452.0000073

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

White, F. M. (2009). *Mecânica dos Fluidos*. *Book*. https://doi.org/10.1111/j.1549-8719.2009.00016.x.Mechanobiology

Wolpert, D., & W., M. (2004). No free lunch theorems for optimization. *IEEE transactions on evolutionary computation 1*, pp. 67-82.

Wong, P., & Larson, R. (1968). Optimization of natural-gas pipeline systems via dynamic programming. *IEEE Transactions on Automatic Control*, *13*(5), 475–481. https://doi.org/10.1109/TAC.1968.1098990.

Zhang, H., Huang, T.-L., & He, W.-J. (2009). Application of Heuristic Genetic Algorithm for Optimal Layout of Flow Measurement Stations in Water Distribution Networks. In *2009 Fifth International Conference on Natural Computation*, 140–143. IEEE. https://doi.org/10.1109/ICNC.2009.513

Zulkifli, S. N., Rahim, H. A., & Lau, W.-J. (2018). Detection of contaminants in water supply: A review on state-of-the-art monitoring technologies and their applications. *Sensors and Actuators B: Chemical*, *255*, pp. 2657–2689. https://doi.org/10.1016/j.snb.2017.09.078

Apéndice A

Costo de los diámetros y válvulas de las instancias.

Diámetro (*mm*)	Costo Tuberías ($)	Costo Válvulas ($)	Diámetro (*mm*)	Costo Tuberías ($)	Costo Válvulas ($)
25.4	100	101.5	279.4	1266	1100.5
38.1	115	149.2	292.1	1586	1156
50.8	130	200.2	304.8	1600	1209
63.5	133	250	317.5	1869	1250
76.2	189	300	330.2	1562.3	1309
88.9	250	355.6	342.9	1638.9	1350.8
101.6	300	405.3	355.6	1856.9	1399.9
114.3	300	451.2	368.3	1900	1449.9
127	399	505.3	381	1910.9	1500
139.7	500	551.2	393.7	2056.8	1550
152.4	550	600.9	406.4	2068.5	1593.2
165.1	600	655.3	419.1	2698.3	1650
177.8	660	717	431.8	3005.8	1700
190.5	680.6	750	444.5	3265.9	1749.9
203.2	658	800	457.2	3458.2	1800
215.9	698.6	850.5	469.9	3566.2	1850
228.6	750.5	900	482.6	3666	1900.87
241.3	800.9	949	495.3	4999	1950
254	900.6	1010	508	6000	1995.8
266.7	951.6	1050.9			

Costo de las bombas de acuerdo a su potencia nominal.

Potencia	Costo ($)
0.093	1005
0.186	1988
0.249	3999
0.559	4015.2
0.746	5000.9
1.119	6098.6
1.491	7000.9
1.864	8089.2
2.237	8999
2.983	9999.9
4.101	11000
5.593	12987
7.457	13000.9
9.321	14009
11.186	15000.8
12.677	15500
14.914	16005

Apéndice B

Código fuente del algoritmo

```
#include "toolkit.h"
 #include <stdio.h>
 #include <math.h>
 #include <string.h>
 #include <stdlib.h>
 #include <time.h>
 #define NNODOS               1000
 #define NTUBERIAS            1000
 #define NDEPS                0
 #define NEMBS                1
 #define NBOMBASComer         17
 #define NDIAMCOMERS          39
 #define DIAMENOR             12.7
 #define PRESIONMAX           1600
 #define PRESIONMIN           30
 #define NCONSIGNA            9
 #define Pres_
          patin_recoleccion   700
 #define Pres_
          estacion_recoleccion700
 #define DemandaDeposito      2
 #define TABU1                10
 #define HEURISTICAS          54
 #define NCOMPONENTES         2
 #define CoeRug               120
 #define Units                "LPS"
 #define Headloss             "D-W"
 #define Specific_Gravity     0.68234686
 #define Viscosity            0.4758288
 #define Trials               40
 #define Accuracy             0.001
 #define Unbalanced           "Continue 10"

typedef struct{
               int NTuberia;
               int Vorigen;
               int Vdestino;
               float diametro;
               float longitud;
               float caudal;
               }tuberia;
typedef struct{
               int idnodo;
               float cota;
               float demanda;
               float coordX;
               float coordY;
               float presion;
               }nodo;
typedef struct{
               int id_dep;
               float altura;
               float coordX;
               float coordY;
```

```
                }embalse;
typedef struct{
                int id_dep;
                float altura;
                float NivelIni;
                float NivelMin;
                float NivelMax;
                float diametro;
                float coordX;
                float coordY;
                }deposito;
typedef struct{
                int id_bom;
                int nodoAsp;
                int nodoImp;
                float potencia;
                int ExisteBomba;
                }bomba;
typedef struct{
                int id_val;
                int H2Oarriba;
                int H2Oabajo;
                int estado;
                float consigna;
                int perdidas;
                int ExisteValvula;
                int Ntuberia;
                }valvula;
typedef struct{
                int id_Diam;
                float CostoDiam;
                float Diam;
                }Diametro;

tuberia  TUBERIAS            [NTUBERIAS];
bomba    BOMBAS              [NTUBERIAS];
valvula  VALVULAS            [NTUBERIAS];
nodo     NODOS               [NNODOS];
embalse  EMBALSES            [NEMBS];
deposito DEPOSITOS           [NEMBS];
Diametro DIACOMER            [NDIAMCOMERS];
tuberia  TUBERIAS_Inicial    [NTUBERIAS];
bomba    BOMBAS_Inicial      [NTUBERIAS];
valvula  VALVULAS_Inicial    [NTUBERIAS];
nodo     NODOS_Inicial       [NNODOS];

 int    TABU=TABU1,

        Evaluar_factibilidad   (),
        RunEPANET              (),
        buscar_tuberia_Vdestino(),
        Evaluacion_EPANET      (int),
        LSolver_EPANET         (int, int),
        contar_datos           (int *,int),
        Elite                  (int*,int*,int* ),
        Torneo                 (int*,int*,int* ),
        ListaTABU              (int, float *,int),
        Boltzman               (float, float, float),
        buscar_tuberia         (int, int *,int),
        buscar_En_Vector       (float, float *,int),
        Ruleta                 (int*,double,int*,int*),
        Vecindad_Bombas        (int,int,int*,int,int),
        Vecindad_Valvulas      (int,int,int*,int,int),
        CambioDiametro         (int *,int,int,int,int),
        Estructura_Vecindad    (int*,int,int,int*,int*),
        Partition              (float*, float*,int, int ),
        HeuristicasBajoNivel   (int,int ,int*,int,int *,int*),
        SelecHeuristiBajoNivel (int,int,int,int, int,double),
```

```
        CombinacionesVecindad  (int,int,int,int,int,int,int),
        EleccionValBom         (int,int,int,int,int,int,int),

void
        CargarValvulasBombas     (),
        DesactivarBombasValvulas (),
        ActivarBombasValvulas    (),
        ConfiguracionInicial     (),
        Agregar_Valvulas         (),
        Agregar_Bombas           (),
        Agregar_Tuberias         (),
        Hiperheuristica          (),
        InicioDiametro           (),
        PonerBomba               (),
        sintonizar               (),
        Save_Best_Solution       (),
        Actualizar_Val_bom       (),
        Cargar_Archivos          (),
        insertar_val             (),
        Crear_archivo_VALVULAS2  (),
        Presion_Embalse_Deposito (),
        chek_Factible_Cruzamiento       (int),
        Seleccion_Cruzamiento           (int),
        ActualizarListaHeuristic        (int),
        Solucion_Anterior               (int*),
        Factibilidad                    (int),
        solucion_diametro_tuberia       (int),
        evaluar_presiones               (int),
        tuberias                        (int),
        LeerConfiguracion_EPANET        (int),
        Estructura_Vecindad_Factibilidad(int),
        ordenarHeurist           (double*),
        ChecaHistory             (float),
        SaveCruzadoEstruc        (float*),
        inicializar_Costo        (float*),
        EvaluaCostoTuberia       (float*),
        Activar_Cruzamiento      (float*),
        Crear_archivo_INP1       (int,int),
        Cruzamiento              (int,int),
        Cargar_datos             (char*,int),
        EleccionVecindad         (int*,int),
        Acomodar                 (float *,int),
        Swap                     (float*,float*),
        guarda_archivo_txt       (float, float,char*),
        Function                 (int, int, int, float,int),
        QuickSortIterative       (float*,int,float,double*),
        Selec_Ruleta_Elite_Torneo(int*,int*,int,double,int*,int*),
        Guardar_Solucion_Inicial (double*,double*,int*,int*,double*,int*,int*,int*),



double sumaCosto(),
float
        evaluacion_soluciones    (),
        CambioBomba              (int),
        busqueda_Local           (int,int,int),
        busqueda_Local_iterada   (int,int,int,int),
        GuardarTiempo            (time_t,time_t,int);

float   Solucion1             [NTUBERIAS*3],
        Solucion2             [NTUBERIAS*3],
        Solucion_Cruzada1     [NTUBERIAS*3],
        Solucion_Cruzada2     [NTUBERIAS*3],
        BombasComerCosto      [NBOMBASComer],
        BombasComerPotencia   [NBOMBASComer],
        ValvulasComerCosto    [NDIAMCOMERS],
        CambiosTABU_Diametros [TABU1*4*3]={0},
```

```
        CambiosTABU_Bombas    [TABU1*4*3]={0},
        CambiosTABU_Valvulas  [TABU1*4*3]={0},
        HeuristicRun          [HEURISTICAS*8]={0},
        HeuristicHistory      [HEURISTICAS*2+2]={0},
        DiamComerCosto        [NDIAMCOMERS][NCOMPONENTES],

int     TABU_Ejecutadas       [TABU1],
        BombasComerID         [NBOMBASComer],
        ValvulasComerID       [NDIAMCOMERS],
        nodosPRESNEG          [NNODOS],
        nodosPRESPOS          [NNODOS],
        nodosPRESok           [NNODOS],
        tuberiaPRESPOS        [NNODOS],
        StruValPos            [NNODOS],
        tubs                          [NTUBERIAS],
        Bombas_Red                    [NTUBERIAS],
        Valvulas_Red                  [NTUBERIAS],
        Diametros_Red                 [NTUBERIAS],
        tuberiaPRESPOSOk              [NTUBERIAS],
        tuberiaPRESNEG                [NTUBERIAS],
        ValoresConsiga                [NCONSIGNA],
        TuberiaPosNeg                 [NTUBERIAS],
        tuberiaPRESPOSDia             [NTUBERIAS],
        TuberiaNegPosValulas          [NTUBERIAS],
        TuberiaNegPosDiametro         [NTUBERIAS],
        tuberiaPRESPOSDia_Vdestino    [NTUBERIAS],
        Soluciones            [NSOLS][NTUBERIAS],
        NSoluciones           [NSOLS][NTUBERIAS],

int     TotalHeuris=0,
        NTubNegPosVal=0,
        NTubNegPosDia=0,li=0,a,b,
        NoFactible=0,SOL=0,cambios=1;
        NTotalBom=0,NTotalVal,
        NVALVULAS=0, NBOMBAS=0,
        bandera,NHeuristic;
float Mejor_costo=0.0,costo_Factible=0.0;

int main()
{       int i;
        time_t inicio,final;
        float tiempo;
                inicio=time(NULL);
                        Factibilidad(1);
                        GuardarTiempo(inicio,final,1);
                        Hiperheuristica();

                final= time(NULL);
                        tiempo=GuardarTiempo(inicio,final,1);
        ENclose();
        ENcloseH();
return 0;
}
/*****************************************************************************/
/**************************AREA DE FUNCIONES**********************************/
/*****************************************************************************/

void Factibilidad(int flag){
int     i;
        NTotalVal=0;NTotalBom=0;
if (flag==1) ConfiguracionInicial();
else DesactivarBombasValvulas();
RunEPANET();
        while (Evaluar_factibilidad()==0){
                srand(time(NULL));
                DesactivarBombasValvulas();
                RunEPANET();}
                Actualizar_Val_bom();
```

```
}

/*****************************************************************************/
/********************** CARGAR DATOS A LAS ESTRUCTURAS ***********************/
/*****************************************************************************/
void ConfiguracionInicial(){
 int    Nval=0,cont=0;
                Cargar_Archivos();
                Agregar_Tuberias();
                Agregar_Valvulas();
                Agregar_Bombas();
                LeerConfiguracion_EPANET(0);
                DesactivarBombasValvulas();
 }
/*****************************************************************************/
/******************************* HIPERHEURISTICA *****************************/
/*****************************************************************************/
void Hiperheuristica()
{
int     i,cont=0,Eleccion;

        iteraciones                   =22;
        Tiempo_iteraciones            =600;
        TiemporepetirBusqueLocal      =60
        Ruleta_Elite_Torneo           =0;
        Repetir_Cruzamiento           =2;
        repetirBusqueLocal            =10;
        TamVecindad                   =1000;
        TamVecindadIte                =15;
        repeticion=numeros_aleatoriosLsLi(9,1);

float   reducir=99999999;
        tiempo=0;
        final;
        inicio=time(NULL);
double SumCosto=0;

                EvaluaCostoTuberia(&Mejor_costo);

         for(i=0;tiempo<=Tiempo_iteraciones;i++){
        if (SelecHeuristiBajoNivel(TamVecindad,TamVecindadIte,
                repetirBusqueLocal,TiemporepetirBusqueLocal,Ruleta_Elite_Torneo,SumCost
        o)==0)
                cont++;
        else cont=0;
        if (cont==Repetir_Cruzamiento){cont=numeros_aleatoriosLsLi(1,0);
        if (cont==1){Seleccion_Cruzamiento(repeticion);
chek_Factible_Cruzamiento(repeticion);}
cont=0;
                }
        Ruleta_Elite_Torneo=numeros_aleatoriosLsLi(1,0);
        QuickSortIterative(HeuristicRun, TotalHeuris,reducir,&SumCosto);

        ordenarHeurist(&SumCosto);tiempo=GuardarTiempo(inicio,final,1);    }
}

/*****************************************************************************/
/************************ CHECA FACTIBILIDAD EN CRUZAMIENTO ******************/
/*****************************************************************************/
void chek_Factible_Cruzamiento(repeticion)
{
int Factible1=1,Factible2=1,cont=0;

                Factible1       =0,
                Factible2       =0;

                Activar_Cruzamiento(Solucion_Cruzada1);
                Factible1=RunEPANET();
```

```
        if(Factible1==1)
                SaveCruzadoEstruc(Solucion_Cruzada1);
        else{
                Activar_Cruzamiento(Solucion_Cruzada2);
                Factible2=RunEPANET();
        if(Factible2==1)
                SaveCruzadoEstruc(Solucion_Cruzada2);
        else
                Factibilidad(2);
                        } }

/*****************************************************************************/
/***************************** HIPERHEURISTICA *******************************/
/*****************************************************************************/

 void Seleccion_Cruzamiento(int repeticion)
{
int flag;
        if(Solucion1[NTUBERIAS*3-1]>0&&Solucion2[NTUBERIAS*3-1]>0)
                flag=numeros_aleatoriosLsLi(1,0);


        else if (Solucion1[NTUBERIAS*3-2]>0&&Solucion2[NTUBERIAS*3-2]>0){
                flag=numeros_aleatoriosLsLi(1,0);
        if (flag==1) flag++;
  }

        else
                flag=numeros_aleatoriosLsLi(2,0);
                cruzamiento(flag,repeticion);
 }
/*****************************************************************************/
/*********************SELECCIONADOR DE HEURISTICA DE BAJO NIVEL*****************/
/*****************************************************************************/
int SelecHeuristiBajoNivel(int TamVecindad, int TamVecindadIte,int repetirBusqueLocal,
                           int TiemporepetirBusqueLocal,int Ruleta_Elite,double
                       SumCosto){

int     flag=0,i,j,Eleccion,posición,
        iterada=0,tabu=0, Costo;

float  inicio=time(NULL),
        final,float tiempo=0;

        Guardar_Solucion_Inicial(TUBERIAS_Inicial ,BOMBAS_Inicial,VALVULAS_Inicial,
                                 NODOS_Inicial,TUBERIAS ,BOMBAS,VALVULAS ,NODOS);
        EvaluaCostoTuberia      (&Costo);
        costo_Factible        =Costo;

  for(i=0;tiempo<=TiemporepetirBusqueLocal;i++){
                Guardar_Solucion_Inicial(TUBERIAS,BOMBAS,VALVULAS,NODOS,
                                         TUBERIAS_Inicial,BOMBAS_Inicial,
                                         VALVULAS_Inicial,NODOS_Inicial);
                ActivarBombasValvulas   (Eleccion);
                Ruleta_Elite=numeros_aleatoriosLsLi(5,0);
        Selec_Ruleta_Elite_Torneo(&Eleccion,&posicion,Ruleta_Elite,SumCosto,&iterada,&
tabu);

        if(i==repetirBusqueLocal/2)
                iterada=1;
        else iterada=0;

                  flag=
HeuristicasBajoNivel(TamVecindad,TamVecindadIte,&Eleccion,posicion,&iterada,&tabu);

          for(j=TABU;j>0;j--)
                TABU_Ejecutadas[j]=TABU_Ejecutadas[j-1];
                TABU_Ejecutadas[0]=Eleccion;
```

```
              tiempo=GuardarTiempo(inicio,final,1);  }
              ActualizarListaHeuristic(i);
return(flag);
}
/*****************************************************************************/
/*******************SELECCIONADOR DE HEURISTICA DE BAJO NIVEL*****************/
/*****************************************************************************/
void ActualizarListaHeuristic(int repetirBusqueLocal){
  int i;
       for(i=0;i<NHeuristic;i+=2)
              HeuristicRun[i]/=repetirBusqueLocal;
}
/*****************************************************************************/
/*********************SELECCIONADOR DE RULETA, ELITE O TORNEO*****************/
/*****************************************************************************/

void Selec_Ruleta_Elite_Torneo(int *Eleccion,int *posicion,int flag1,double
SumCosto,int *iterada,int *tabu){
                if (flag1==0)*posicion=Elite(Eleccion,iterada,tabu);
                else *posicion=ruleta(Eleccion,SumCosto,iterada,tabu);
 }
/*****************************************************************************/
/*********************HEURISTICAS DE BAJO NIVEL*******************************/
/*****************************************************************************/
int HeuristicasBajoNivel(int TamVecindad,int TamVecindadIte,int *Eleccion,
                         int posicion,int *Local_iterada,int *tabu)
{  int flag=0,i,j,ejecutados=0,flag1=0, mejora=0,aleatorio;
   float Costo;

       if(*Eleccion>=3&&*Eleccion<=10){TamVecindad=NBOMBASComer*NTotalBom;}
       else TamVecindad=1000;

       if(*Local_iterada==0){mejora=busqueda_Local(TamVecindad,*Eleccion,*tabu);
              Costo=Mejor_costo;
                    }
       else {TamVecindadIte=numeros_aleatoriosLsLi(10,2);
              mejora=busqueda_Local_iterada(TamVecindadIte,TamVecindad,*Eleccion,*tab
              u);
              Costo=Mejor_costo;
              }

       if(Costo<Mejor_costo){Mejor_costo=Costo;flag=1;
                     }
              HeuristicRun[posicion-1]+=Costo;
              HeuristicHistory[*Eleccion*2-2]+=mejora;
              HeuristicHistory[*Eleccion*2-1]++;
       if(*tabu==1){
       if(mejora==-2){
       if(TABU==1)HeuristicHistory[HEURISTICAS*2+1]=posicion;
              mejora++;
                              }
              HeuristicHistory[HEURISTICAS*2-1]+=mejora;
              HeuristicHistory[HEURISTICAS*2]++;
                  }

       if(ejecutados<10)ejecutados++;else ejecutados=0;    return(flag1);
       }
/*****************************************************************************/
/********************** GUARDA LAS MEJORES SOLUCIONES ************************/
/*****************************************************************************/
void Save_Best_Solution()
{int i,j;
       for(i=0;i<NTUBERIAS*3;i++){
                                     Solucion2[i]=Solucion1[i];
                                     Solucion1[i]=0;}

       for(i=0;i<NTUBERIAS;i++)
                                     Solucion1[i]=  TUBERIAS[i].diametro;
```

```
                                        j=NTUBERIAS;
        for(i=0;i<NTotalBom;i+=1){
                                        Solucion1[j]=  BOMBAS[Bombas_Red[i]-1].potencia;
                                        Solucion1[j+1]=Bombas_Red[i];
                                        j+=2;
                }
                                        j=NTUBERIAS+NTotalBom;
        for(i=0;i<NTotalVal;i+=1){
                                        Solucion1[j]=  VALVULAS[Valvulas_Red[i]-
                1].consigna;
                                        Solucion1[j+1]=Valvulas_Red[i];
                                        j+=2;
                }

                Solucion1[NTUBERIAS*3-1]=NTotalBom;
                Solucion1[NTUBERIAS*3-2]=NTotalVal;
        }
/*****************************************************************************/
/**************************BUSQUEDA LOCAL *************************************/
/*****************************************************************************/
float busqueda_Local(int TamVecindad,int flag,int tabu)
{ int   Factible,i,cont2=0,cont=0,
        Eleccion,mejora=0,flag1=0;int flagtabu=0,
        Cancela_tabu=0, bandera44[3];
float   costo;
cambios=1;

        for(i=1;i<=TamVecindad;i++){
                EleccionVecindad(&Eleccion,flag);
                cambios=numeros_aleatoriosLsLi(1,1);
                Estructura_Vecindad(&i,Eleccion,tabu,&flagtabu,bandera44);
                Factible=RunEPANET();
                cont++;
                Cancela_tabu+=flagtabu;
                Actualizar_Val_bom();
        if(Factible!=1){i--;cont2++;
                Solucion_Anterior(bandera44);
                Factible=RunEPANET();
                Actualizar_Val_bom();
                                }
        else {cont2=0;
                EvaluaCostoTuberia(&costo);
        if (costo<Mejor_costo){
                Crear_archivo_INP1(SOL,0);
                Save_Best_Solution();
                Mejor_costo=costo;
                mejora=1;
                flag1=1;
                        }
        else {
        if(flag1==1)mejora=1;
        else if(flag1==0&&Cancela_tabu==cont)mejora=-2;
        else if(flag1==0&&Cancela_tabu!=cont)mejora=0;

                Solucion_Anterior(bandera44);Actualizar_Val_bom();
                flag1=2;
                }       }
        if(cont2==NTUBERIAS/3){
        if(flag1==1) mejora=1;
        else if(flag1==2&&Cancela_tabu==cont)mejora=-2;
        else if(flag1==2&&Cancela_tabu!=cont)mejora= 0;
        else if(flag1==0&&Cancela_tabu==cont)mejora=-2;
        else if(flag1==0&&Cancela_tabu!=cont)mejora=-1;
                return(mejora);
                }       }
                return(mejora);
        }
/*****************************************************************************/
```

```
/***********************BUSQUEDA LOCAL ITERADA *********************************/
/*******************************************************************************/
float busqueda_Local_iterada(int TamVecindadIte,int TamVecindad,int flag,int tabu)
{int     j,cont2=0,mejora=0,flat=0;
 float   costo;
 double  Costo_Anterior= Mejor_costo*10000;
 costo_Factible=Mejor_costo*10000;
       for(j=1;j<=TamVecindadIte;j++){
              Factibilidad(0);
              EvaluaCostoTuberia(&costo);
       if (numeros_aleatoriosLsLi(3,0)==1){
              Guardar_Solucion_Inicial(TUBERIAS_Inicial,
              BOMBAS_Inicial,VALVULAS_Inicial ,NODOS_Inicial,
              TUBERIAS ,BOMBAS,VALVULAS ,NODOS);
              flat=1;}
              busqueda_Local(TamVecindad,flag,tabu);
       if (Mejor_costo<Costo_Anterior){
              cont2=0;mejora=1;
              Costo_Anterior=Mejor_costo;
       }
       else {
             cont2++;
             mejora=0;
             }
       if(cont2==TamVecindadIte){
              mejora=0;
     if(flat==1){
       Guardar_Solucion_Inicial(TUBERIAS,BOMBAS,VALVULAS,NODOS,
                               TUBERIAS_Inicial,BOMBAS_Inicial,
                               VALVULAS_Inicial,NODOS_Inicial);
              ActivarBombasValvulas();}
              return(mejora);
                           }}
       if(flat==1){
              Guardar_Solucion_Inicial(TUBERIAS,BOMBAS,VALVULAS,NODOS,TUBERIAS_Inicia
       l,
              BOMBAS_Inicial, VALVULAS_Inicial,NODOS_Inicial);
              ActivarBombasValvulas();}
       return(mejora);
}

/*******************************************************************************/
/***************** Cruzamiento de valvulas y/o bombas **************************/
/*******************************************************************************/
void cruzamiento(int flag,int veces){
  int   i,cruzar1,cruzar2;
  float *a,*b,aux;
       for(i=0;i<NTUBERIAS*3;i++)
              Solucion_Cruzada1[i]=Solucion1[i];

       for(i=0;i<NTUBERIAS*3;i++)
              Solucion_Cruzada2[i]= Solucion2[i];
       if (flag==0){ veces=veces*NTUBERIAS/100;

       for(i=0;i<veces;i++){
              cruzar1=numeros_aleatoriosLsLi(NTUBERIAS-1,0);
              cruzar2=numeros_aleatoriosLsLi(NTUBERIAS-1,0);
              Swap(&Solucion_Cruzada1[cruzar1],&Solucion_Cruzada2[cruzar2]);
              }}

else if (flag==1){

veces=veces*NTotalBom/100;

       for(i=0;i<veces;i++){
       cruzar1=numeros_aleatoriosLsLi(Solucion_Cruzada1[NTUBERIAS*3-1]+NTUBERIAS-
1,NTUBERIAS);
```

```
        cruzar2=numeros_aleatoriosLsLi(Solucion_Cruzada2[NTUBERIAS*3-1]+NTUBERIAS-
1,NTUBERIAS);

        if(cruzar1%2!=0)cruzar1--;
        if(cruzar2%2!=0)cruzar2--;
                Swap(&Solucion_Cruzada1[cruzar1],&Solucion_Cruzada2[cruzar2]);
        }}

        else          {
                veces=veces*NTotalVal/100;
        for(i=0;i<veces;i++){
        cruzar1=numeros_aleatoriosLsLi
        ((Solucion_Cruzada1[NTUBERIAS*3-2]+
          Solucion_Cruzada1[NTUBERIAS*3-1]+NTUBERIAS-1),NTUBERIAS+
          Solucion_Cruzada1[NTUBERIAS*3-1]);

        cruzar2=numeros_aleatoriosLsLi
        ((Solucion_Cruzada2[NTUBERIAS*3-2]+
          Solucion_Cruzada2[NTUBERIAS*3-1]+NTUBERIAS-1),NTUBERIAS+
          Solucion_Cruzada2[NTUBERIAS*3-1]);

            if(cruzar1%2!=0)cruzar1--;
            if(cruzar2%2!=0)cruzar2--;
                Swap(&Solucion_Cruzada1[cruzar1],&Solucion_Cruzada2[cruzar2]);
        }}}
/*****************************************************************************/
/*************************SELECCIONADOR DE POR TORNEO*************************/
/*****************************************************************************/
int Torneo(int *Eleccion,int *iterada,int *tabu){
int     elejirHeuristica,elejirHeuristica1,
        aux,cont=0,j=0,i;
                elejirHeuristica =numeros_aleatoriosLsLi(TotalHeuris/2,1);
                elejirHeuristica1=numeros_aleatoriosLsLi(TotalHeuris,TotalHeuris/2+1);
                aux              =numeros_aleatoriosLsLi(1,0);
        if(aux==1)
                *Eleccion        =HeuristicRun[elejirHeuristica*2-1];
        else
                *Eleccion        =HeuristicRun[elejirHeuristica1*2-1];
return(elejirHeuristica*2-1);
}
/*****************************************************************************/
/************************SELECCIONADOR DE POR RULETA**************************/
/*****************************************************************************/
int ruleta(int *Eleccion,double costo,int *iterada,int *tabu){
  int   elejirHeuristica,
        i,aux,aux1,
        posición,
  double w=TotalHeuris*2,suma=costo;
*iterada=0,*tabu=0;

do{*Eleccion=0;
     elejirHeuristica=numeros_aleatoriosLsLi((int)costo,1);

     for(i=0;i<TotalHeuris*2;i+=2){
          suma=suma-w;
          w-=2;

        if(elejirHeuristica>=(int)suma){
          *Eleccion=HeuristicRun[i+1];
          if(*iterada==0&&*tabu==0)aux=i+1;
          break;

        }     }
                  if(*Eleccion==1000){*iterada=1;
                                      aux=TotalHeuris*2-1;      }

                  if(*Eleccion==100){*tabu=1;
```

```
  }while(*Eleccion==1000||*Eleccion==100);
         return(aux);}
/*****************************************************************************/
/*********************SELECCIONADOR DE POR RULETA*****************************/
/*****************************************************************************/

int Elite(int *Eleccion,int *iterada,int *tabu){
int    elejirHeuristica,
       cont=0,j=0,i,max;
         *Eleccion=HeuristicRun[1];  *tabu=0;

                if(*Eleccion==100){*tabu=1;
                                  *Eleccion=HeuristicRun[3];
                                   return(3);}
                else{
                     return(1);}
  }

/*****************************************************************************/
/************************** Estructura de Vecindad ***************************/
/*****************************************************************************/
int Checar_ListTABU(int *vector,int indice,float Tuberia,float Valor, int flag){
int i;
for(i=0;i<TABU*4;i+=4)
       if (vector[i]==(float)indice&&vector[i+1]==
       float)Tuberia&&vector[i+2]==Valor&&vector[i+3]==(float)flag)
                     return 1;
       }

/*****************************************************************************/
/***************************** HIPERHEURISTICA *******************************/
/*****************************************************************************/
void EleccionVecindad(int *Eleccion,int flag) {
   int i;

            if(flag==1)*Eleccion=1;//Reducir    Diametro
       else if(flag==2)*Eleccion=2;//CambioAlea Diametro
       else if(flag==3)*Eleccion=4;//Cambiar    Bomba
       else if(flag==4)*Eleccion=5;//Quitar     Bomba
       else if(flag==5)*Eleccion=7;//Cambiar    Valvulas
       else if(flag==6)*Eleccion=8;//Quitar     Valvulas

else if(flag==7) *Eleccion=numeros_aleatoriosLsLi   (6,5);
else if(flag==8) *Eleccion=CombinacionesVecindad    (4,5,0,0,0,0,0,0,2);
else if(flag==9) *Eleccion=CombinacionesVecindad    (4,6,0,0,0,0,0,0,2);
else if(flag==10)*Eleccion=numeros_aleatoriosLsLi   (6,4);
else if(flag==11)*Eleccion=CombinacionesVecindad    (5,1,0,0,0,0,0,0,2);
else if(flag==12)*Eleccion=CombinacionesVecindad    (5,2,0,0,0,0,0,0,2);
else if(flag==13)*Eleccion=CombinacionesVecindad    (5,3,0,0,0,0,0,0,2);
else if(flag==14)*Eleccion=CombinacionesVecindad    (6,1,0,0,0,0,0,0,2);
else if(flag==15)*Eleccion=CombinacionesVecindad    (6,2,0,0,0,0,0,0,2);
else if(flag==16)*Eleccion=numeros_aleatoriosLsLi   (4,3);
else if(flag==17)*Eleccion=CombinacionesVecindad    (4,1,0,0,0,0,0,0,2);
else if(flag==18)*Eleccion=CombinacionesVecindad    (4,2,0,0,0,0,0,0,2);
else if(flag==19)*Eleccion=CombinacionesVecindad    (4,5,6,1,2,0,0,0,5);
else if(flag==20)*Eleccion=CombinacionesVecindad    (5,8,6,9,0,0,0,0,4);
else if(flag==21)*Eleccion=CombinacionesVecindad    (5,8,4,7,0,0,0,0,4);
else if(flag==22)*Eleccion=CombinacionesVecindad    (6,9,4,7,0,0,0,0,4);
else if(flag==23)*Eleccion=numeros_aleatoriosLsLi   (9,4);
else if(flag==24)*Eleccion=numeros_aleatoriosLsLi   (9,8);
else if(flag==25)*Eleccion=numeros_aleatoriosLsLi   (8,7);
else if(flag==26)*Eleccion=CombinacionesVecindad    (9,7,0,0,0,0,0,0,2);
else if(flag==27)*Eleccion=numeros_aleatoriosLsLi   (9,7);
else if(flag==28)*Eleccion=CombinacionesVecindad    (8,1,0,0,0,0,0,0,2);
else if(flag==29)*Eleccion=CombinacionesVecindad    (8,2,0,0,0,0,0,0,2);
else if(flag==30)*Eleccion=CombinacionesVecindad    (8,3,0,0,0,0,0,0,2);
else if(flag==31)*Eleccion=CombinacionesVecindad    (9,1,0,0,0,0,0,0,2);
else if(flag==32)*Eleccion=CombinacionesVecindad    (9,2,0,0,0,0,0,0,2);
```

```
else if(flag==33)*Eleccion=CombinacionesVecindad    (7,3,0,0,0,0,0,0,2);
else if(flag==34)*Eleccion=CombinacionesVecindad    (7,1,0,0,0,0,0,0,2);
else if(flag==35)*Eleccion=CombinacionesVecindad    (7,2,0,0,0,0,0,0,2);
else if(flag==36)*Eleccion=CombinacionesVecindad    (1,3,7,8,9,0,0,0,5);
else if(flag==37)*Eleccion=CombinacionesVecindad    (4,7,5,8,3,0,0,0,5);
else if(flag==38)*Eleccion=CombinacionesVecindad    (4,7,5,8,1,0,0,0,5);
else if(flag==39)*Eleccion=CombinacionesVecindad    (4,7,5,8,2,0,0,0,5);
else if(flag==40)*Eleccion=CombinacionesVecindad    (4,7,6,9,3,0,0,0,5);
else if(flag==41)*Eleccion=CombinacionesVecindad    (4,7,6,9,1,0,0,0,5);
else if(flag==42)*Eleccion=CombinacionesVecindad    (4,7,6,9,2,0,0,0,5);
else if(flag==43)*Eleccion=CombinacionesVecindad    (5,8,6,9,3,0,0,0,5);
else if(flag==44)*Eleccion=CombinacionesVecindad    (5,8,6,9,1,0,0,0,5);
else if(flag==45)*Eleccion=CombinacionesVecindad    (5,8,6,9,2,0,0,0,5);
else if(flag==46)*Eleccion=CombinacionesVecindad    (5,8,1,0,0,0,0,0,3);
else if(flag==47)*Eleccion=CombinacionesVecindad    (5,8,2,0,0,0,0,0,3);
else if(flag==48)*Eleccion=CombinacionesVecindad    (5,8,3,0,0,0,0,0,3);
else if(flag==49)*Eleccion=CombinacionesVecindad    (6,9,1,0,0,0,0,0,3);
else if(flag==50)*Eleccion=CombinacionesVecindad    (6,9,2,0,0,0,0,0,3);
else if(flag==51)*Eleccion=CombinacionesVecindad    (4,7,3,0,0,0,0,0,3);
else if(flag==52)*Eleccion=CombinacionesVecindad    (4,7,1,0,0,0,0,0,3);
else if(flag==53)*Eleccion=CombinacionesVecindad    (4,7,2,0,0,0,0,0,3);
else if(flag==54)*Eleccion=CombinacionesVecindad    (1,3,4,5,6,7,8,9,8);
 }
/*****************************************************************************************/
/*****************************BUSQUEDA LOCAL ***********************************************/
/*****************************************************************************************/
int CombinacionesVecindad(int a1,int a2,int a3,int a4,int a5,int a6,int a7,int a8,int
max){
int     vector[8];int A;
        vector[0]=a1;
        vector[1]=a2;
        vector[2]=a3;
        vector[3]=a4;
        vector[4]=a5;
        vector[5]=a6;
        vector[6]=a7;
        vector[7]=a8;
        A=numeros_aleatoriosLsLi(max-1,0);
return (vector[A]);
}
/*****************************************************************************************/
/********************ELECCION CUANDO NO HAY VALVULAS O BOMBAS ***********************/
/*****************************************************************************************/
int EleccionValBom(int a1,int a2,int a3,int a4,int a5,int a6, int flag){
int A;
if (flag==1)A=numeros_aleatoriosLsLi(3,1);
else A=numeros_aleatoriosLsLi(2,1);

    if(A==1)A=numeros_aleatoriosLsLi(a2,a1);
      else if (A==2)A=numeros_aleatoriosLsLi(a4,a3);
        else A=numeros_aleatoriosLsLi(a6,a5);

return(A);
}
/*****************************************************************************************/
/****************************** Estructura de Vecindad ***********************************/
/*****************************************************************************************/
int Estructura_Vecindad(int *i,int Eleccion,int tabu,int *flagtabu,int *bandera44){
int j;
for(j=0;j<cambios;j++){
        switch(Eleccion){
        case 1:{
                CambioDiametro(tuberiaPRESPOSDia_Vdestino, NTUBERIAS,1,1,tabu);
                bandera44[j]=1;
        break;
        }
        case 2:{
                CambioDiametro(tuberiaPRESPOSDia_Vdestino, NTUBERIAS,2,1,tabu);
```

```
            bandera44[j]=1;
        break;
        }
        case 3:{
            CambioDiametro(tuberiaPRESPOSDia_Vdestino, NTUBERIAS,3,1,tabu);
            bandera44[j]=1;
        break;
        }
        case 4:{
        if (NTotalBom>0)*flagtabu=Vecindad_Bombas(1,NTotalBom,Bombas_Red,0,tabu);
            bandera44[j]=2;
        break;
        }
        case 5:{
        if (NTotalBom>0)*flagtabu=Vecindad_Bombas(2,NTotalBom,Bombas_Red,0,tabu);
            bandera44[j]=2;
        break;
        }
        case 6:{
            *flagtabu=Vecindad_Bombas(3,NTUBERIAS,Bombas_Red,1,tabu);
            bandera44[j]=3;
        break;
        }
        case 7:{
        if (NTotalVal>0)*flagtabu=Vecindad_Valvulas(1,NTotalVal,Valvulas_Red,0,tabu);
            bandera44[j]=4;
        break;
        }
        case 8:{
        if (NTotalVal>0)*flagtabu=Vecindad_Valvulas(2,NTotalVal,Valvulas_Red,0,tabu);
                            bandera44[j]=4;
        break;
        }
        case 9:{
        if (NTotalVal>0)*flagtabu=Vecindad_Valvulas(3,NVALVULAS,Valvulas_Red,1,tabu);
             bandera44[j]=5;
        break;
        }
        case 10:{
        if
(NValPos>0)*flagtabu=Vecindad_Valvulas(3,NTubNegPosVal,TuberiaNegPosValulas,0,tabu);
             bandera44[j]=5;
        break;
        }
        case 11:{
             CambioDiametro(TuberiaNegPosDiametro, NTubNegPosDia,1,0,tabu);
                            bandera44[j]=1;
        break;
        }
        case 12:{
        if (NValPos>0)*flagtabu=Vecindad_Valvulas(3,NValPos,tuberiaPRESPOS,0,tabu);
             bandera44[j]=5;
        break;
        }
        case 13:{
             bandera44[j]=1;
        break;
          }
}}}
/*****************************************************************************/
/***************************** DISTANCIA ENTRE LOS NODOS *********************/
/*****************************************************************************/
void Agregar_Tuberias(){
int i;
        for(i=0;i<NTUBERIAS;i++){
        if(i==0){
            TUBERIAS[i].Vorigen=EMBALSES[i].id_dep;
            TUBERIAS[i].Vdestino=i+1;
```

```
        }
        else {
            TUBERIAS[i].Vorigen=i;
            TUBERIAS[i].Vdestino=i+1;
        }
            TUBERIAS[i].NTuberia=i+1;
            TUBERIAS[i].diametro=DIACOMER[numeros_aleatoriosLsLi(NDIAMCOMERS-
1,0)].Diam;
        }
            LongitudTuberias();
}
/*****************************************************************************/
/************************** CALCULO DE LONGITUD ******************************/
/*****************************************************************************/

void LongitudTuberias(){
int i;
for(i=0;i<NTUBERIAS;i++){
  if (i==0)
    TUBERIAS[i].longitud=pow((pow((EMBALSES[0].altura-NODOS[0].cota),2)+
                             pow((EMBALSES[0].coordX-NODOS[0].coordX),2)+
                             pow((EMBALSES[0].coordY-NODOS[0].coordY),2)),0.5);
  else
    TUBERIAS[i].longitud=pow((pow((NODOS[i].cota-NODOS[i-1].cota),2)+
                             pow((NODOS[i].coordX-NODOS[i-1].coordX),2)+
                             pow((NODOS[i].coordY-NODOS[i-1].coordY),2)),0.5);
} }
/*****************************************************************************/
/****************************** CARGAR ARCHIVOS ******************************/
/*****************************************************************************/

void Cargar_Archivos(){
int i;
Cargar_datos("archivos/costos/Bomba.txt",6);
Cargar_datos("archivos/costos/DiamComer.txt",8);
Cargar_datos("archivos/costos/valvulasCosto.txt",9);
Cargar_datos("archivos/Coordenadas.txt",10);
}
/*****************************************************************************/
/*****************************CARGAR DATOS DE LA INSTANCIA ********************/
/*****************************************************************************/
void Cargar_datos(char *Arch,int bandera){
 int i; int j;
    FILE *fp;
    if((fp=fopen(Arch,"rt"))==NULL){
           exit(0);
       }

       if (bandera==1){   for(i=0;!feof(fp);i++){
       }}
       if (bandera==2){   for(i=0;!feof(fp);i++){

fscanf(fp,"%d%f%f%f",&EMBALSES[i].id_dep,&EMBALSES[i].altura,&EMBALSES[i].coordX,&EMBA
LSES[i].coordY);
       }}
       if (bandera==3){   for(i=0;!feof(fp);i++){

fscanf(fp,"%d%f%f%f%f%f%f%f",&DEPOSITOS[i].id_dep,&DEPOSITOS[i].altura,&DEPOSITOS[i].N
ivelIni,&DEPOSITOS[i].NivelMax,

&DEPOSITOS[i].NivelMin,&DEPOSITOS[i].diametro,&DEPOSITOS[i].coordX,&DEPOSITOS[i].coord
Y);
       }}
       if (bandera==4){   for(i=0;!feof(fp);i++){

fscanf(fp,"%d%d%d%f%f",&TUBERIAS[i].NTuberia,&TUBERIAS[i].Vorigen,&TUBERIAS[i].Vdestin
o,&TUBERIAS[i].longitud,
                    &TUBERIAS[i].diametro);
```

```
        }}
        if (bandera==5){   for(i=0;!feof(fp);i++){
            fscanf(fp,"%f %f",&DiamComerCosto[i][0],&DiamComerCosto[i][1]);
        }}
        if (bandera==6){   for(i=0;!feof(fp);i++){

fscanf(fp,"%d%f%f",&BombasComerID[i],&BombasComerCosto[i],&BombasComerPotencia[i]);
        }}
        if (bandera==7){   for(i=0;!feof(fp);i++){
            fscanf(fp,"%d%d%d%f",&BOMBAS[i].id_bom,&BOMBAS[i].nodoAsp,
                       &BOMBAS[i].nodoImp,&BOMBAS[i].potencia);
              BOMBAS[i].ExisteBomba=0;
        }}
        if (bandera==8){   for(i=0;!feof(fp);i++){

fscanf(fp,"%d%f%f",&DIACOMER[i].id_Diam,&DIACOMER[i].CostoDiam,&DIACOMER[i].Diam);
        }}
        if (bandera==9){   for(i=0;!feof(fp);i++){
            fscanf(fp,"%d%f",&ValvulasComerID[i],&ValvulasComerCosto[i]);
        }}
        if (bandera==10){
          for(i=0;!feof(fp);i++){
            if(i==0){
                    EMBALSES[i].id_dep=2000;

fscanf(fp,"%f%f%f",&EMBALSES[i].coordX,&EMBALSES[i].coordY,&EMBALSES[i].altura);
            }

            else {  NODOS[i-1].idnodo=i;
                    NODOS[i-1].demanda=0.0;
                    fscanf(fp,"%f%f%f",&NODOS[i-1].coordX,&NODOS[i-1].coordY,&NODOS[i-
1].cota);
          }       }
                    NODOS[i-2].demanda=DemandaDeposito;
        }

        if (bandera==11){   for(i=0;!feof(fp);i++){

fscanf(fp,"%d%d%d%f%d%d",&VALVULAS[i].id_val,&VALVULAS[i].H2Oarriba,&VALVULAS[i].H2Oab
ajo,

&VALVULAS[i].consigna,&VALVULAS[i].ExisteValvula,&VALVULAS[i].Ntuberia);
            VALVULAS[i].ExisteValvula=0;
        }}

 fclose(fp);
}
/*****************************************************************************/
/********************** AGREGAR VALVULAS Y BOMBAS A LAS ESTRUCTURAS ***************/
/*****************************************************************************/
void Agregar_Bombas(){
   int i;NBOMBAS=0;

   for(i=0;i<NTUBERIAS;i++){
        BOMBAS[i].id_bom=TUBERIAS[i].NTuberia+8000;
        BOMBAS[i].nodoAsp=TUBERIAS[i].Vorigen;
        BOMBAS[i].nodoImp=TUBERIAS[i].Vdestino;
        BOMBAS[i].potencia=0.01;
        BOMBAS[i].ExisteBomba=0;NBOMBAS++;

   }}

 void Agregar_Valvulas(){
   int i,j,bandera=0,valve;NVALVULAS=0;

   for(i=0;i<NTUBERIAS;i++){bandera=0;
   for(j=0;j<NEMBS;j++)if((TUBERIAS[i].Vorigen==EMBALSES[j].id_dep)||
```

```
(TUBERIAS[i].Vdestino==EMBALSES[j].id_dep)){bandera=1;}

for(j=0;j<i;j++)if((TUBERIAS[i].Vorigen==VALVULAS[j].H2Oarriba)||

(TUBERIAS[i].Vdestino==VALVULAS[j].H2Oabajo)||(TUBERIAS[i].Vdestino==VALVULAS[j].H2Oar
riba)||

(TUBERIAS[i].Vorigen==VALVULAS[j].H2Oabajo))bandera=1;

 if(bandera==0){
       VALVULAS[NVALVULAS].id_val=NVALVULAS+5001;;
       VALVULAS[NVALVULAS].H2Oarriba=TUBERIAS[i].Vorigen;
       VALVULAS[NVALVULAS].H2Oabajo=TUBERIAS[i].Vdestino;
       VALVULAS[NVALVULAS].estado=0;
       VALVULAS[NVALVULAS].perdidas=0;
       VALVULAS[NVALVULAS].consigna=0;
       VALVULAS[NVALVULAS].ExisteValvula=0;
       VALVULAS[NVALVULAS].Ntuberia=TUBERIAS[i].NTuberia;
       NVALVULAS++;
       }}}
/*****************************************************************************/
/****************** DESACTIVAR VALULAS Y BOMBAS Y ACTIVAR LAS TUBERIAS ***************/
/*****************************************************************************/
void DesactivarBombasValvulas(){
  int i, indice;
  for(i=0;i<NTUBERIAS;i++){
           indice=i+1+NTUBERIAS;
           ENsetlinkvalue(indice,EN_INITSTATUS,0);
           ENsetlinkvalue(i+1,EN_INITSTATUS,1);
           ENsetlinkvalue(i+1,EN_DIAMETER,TUBERIAS[i].diametro);
           BOMBAS[i].potencia=0.01;
           BOMBAS[i].ExisteBomba=0;
           if(i<NVALVULAS){
             VALVULAS[i].ExisteValvula=0;
             VALVULAS[i].consigna=0;
             indice=NTUBERIAS+NBOMBAS+1+i;
             ENsetlinkvalue(indice,EN_INITSTATUS,0);
  }          }}
/*****************************************************************************/
/************** ACTIVAR VALVULAS Y BOMBAS Y DESACTIVAR LAS TUBERIAS ***************/
/*****************************************************************************/
void ActivarBombasValvulas(){
int i, indice;

for(i=0;i<NTUBERIAS;i++){
           ENsetlinkvalue(i+1,EN_DIAMETER,TUBERIAS[i].diametro); indice=i+1+NTUBERIAS;
  if(BOMBAS[i].ExisteBomba==1){
           ENsetlinkvalue(i+1,EN_INITSTATUS,0);
           ENsetlinkvalue(indice,EN_INITSETTING,BOMBAS[i].potencia);

  }
  else{
           ENsetlinkvalue(indice,EN_INITSTATUS,0);
           ENsetlinkvalue(i+1,EN_INITSTATUS,1);

}}

for(i=0;i<NVALVULAS;i++){
           indice=NTUBERIAS+NBOMBAS+1+i;
  if(VALVULAS[i].ExisteValvula==1){
           ENsetlinkvalue(VALVULAS[i].Ntuberia,EN_INITSTATUS,0);
           ENsetlinkvalue(indice,EN_INITSETTING,VALVULAS[i].consigna);
  }
  else{
           ENsetlinkvalue(VALVULAS[i].Ntuberia,EN_INITSTATUS,1);
           ENsetlinkvalue(indice,EN_INITSETTING,0);
           ENsetlinkvalue(indice,EN_INITSTATUS,0);
```

```
  }}}
/*****************************************************************************/
/******************************** CREAR ARCHIVO VALVULAS *********************/
/*****************************************************************************/
void Activar_Cruzamiento(float *vector){
int i;
DesactivarBombasValvulas();
for(i=0;i<NTUBERIAS;i++)
          ENsetlinkvalue(i+1,EN_DIAMETER,vector[i]);
for(i=NTUBERIAS;i<NTUBERIAS+(int)vector[NTUBERIAS*3-1]+vector[NTUBERIAS*3-2];i+=2){
          ENsetlinkvalue((int)vector[i+1],EN_INITSTATUS,0);
          ENsetlinkvalue((int)vector[i+1]+NTUBERIAS,EN_INITSTATUS,1);
          ENsetlinkvalue((int)vector[i+1]+NTUBERIAS,EN_INITSETTING,vector[i]); } }

/*****************************************************************************/
/******************************** CREAR ARCHIVO VALVULAS *********************/
/*****************************************************************************/
void insertar_val(){
  int i,j,k;NVALVULAS=0;
for(i=0;i<NTUBERIAS;i++){
  a=1;
  if(a==1){
       for(j=0;j<NVALVULAS;j++){

if((TUBERIAS[i].Vorigen==VALVULAS[j].H2Oarriba)||(TUBERIAS[i].Vdestino==VALVULAS[j].H2
Oabajo)||(
TUBERIAS[i].Vdestino==VALVULAS[j].H2Oarriba||TUBERIAS[i].Vorigen==VALVULAS[j].H2Oabajo
))
           {a=0;break

       for(j=0;j<NEMBS;j++)

if((TUBERIAS[i].Vorigen==EMBALSES[j].id_dep)||(TUBERIAS[i].Vdestino==EMBALSES[j].id_de
p))
         {a=0; break;}         }

if(a==1){

   VALVULAS[NVALVULAS].ExisteValvula;
   VALVULAS[NVALVULAS].id_val=NVALVULAS+8001;
   VALVULAS[NVALVULAS].H2Oarriba=TUBERIAS[i].Vorigen;
   VALVULAS[NVALVULAS].H2Oabajo=TUBERIAS[i].Vdestino;
   VALVULAS[NVALVULAS].estado=0;
   VALVULAS[NVALVULAS].perdidas=0;
   VALVULAS[NVALVULAS].consigna=0;
   VALVULAS[NVALVULAS].Ntuberia=TUBERIAS[i].NTuberia;
NVALVULAS++;
}      }      }

/*****************************************************************************/
/****************** CREA ARCHIVOS DE VALVULAS ********************************/
/*****************************************************************************/

void Crear_archivo_VALVULAS(){
      int i;
      char TIPO[]="PRV";
    FILE *fichero= fopen("xnom","a");
    if (!fichero) {
        printf("\nERROR AL LEER FICHERO TXT");
    }
    else {
   for(i=0;i<NVALVULAS;i++)
         fprintf(fichero,"%d\t%d\t%d\t%0.2f\t%s\t%.2f\t%d\t;\n",

VALVULAS[i].id_val,VALVULAS[i].H2Oarriba,VALVULAS[i].H2Oabajo,TUBERIAS[VALVULAS[i].Ntu
beria-1].diametro,TIPO,VALVULAS[i].consigna,VALVULAS[i].perdidas);
    }
    fclose(fichero);
```

```
}
void Crear_archivo_VALVULAS2(){
      int i;
      char TIPO[]="PRV";
    FILE *fichero= fopen("ValvulasActivas","a");
    if (!fichero) {
        printf("\nERROR AL LEER FICHERO TXT");
    }
    else {
    for(i=0;i<NVALVULAS;i++)
         fprintf(fichero,"%d\t%d\t%d\t%0.2f\t0\t0\t%d\n",

VALVULAS[i].id_val,VALVULAS[i].H2Oarriba,VALVULAS[i].H2Oabajo,TUBERIAS[VALVULAS[i].Ntu
beria-1].diametro,VALVULAS[i].Ntuberia);
    }
    fclose(fichero);
}
/*****************************************************************************/
/************************ LEE CONFIGURACIÓN CARGA A EPANET ************************/
/*****************************************************************************/
void LeerConfiguracion_EPANET(int sol){
 int i, nnodes,nlinks,npumps;
 int cont=0;
FILE *f;
 char ArchINP[60],ArchRPT[60];
        sprintf(ArchINP, "archivos/instancia.inp");
       if ((f = fopen(ArchRPT,"wt")) == NULL);
              if (ENopen(ArchINP,"soluciones/mensaje.rpt","")>0){
           printf("\n1.-Error No se puedo abrir el archivo INP");
           ENclose();
           }

       else{
           ENopenH();
           ENgetcount(EN_NODECOUNT, &nnodes);
           ENgetcount(EN_LINKCOUNT, &nlinks);
           ENgetcount(EN_PUMPCOUNT,&npumps);           }
}
/*****************************************************************************/
/********************** Evaluar factibilidad de la solución ************************/
/*****************************************************************************/
int Evaluar_factibilidad(){
int    sol,Factible=0,nverts=0,
       ntubs=0,i,presOld,
       bandera=0,
       ContadorInfactible=0,
       contador=0,
       k=0;int cont=0;int Aux0=0,presOldN;
                    Factible=RunEPANET(); Actualizar_Val_bom();
        for(i=0;(Factible!=1 && i<1000);i++){
                           presOld=presPos;presOldN=presNeg;
             if(i<2&&presNeg>0){
                           a=3;swap(a);}
                    else if(Factible==2||Factible==0){
                           a=numeros_aleatoriosLsLi(3,1);
                           swap(a);}
                    else if(Factible==3&&contador==0 {
                           a=4;
                           contador=0;swap(a);}
                           Factible=RunEPANET();
                           Actualizar_Val_bom();
             if(presOld==presPos&&presOldN==presNeg)cont++; else cont=0;
                    if(cont==50){return 0;}
       }Factible=RunEPANET();
if(Factible==1){
return 1;
}
else return(0);
```

```
}

/*****************************************************************************/
/****************** Actualizar la exitencia de bombas y valulas*****************/
/*****************************************************************************/
void Actualizar_Val_bom(){NTotalVal=0;NTotalBom=0;int i;
for(i=0;i<NVALVULAS;i++)
  if(VALVULAS[i].ExisteValvula==1){Valvulas_Red[NTotalVal]=VALVULAS[i].Ntuberia;
                        NTotalVal++;
                                  }
for(i=0;i<NBOMBAS;i++){Diametros_Red[i]=TUBERIAS[i].diametro;
  if(BOMBAS[i].ExisteBomba==1){Bombas_Red[NTotalBom]=TUBERIAS[i].NTuberia;

                                NTotalBom++;
                                }}}
/*****************************************************************************/
/********************* BAJAR ESTRUCTURA DE VECINDAD BOMBA ************************/
/*****************************************************************************/
void BajarPresion1(){
  int Eleccion,AnteriorNeg,AnteriorPos,i=1,Factible=0,flag;
  int flagtabu, cambios=1, bandera44[3];

   Eleccion=numeros_aleatoriosLsLi(9,1);

        Estructura_Vecindad(&i,Eleccion,0,&flagtabu,bandera44);
                                                AnteriorNeg=presNeg;
                                                AnteriorPos=presPos;

Factible=RunEPANET();Actualizar_Val_bom();
                                    if(presNeg>0||presPos>AnteriorPos){

Solucion_Anterior(bandera44);Factible=RunEPANET();Actualizar_Val_bom();
}             }

/*****************************************************************************/
/********************* BAJAR ESTRUCTURA DE VECINDAD BOMBA ************************/
/*****************************************************************************/
void BajarPresion(){
float demand, pressure,c,AnteriorNeg=presNeg,AnteriorPos=presPos;
int     Aleatorio=3,b=0,i,x,z,y;int f=0,
        Factible,consigna;int cont,indice; int bandera=0,
        cambios=1,
        tuberia=TUBERIAS[a].NTuberia;
        a=numeros_aleatoriosLsLi(NTUBERIAS-1,0);

for(x=0;x<NVALVULAS;x++){
        if(tuberia==VALVULAS[x].Ntuberia)
                break;}
        indice=tuberiaPRESNEG[a]+NTUBERIAS;

        if(VALVULAS[x].ExisteValvula==1&&x<NVALVULAS){
                indice=x+NTUBERIAS+NBOMBAS+1;
                c=VALVULAS[x].consigna;
                VALVULAS[x].consigna=numeros_aleatorios();
                b=numeros_aleatoriosLsLi(1,0);
        if(b==0){
                funcion(tuberia, 0, indice, VALVULAS[x].consigna,1);bandera=1;
                 }
        else {funcion(tuberia, 1, indice, VALVULAS[x].consigna,0);bandera=1;
                                        VALVULAS[x].ExisteValvula=0;
                                                }}
        else
if(VALVULAS[x].ExisteValvula==0&&x<NVALVULAS&&BOMBAS[tuberia].ExisteBomba==0){
                indice=x+NTUBERIAS+NBOMBAS+1;
                VALVULAS[x].ExisteValvula=1;
                VALVULAS[x].consigna=numeros_aleatorios();
                c=VALVULAS[x].consigna;
            funcion(tuberia, 0, indice, VALVULAS[x].consigna,1);bandera=2;
```

```
                    }
       else if(BOMBAS[a].ExisteBomba==1){int bandera2=numeros_aleatoriosLsLi(1,0);
      if(bandera2==1) { indice=tuberia+NTUBERIAS;
              c=BOMBAS[tuberia-1].potencia;
BOMBAS[tuberia-1].potencia=BombasComerPotencia[numeros_aleatoriosLsLi(NBOMBASComer-
1,0)];
funcion(tuberia, 0, indice,  BOMBAS[tuberia-1].potencia,1);bandera=3;
                                    }
                    else {
              indice=tuberia+NTUBERIAS;
              BOMBAS[tuberia-1].ExisteBomba=0;
              c=BOMBAS[tuberia-1].potencia;
funcion(tuberia, 1, indice,  BOMBAS[tuberia-1].potencia,0);bandera=4;
                                 }          }
                    else {

CambioDiametro(tuberiaPRESPOSDia_Vdestino,NTUBERIAS,1,1,0);
                               bandera=5;
                    }
                        AnteriorNeg=presNeg;
                        AnteriorPos=presPos;
                        Factible=RunEPANET();
if(bandera==1&&(AnteriorNeg<presNeg||AnteriorPos<presPos)){
           VALVULAS[x].ExisteValvula=1;
           VALVULAS[x].consigna=c;
    funcion(tuberia, 0, indice, VALVULAS[x].consigna,1);
                                    }
Else if(bandera=2&&(AnteriorNeg<presNeg||AnteriorPos<presPos))
{VALVULAS[x].ExisteValvula=0;
    funcion(tuberia, 1, indice,  0,0);
                                    }
else if(bandera==3&&(AnteriorNeg<presNeg||AnteriorPos<presPos)){BOMBAS[tuberia-
1].potencia=c;
    funcion(tuberia, 0, indice,  BOMBAS[tuberia-1].potencia,1);
                                    }
else if(bandera==4&&(AnteriorNeg<presNeg||AnteriorPos<presPos)){
                                 BOMBAS[tuberia-1].potencia=c;
                                 BOMBAS[tuberia-1].ExisteBomba=1;
    funcion(tuberia, 0, indice,  BOMBAS[tuberia-1].potencia,1);
                                    }
else if(bandera==5&&(AnteriorNeg<=presNeg||AnteriorPos<=presPos)){
    funcion(CambiosTABU_Diametros[1], 0, CambiosTABU_Diametros[i],
CambiosTABU_Diametros[2],7);
       CambiosTABU_Diametros[3]=bandera;
       TUBERIAS[(int)CambiosTABU_Diametros[1]-1].diametro=CambiosTABU_Diametros[2];
                                    }Factible=RunEPANET();}

/*****************************************************************************/
/************************ ESTRUCTURA DE VECINDAD BOMBA ***************************/
/*****************************************************************************/
int  Vecindad_Bombas( int Eleccion,int Nbombas,int *vector,int flag,int tabu){
int i,indice,z,ciclos=0,flagtabu=0,cont=0,Cancela_tabu=0;
             do{
             a=numeros_aleatoriosLsLi(Nbombas-1,0);
             if(flag==1)vector[a]=a+1;
while (((Eleccion!=3&&BOMBAS[vector[a]-1].ExisteBomba==0)||
                           (Eleccion==3&&BOMBAS[vector[a]-1].ExisteBomba==1)) )
                                    {a=numeros_aleatoriosLsLi(Nbombas-1,0);
                    if(flag==1) vector[a]=a+1;
                    if(tabu==1){
                      if (Eleccion==2){
                         if(ListaTABU(1, vector[a],
CambiosTABU_Bombas,TABU*4)!=0&&CambiosTABU_Bombas[3]==2){
                                    flagtabu=1;  cont++;
                                    if(cont==NTotalBom*5){flagtabu=0;
                                    Cancela_tabu=1;
                                    }    }
                        else {flagtabu=0;
```

```
                        }     }
                      else{
if((ListaTABU(1, vector[a], CambiosTABU_Bombas,TABU*4)!=0)&&(ListaTABU(2,
BOMBAS[vector[a]-1].potencia,CambiosTABU_Bombas,TABU*4)!=0)&&
        CambiosTABU_Bombas[3]!=2){flagtabu=1;
            }
                      else {flagtabu=0;
                              }}}
             }while(flagtabu==1);
                                          indice=vector[a]+NTUBERIAS;
        for(z=4*TABU-1;z>4;z--){
               CambiosTABU_Bombas[z]=CambiosTABU_Bombas[z-4];}
               CambiosTABU_Bombas[0]=(float)indice;
               CambiosTABU_Bombas[1]=(float)vector[a];
               CambiosTABU_Bombas[2]=BOMBAS[vector[a]-1].potencia;
               CambiosTABU_Bombas[3]=(float)Eleccion;
               BOMBAS[vector[a]-1].potencia=BombasComerPotencia[numeros_aleatoriosLsLi
                       (NBOMBASComer-1,0)];
if(Eleccion==2){BOMBAS[vector[a]-1].ExisteBomba=0;
        funcion(vector[a], 0, indice,BOMBAS[vector[a]-1].potencia,0);}
else{BOMBAS[vector[a]-1].ExisteBomba=1;
        funcion(vector[a], 0, indice,BOMBAS[vector[a]-1].potencia,1);}
return(Cancela_tabu);}

/*****************************************************************************/
/************************* ESTRUCTURA DE VECINDAD VALVULAS ************************/
/*****************************************************************************/
int  Vecindad_Valvulas(int Eleccion,int NValvulas,int *vector,int flag,int tabu){
int     i,indice,z,ciclos=0,
        x,int flagtabu=0;
 do{
          a=numeros_aleatoriosLsLi(NValvulas-1,0);
    if(flag==1){vector[a]==VALVULAS[a].Ntuberia;x=a;}
   else for(x=0;x<NVALVULAS;x++){
  if(vector[a]==VALVULAS[x].Ntuberia)break;}
 while ((Eleccion!=3&&VALVULAS[x].ExisteValvula==0)||
(Eleccion==3&&VALVULAS[x].ExisteValvula==1)){a=numeros_aleatoriosLsLi(NValvulas-1,0);
if(flag==1){vector[a]==VALVULAS[a].Ntuberia;x=a;}
        else for(x=0;x<NVALVULAS;x++){
               if(vector[a]==VALVULAS[x].Ntuberia)break
                       if(tabu==1){
                               if (Eleccion==2){
               if((ListaTABU(1, VALVULAS[x].Ntuberia,CambiosTABU_Valvulas,TABU*4)!=0)
                               &&CambiosTABU_Valvulas[3]==2)flagtabu=1;
                          else flagtabu=0;                   }
                        else{
        if((ListaTABU(1, VALVULAS[x].Ntuberia,
CambiosTABU_Valvulas,TABU*4)!=0)&&(ListaTABU(2, VALVULAS[x].consigna,
CambiosTABU_Valvulas,TABU*4)!=0)&&CambiosTABU_Valvulas[3]!=2)flagtabu=1;
        else flagtabu=0;
                             }          }
              }while(flagtabu==1);
                             indice=x+NTUBERIAS+NTUBERIAS+1;
                             VALVULAS[x].consigna=numeros_aleatoriosLsLi(50,10);

                      for(z=4*TABU-1;z>4;z--){
CambiosTABU_Valvulas[z]=CambiosTABU_Valvulas[z-4];}
CambiosTABU_Valvulas[0]=(float)indice;
CambiosTABU_Valvulas[1]=(float)VALVULAS[x].Ntuberia;
CambiosTABU_Valvulas[2]=VALVULAS[x].consigna;
CambiosTABU_Valvulas[3]=(float)Eleccion;

        if(Eleccion==2){VALVULAS[x].ExisteValvula=0;VALVULAS[x].consigna=0;
funcion(VALVULAS[x].Ntuberia, 0, indice,VALVULAS[x].consigna,0);}
        else{VALVULAS[x].ExisteValvula=1;
funcion(VALVULAS[x].Ntuberia, 0, indice,VALVULAS[x].consigna,1);}
                    return(0);
}
```

```
/*****************************************************************************/
/************************* CAMBIAR A UN DIAMETRO MENOR ***************************/
/*****************************************************************************/
int CambioDiametro(int *vector, int Tam_Arreglo, int bandera,int flag,int tabu){int
z,i,flagtabu=0
        do{
                a=numeros_aleatoriosLsLi(Tam_Arreglo-1,0);
                        if(flag==1)vector[a]=a+1;
        if(bandera==1){
                if(TUBERIAS[vector[a]-1].diametro>DIACOMER[0].Diam){
                        for(z=0;z<NDIAMCOMERS;z++){
                if(TUBERIAS[vector[a]-1].diametro==DIACOMER[z].Diam){
                                b=numeros_aleatoriosLsLi(z-1,0);
                                break;}
                        }}
                else {return(1);}
                        }
                else if(bandera==3){
        if(TUBERIAS[vector[a]-1].diametro<DIACOMER[NDIAMCOMERS-1].Diam){
for(z=0;z<NDIAMCOMERS;z++)
        if(TUBERIAS[vector[a]-1].diametro==DIACOMER[z].Diam){
                                b=numeros_aleatoriosLsLi(NDIAMCOMERS-1,z+1);break;}
                                }
                else {return(1);}
                        }
        else if(bandera==2) {
                                b=numeros_aleatoriosLsLi(NDIAMCOMERS-1,0);
                                    }
        if(tabu==1){
        if ((ListaTABU(1, vector[a], CambiosTABU_Bombas,TABU*4)!=0)&&(ListaTABU
            (2,TUBERIAS[vector[a]-1].diametro, CambiosTABU_Bombas,TABU*4)!=0))
               {flagtabu=1;}
        else flagtabu=0;
        }}
                while(flagtabu==1);
                        for(z=4*TABU-1;z>4;z--)       {
CambiosTABU_Diametros[z]=CambiosTABU_Diametros[z-4];}
CambiosTABU_Diametros[0]=(float)a;
CambiosTABU_Diametros[1]=(float)vector[a];
CambiosTABU_Diametros[2]=TUBERIAS[vector[a]-1].diametro;
CambiosTABU_Diametros[3]=(float)bandera;
TUBERIAS[vector[a]-1].diametro= DIACOMER[b].Diam;
ENsetlinkvalue(vector[a],EN_DIAMETER,TUBERIAS[vector[a]-1].diametro);
                      return(0);
}

/*****************************************************************************/
/********************************* QUITAR BOMBAS *****************************/
/*****************************************************************************/
void PonerBomba(){
                        int i,indice,z,ciclos,bandera=0;
                        a=numeros_aleatoriosLsLi(NTUBERIAS-1,0);
                while (BOMBAS[a].ExisteBomba==1)
                {a=numeros_aleatoriosLsLi(NTUBERIAS-1,0)            }

                                              indice=TUBERIAS[a].NTuberia+NTUBERIAS;
                        for(z=4*TABU-1;z>4;z--)       {

        CambiosTABU_Bombas[z]=CambiosTABU_Bombas[z-4];}
                                CambiosTABU_Bombas[0]=(float)indice;
                                CambiosTABU_Bombas[1]=(float)TUBERIAS[a].NTuberia;
                                CambiosTABU_Bombas[2]=BOMBAS[a].potencia;
                                CambiosTABU_Bombas[3]=(float)bandera;


        BOMBAS[a].potencia=BombasComerPotencia[numeros_aleatoriosLsLi(NBOMBASComer-
1,0)];
```

```
                                                BOMBAS[a].ExisteBomba=1;
        funcion(TUBERIAS[a].NTuberia, 0, indice,BOMBAS[a].potencia,1);
        }
/*****************************************************************************/
/****************** BUSCA TUBERIA Q CONECTA AL NODO DESCARGA **********************/
/*****************************************************************************/
int buscar_tuberia_Vdestino(){
int     j,z;
        NDiaPos_Vdestino=0;
tuberiaPRESPOSDia_Vdestino[NTUBERIAS];
for(z=0;z<presPos;z++)
        for(j=0;j<NTUBERIAS;j++){

                        if( nodosPRESPOS[z]==TUBERIAS[j].Vdestino){
        tuberiaPRESPOSDia_Vdestino[NDiaPos_Vdestino]=TUBERIAS[j].NTuberia;
                        NDiaPos_Vdestino++;
                        }}}
/*****************************************************************************/
/*************** FUNCION DE EPANET ACTIVAR/DESACTIVAR ****************************/
/*****************************************************************************/

void funcion(int tuberia, int statusTuberia, int indiceEPANET, float Csgna_Ptencia,int
flag){
  if(flag==7)ENsetlinkvalue(tuberia,EN_DIAMETER,Csgna_Ptencia);
        else{
                ENsetlinkvalue(tuberia,EN_INITSTATUS,statusTuberia);
                ENsetlinkvalue(indiceEPANET,EN_INITSETTING,Csgna_Ptencia);
  if (flag==0)
                ENsetlinkvalue(indiceEPANET,EN_INITSTATUS,0);
}}
/*****************************************************************************/
/************************ BUSCA EN LA LISTA TABU  ********************************/
/*****************************************************************************/
int ListaTABU(int inicio, int tub, float *vector,int N){
int j=inicio;

for(;j<N;j+=4){
if(vector[j]==tub&&vector[j]!=0.0){
return j;
break;}}
return 0;
}
int RunEPANET(){int i, nodo,j,z=0; float
        pressure=0,presion,demand;presNeg=0,presPos=0;NValNeg=0,NValPos=0,NDiaPos=0,pr
        esOk=0,NTubeOk=0,NTubNegPos=0;
    char id[15];
ENsolveH();
    for (i=1; i<=NNODOS; i++){
        ENgetnodeid(i, id);
        ENgetnodevalue(i, EN_DEMAND, &demand);
        ENgetnodevalue(i, EN_PRESSURE, &pressure);
                if(pressure<=PRESIONMIN){
                        nodo=atoi(id);
                        nodosPRESNEG[presNeg]=nodo;
                        presNeg++;
                        }
                else {                  nodo=atoi(id);
        nodosPRESok[presOk]=nodo;       presOk++;
                }

                if(pressure>=PRESIONMAX){
                        nodo=atoi(id); nodosPRESPOS[presPos]=nodo;
                                                presPos++;              }
    }
                        GuardarPresNegativas();GuardarPresok();
                        GuardarPresPositivas();
                        GuardarPresPosNeg();
                        GuardarPresNegPos();
```

```
                    buscar_tuberia_Vdestino();
    if(presPos>=1 && presNeg>=1)     return 0;
    else if(presPos==0 && presNeg>=1)       return 2;
    else if(presPos>0 && presNeg==0) return 3;
   else if(presPos==0 && presNeg==0) return 1;
}
/*****************************************************************************/
/*************GUARDA Y BUSCA LAS TUBERIAS DE NODOS NEGATIVOS Y POSITIVOS ***********/
/*****************************************************************************/
void GuardarPresNegativas(){
int     i,z;
        NValNeg=0;
for(z=0;z<presNeg;z++){
        for(i=0;i<NTUBERIAS;i++)

if(nodosPRESNEG[z]==TUBERIAS[i].Vdestino||nodosPRESNEG[z]==TUBERIAS[i].Vorigen){
        if(NValNeg>0)
                if(tuberiaPRESNEG[NValNeg-1]==TUBERIAS[i].NTuberia)break;
                        tuberiaPRESNEG[NValNeg]=TUBERIAS[i].NTuberia;
                        NValNeg++;}}}


void GuardarPresPosNeg(){int i,z,j;
    for(z=0;z<presNeg;z++){
        for(i=0;i<NTUBERIAS;i++){
         if(nodosPRESNEG[z]==TUBERIAS[i].Vdestino){
           if(buscar_tuberia(TUBERIAS[i].Vorigen,nodosPRESok ,presOk)!=0){
                TuberiaPosNeg[NTubNegPos]=TUBERIAS[i].NTuberia;
                NTubNegPos++;
                              }            }
        if((nodosPRESNEG[z]==TUBERIAS[i].Vdestino)&&(TUBERIAS[i].Vorigen>NNODOS)){
                TuberiaPosNeg[NTubNegPos]=TUBERIAS[i].NTuberia;
                NTubNegPos++;
                          }                   }                       }}
void GuardarPresNegPos(){int i,z,j;NTubNegPosDia=0;NTubNegPosVal=0;
for(z=0;z<presPos;z++){
for(i=0;i<NTUBERIAS;i++){

   if(nodosPRESPOS[z]==TUBERIAS[i].Vdestino){
     if(buscar_tuberia(TUBERIAS[i].Vorigen,nodosPRESok ,presOk)!=0){
        if(buscar_tuberia(TUBERIAS[i].NTuberia, tuberiaPRESPOS,NValPos)!=0){
                TuberiaNegPosValulas[NTubNegPosVal]=TUBERIAS[i].NTuberia;
                NTubNegPosVal++;

else{TuberiaNegPosDiametro[NTubNegPosDia]=TUBERIAS[i].NTuberia;
                NTubNegPosDia++;} }
if((nodosPRESPOS[z]==TUBERIAS[i].Vdestino)&&(TUBERIAS[i].Vorigen>NNODOS)){

        TuberiaNegPosDiametro[NTubNegPosDia]=TUBERIAS[i].NTuberia;NTubNegPosDia++;
 }                    }                       }}
/*****************************************************************************/
/********************** GUERDA NODOS CON PRESIONES FACTIBLES ***********************/
/*****************************************************************************/
 void GuardarPresok(){
int     i,z,j;
for(z=0;z<presOk;z++){
        for(j=0;j<NTUBERIAS;j++){
           if(( nodosPRESok[z]==TUBERIAS[j].Vorigen)||(
nodosPRESok[z]==TUBERIAS[j].Vdestino)){
           if(buscar_tuberia(TUBERIAS[j].NTuberia, tuberiaPRESPOSOk,NTubeOk)!=0)break;
                           tuberiaPRESPOSOk[NTubeOk]=TUBERIAS[j].NTuberia;
                           NTubeOk++;
}}} }
/*****************************************************************************/
/***************************** GUARDA PRESIONES POSITIVAS ************************/
/*****************************************************************************/
 void GuardarPresPositivas(){
int i,z,j;
```

```
        for(z=0;z<presPos;z++){
                for(j=0;j<NTUBERIAS;j++){
if(nodosPRESPOS[z]==TUBERIAS[j].Vdestino||nodosPRESPOS[z]==TUBERIAS[j].Vorigen){
if(buscar_tuberia(TUBERIAS[j].NTuberia, tuberiaPRESPOSDia,NDiaPos)!=0)break;
                                tuberiaPRESPOSDia[NDiaPos]=TUBERIAS[j].NTuberia;
                                NDiaPos++;
                        }}

                for(i=0;i<NVALVULAS;i++)
if(nodosPRESPOS[z]==VALVULAS[i].H2Oarriba||nodosPRESPOS[z]==VALVULAS[i].H2Oabajo){
if(buscar_tuberia(TUBERIAS[j].NTuberia, tuberiaPRESPOS,NValPos)!=0)break;
                           tuberiaPRESPOS[NValPos]=VALVULAS[i].Ntuberia;
                        NValPos++;}}
                        }
/*****************************************************************************/
/*********************CREA ARCHIVO .INP PARA ENVIARLO A EPANET ********************/
/*****************************************************************************/
void Crear_archivo_INP1(int SOL,int inicial){
int     i,j,mirrorloss=0,nodo,tub,dep,val,t,emb,bandera=0;
float D;
char    nombre[60],
        status[10]="open",
        statusClose[10]="closed",
        pattern[10]=";",
        TIPO[]="PRV";
       FILE *fichero;
if(inicial==2)sprintf(nombre, "archivos/instancia_inicial.inp");
if(inicial==1)sprintf(nombre, "archivos/instancia.inp");
else sprintf(nombre, "soluciones/SOL-%d.inp", SOL);
        fichero = fopen(nombre,"wt");
        fprintf(fichero,"[TITLE]\nprueba\n[JUNCTIONS]\n");
        fprintf(fichero,";ID Nudo\tCota\tDemanda \tCurva de Modulac.\t \n");
for(nodo=0;nodo<NNODOS;nodo++){
        fprintf(fichero,"%d\t%.2f\t%.3f\t;\t\n",NODOS[nodo].idnodo,NODOS[nodo].
                 cota,NODOS[nodo].demanda);
                  }
        fprintf(fichero,"\n[RESERVOIRS]\n");
        fprintf(fichero,";ID Nudo\tAltura\t Curva modulac.\n");
for(emb=0;emb<NEMBS;emb++){

        fprintf(fichero,"%d\t%.2f\t;\n",EMBALSES[emb].id_dep,EMBALSES[emb].altura);
          }
        fprintf(fichero,"\n[TANKS]\n");
        fprintf(fichero,";IDnudo\tCota\tNivelIni\tNivelMin\tNivelMax
                 \tDiametro\tVolMen\tCurvCubi\n");

        fprintf(fichero,"\n[PIPES]\n");
        fprintf(fichero,";ID Linea\tNudo1\tNudo2\tLongitud
                   \tDiametro\tRugosidad\tPerdMen\tEstado\n");

for(t=0;t<NTUBERIAS;t++){
        bandera=0;
for(i=0;i<NVALVULAS;i++){
if(TUBERIAS[t].NTuberia==VALVULAS[i].Ntuberia&&VALVULAS[i].ExisteValvula==1){
        bandera=1;
                break;}}
if(BOMBAS[t].ExisteBomba==1)
                bandera=1;
        ENgetlinkvalue(t+1,EN_STATUS,&D);
if(inicial==1) bandera=0;
if(bandera==1) fprintf(fichero,"%d\t%d\t%d\t%.2f\t%.2f\t%d\t%d\t%s\t%s\t\n",
        TUBERIAS[t].NTuberia,TUBERIAS[t].Vorigen,TUBERIAS[t].Vdestino,
        TUBERIAS[t].longitud,TUBERIAS[t].diametro,CoeRug,0,statusClose,pattern);
else
        fprintf(fichero,"%d\t%d\t%d\t%.2f\t%.2f\t%d\t%d\t%s\t%s\t\n",
        TUBERIAS[t].NTuberia,TUBERIAS[t].Vorigen,TUBERIAS[t].Vdestino,TUBERIAS[t].long
        itud,TUBERIAS[t].diametro,CoeRug,0,status,pattern);
          }
```

```
        fprintf(fichero,"\n[PUMPS]\n");
        fprintf(fichero,";ID \tNudoAsp\tNudoImp\tPar�metros\n");
for(t=0;t<NTUBERIAS;t++){if(BOMBAS[t].ExisteBomba==1||inicial==1){
        fprintf(fichero,"%d\t%d\t%d\tPOWER %f\t%s\t\n",BOMBAS[t].id_bom,
                 BOMBAS[t].nodoAsp,BOMBAS[t].nodoImp,BOMBAS[t].potencia,pattern);
          }}
          fprintf(fichero,"\n[VALVES]\n");
          fprintf(fichero,";ID
\tNudoAgArr\tNudoAgAbj\tDi�metro\tTipo\tConsigna\tP�rdMen\n");


for(t=0;t<NVALVULAS;t++){if(VALVULAS[t].ExisteValvula==1||inicial==1){

        fprintf(fichero,"%d\t%d\t%d\t%0.2f\t%s\t%.2f\t%d\n",VALVULAS[t].id_val,
               VALVULAS[t].H2Oarriba,VALVULAS[t].H2Oabajo,TUBERIAS[VALVULAS[t].Ntuberi
               a-1].diametro,TIPO,VALVULAS[t].consigna,VALVULAS[t].perdidas);
          }}
        fprintf(fichero,"\n[STATUS]\n");
        fprintf(fichero,";Link\tStatus/Setting\n");
for(t=0;t<NTUBERIAS;t++){if(BOMBAS[t].ExisteBomba==1){


         fprintf(fichero,"%d\tOPEN\t%f\n",BOMBAS[t].id_bom,BOMBAS[t].potencia);}
                        else if(inicial==1)
         fprintf(fichero,"%d\tCLOSED\n",BOMBAS[t].id_bom);
          }
for(t=0;t<NVALVULAS;t++){if(VALVULAS[t].ExisteValvula==1){
         fprintf(fichero,"%d\tOPEN\t%f\n",VALVULAS[t].id_val,VALVULAS[t].consigna);}
else if(inicial==1)
        fprintf(fichero,"%d\tCLOSED\n",VALVULAS[t].id_val);}
        fprintf(fichero,"\n[OPTIONS]\n");
        fprintf(fichero,"Units\t%s\n",Units);
        fprintf(fichero,"Headloss\t%s\n",Headloss);
        fprintf(fichero,"Specific Gravity \t%f\n",Specific_Gravity);
        fprintf(fichero,"Viscosity\t%f\n", Viscosity);
        fprintf(fichero,"Trials\t%d\n",Trials);
        fprintf(fichero,"Accuracy\t%.3f\n",Accuracy);
        fprintf(fichero,"Unbalanced\t%s\n",Unbalanced);
        fprintf(fichero,"\n[COORDINATES]\n");
        fprintf(fichero,";ID Nudo\tCoord X\tCoord Y\n");
for(i=0;i<NNODOS;i++){
          fprintf(fichero,"%d\t%.2f\t%.2f\t;\n",NODOS[i].idnodo,NODOS[i].
                  coordX,NODOS[i].coordY);
          }
for(emb=0;emb<NEMBS;emb++){
        fprintf(fichero,"%d\t%.2f\t%.2f\t;\n",EMBALSES[emb].id_dep,
                  EMBALSES[emb].coordX,EMBALSES[emb].coordY);
          }
for(dep=0;dep<NDEPS;dep++){

        fprintf(fichero,"%d\t%.2f\t%.2f\t;\n",DEPOSITOS[dep].id_dep,DEPOSITOS[dep].coo
        rdX,DEPOSITOS[dep].coordY);
          }
          fprintf(fichero,"\n[VERTICES]\n");
          fprintf(fichero,";ID linea\tCoord X\tCoord Y\n");
          fprintf(fichero,"\n[LABELS]\n");
          fprintf(fichero,";Coord X\tCoord Y\tR�tulo y Nudo Anclaje\n");
          fprintf(fichero,"\n[BACKDROP]\n");
          fprintf(fichero,"DIMENSIONS\n");
          fprintf(fichero,"UNITS\t None\n ");
          fprintf(fichero,"FILE\n");
          fprintf(fichero,"\n[END]\n");

      fclose(fichero);
}
/*****************************************************************************/
/************************* EVALUA COSTOS DE TODAS LAS TUBERIAS *****************/
```

```
/*****************************************************************************/
void EvaluaCostoTuberia(float *A){
  int i,j;
  float D=0,B=0,C=0;
    for(i=0;i<NTUBERIAS;i++){

      for(j=0;j<NDIAMCOMERS;j++){
          if(TUBERIAS[i].diametro==DIACOMER[j].Diam)
          break;
    }

  D+=DIACOMER[j].CostoDiam*TUBERIAS[i].longitud;
    }
                for(i=0;i<NTUBERIAS;i++){
                     if (BOMBAS[i].ExisteBomba==1){
                  for(j=0;j<NTUBERIAS;j++){
                     if(BOMBAS[i].potencia==BombasComerPotencia[j])break;
                     }
                     B+=BombasComerCosto[j];
          }

}
                for(i=0;i<NVALVULAS;i++){
                     if(VALVULAS[i].ExisteValvula==1){
                for(j=0;j<NVALVULAS;j++){
                     if(TUBERIAS[VALVULAS[i].Ntuberia-
1].diametro==DIACOMER[j].Diam)break;
                     }
                       C+=ValvulasComerCosto[j];
          }

}
 *A=B+D+C;
}
/*****************************************************************************/
/******************** CALCULA TIEMPO DE EJECUCION ********************************/
/*****************************************************************************/
float GuardarTiempo(time_t inicio, time_t final,int rank){
char   nombre[20];
float  f_Tiempo=0,f_Tiempo1=0;
FILE *fichero;
final=time(NULL);
f_Tiempo=difftime(final,inicio);
f_Tiempo1=f_Tiempo/60;
return(f_Tiempo1);
}

void guarda_archivo_txt(float y, float x,char *mje){

    FILE *fout= fopen(mje,"a");
    if (!fout) {
        printf("\nERROR AL LEER FICHERO TXT");
    }
    else {
                fprintf(fout,"%d\t %f\n",y,x);
    }
    fclose(fout);
}
/*****************************************************************************/
/***************************GUARDA SOLUCION ANTERIOR*****************************/
/*****************************************************************************/
void Solucion_Anterior(int *bandera44){
  int  i,j0=0,j1=1,j2=2,j3=3;

for(i=0;i<cambios;i++){
switch(bandera44[i]){
               case 1:{
```

```
        funcion(CambiosTABU_Diametros[j1], 0, CambiosTABU_Diametros[j0],
        CambiosTABU_Diametros[j2],7);
        CambiosTABU_Diametros[3]=bandera;
        TUBERIAS[(int)CambiosTABU_Diametros[j1]-1].diametro=CambiosTABU_Diametros[j2];
               break;
          }
               case 2:{
        funcion(CambiosTABU_Bombas[j1], 0, CambiosTABU_Bombas[j0],
        CambiosTABU_Bombas[j2],1);
        BOMBAS[(int)CambiosTABU_Bombas[j1]-1].ExisteBomba=1;
        BOMBAS[(int)CambiosTABU_Bombas[j1]-1].potencia=CambiosTABU_Bombas[j2];
                                                    CambiosTABU_Bombas[j3]=bandera;

               break;
          }
               case 3:{
        QuitarBomba(CambiosTABU_Bombas[j1]);
        Función    (CambiosTABU_Bombas[j1], 1, CambiosTABU_Bombas[j0],
        CambiosTABU_Bombas[j2],0);
        BOMBAS     [(int)CambiosTABU_Bombas[j1]-1].ExisteBomba=0;
        CambiosTABU_Bombas[j3]=bandera;
               break;
          }
               case 4:{

        funcion(CambiosTABU_Valvulas[j1], 0, CambiosTABU_Valvulas[j0],
        CambiosTABU_Bombas[j2],1);

VALVULAS[(int)CambiosTABU_Valvulas[j0]-NTUBERIAS-NTUBERIAS-1].ExisteValvula=1;
VALVULAS[(int)CambiosTABU_Valvulas[j0]-NTUBERIAS-NTUBERIAS-1].consigna=
        CambiosTABU_Bombas[j2];
        CambiosTABU_Valvulas[j3]=bandera;

               break;
          }
               case 5:{
        funcion(CambiosTABU_Valvulas[j1], 1, CambiosTABU_Valvulas[j0],
        CambiosTABU_Bombas[j2],0);
VALVULAS[(int)CambiosTABU_Valvulas[j0]-NTUBERIAS-NTUBERIAS-1].ExisteValvula=0;
        CambiosTABU_Valvulas[j3]=bandera;
               break;
          }
        }
        j0+=4;
        j1+=4;
        j2+=4;
        j3+=4;
}}
/*****************************************************************************/
/********* COPIA DE LAS ESTRUCTURAS PARA INICIAR CON LA MISMA SOLN *****************/
/*****************************************************************************/
void Guardar_Solucion_Inicial(tuberia *Tub,bomba *Pums,valvula *Vals,nodo *Node,
                              tuberia *Tub2,bomba *Pums2,valvula *Vals2,nodo *Node2){
   int i;
   for(i=0;i<NTUBERIAS;i++){
                              Tub[i] =Tub2[i];
                              Pums[i]=Pums2[i];
                               }
   for(i=0;i<NVALVULAS;i++){
                              Vals[i]=Vals2[i];
                               }
   for(i=0;i<NNODOS;i++){
                              Node[i]=Node2[i];
   } }
/*****************************************************************************/
/************************FUNCION DE APRENDIZAJE*********************************/
/*****************************************************************************/
void ordenarHeurist(double *SumCosto){
```

```
int     k,w;

TotalHeuris++;
w=TotalHeuris*2;
*SumCosto=0;
HeuristicRun[TotalHeuris*2-1]=1000;

for(k=0;k<(TotalHeuris);k++){
                                *SumCosto+=w;
                                w-=2;

}
HeuristicRun[TotalHeuris*2-2]=HeuristicRun[TotalHeuris*2-4]+5;
}
```