UNIVERSIDAD DE TALCA
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA CIVIL EN COMPUTACIÓN

Herramienta para la optimización de redes de distribución de agua potable

GABRIEL GONZALO ALEXANDER SANHUEZA FUENTES

Profesor Guía: JIMMY GUTIERREZ BAHAMONDES

Profesor Co-guía: DANIEL MORA MELIA

Memoria para optar al título de
Ingeniero Civil en Computación

Curicó – Chile
mes, año

UNIVERSIDAD DE TALCA
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA CIVIL EN COMPUTACIÓN

Herramienta para la optimización de redes de distribución de agua potable

GABRIEL GONZALO ALEXANDER SANHUEZA FUENTES

Profesor Guía: JIMMY GUTIERREZ BAHAMONDES

Profesor Co-guía: DANIEL MORA MELIA

Profesor Informante: PROFESOR INFORMANTE 1

Profesor Informante: PROFESOR INFORMANTE 2

Memoria para optar al título de
Ingeniero Civil en Computación

El presente documento fue calificado con nota: _____

Curicó – Chile

mes, año

Dedicado a ...

AGRADECIMIENTOS

Agradecimientos a ...

TABLA DE CONTENIDOS

página

ÍNDICE DE FIGURAS

página

ÍNDICE DE TABLAS

página

RESUMEN

La escasez de agua potable es un problema a nivel mundial. Cada día aumenta la necesidad de utilizar eficientemente los recursos hídricos disponibles. Sin embargo, la optimización de todos los procesos involucrados en su gestión es una tarea compleja. Especialmente, la distribución del agua es un procedimiento difícil de optimizar debido a que implica el modelamiento de variables físicas que se relacionan de manera no lineal [2]. Diversos investigadores han abordado el problema desde diferentes perspectivas. Donde una gran cantidad de trabajos ha destacado a los algoritmos metaheurísticos como herramientas eficientes frente a la resolución de problemas complejos de esta área. Debido principalmente a su capacidad de exploración del espacio de posibles soluciones en tiempos razonables. A pesar de ello, existen muy pocos sistemas computacionales que permitan a los ingenieros hidráulicos optimizar el diseño y operación de las redes de distribución de agua (RDA) desde una interfaz gráfica, sin tener que codificar.

Es por ello que este proyecto busca llevar a cabo el desarrollo de una herramienta que permita la optimización de problemas presentes en RDA. En términos generales, el objetivo es diseñar e implementar una aplicación de escritorio que pueda ser utilizada por personas interesadas en la labores del diseño y operaciones de RDA. A pesar de que existe muchos problemas relacionados con esta temática, este proyecto se centrará en dos. Los cuales han sido seleccionados por un grupo de expertos que apoya este proyecto. El primero corresponde a la optimización del diseño de RDA basado en la minimización del costo de inversión según la selección de diámetros de tuberías. En segundo lugar, el problema de planificaciones de las operaciones de los sistemas de bombeo que optimiza simultáneamente los costos energéticos y de mantenimiento de los equipos. Para ello, el proyecto contempla la implementación de un Algoritmo Genético (GA) [11] para la resolución de problemas monoobjetivos, el algoritmo "Nondominated Sorting Genetic Algorithm II"(NSGA-II) [8] para el caso de problemas multiobjetivos y todos los métodos que permiten realizar las simulaciones hidráulicas de RDA.

Adicionalmente, es importante destacar que este nuevo sistema debe poseer una arquitectura escalable que sirva como base para la adición de nuevos problemas y algoritmos a medida que sean requeridos.

A continuación se presentan los conceptos básicos que son necesarios para el desarrollo de este proyecto. Posteriormente, se presenta el problema y contexto del proyecto. Finalmente, se presenta la propuesta de solución al problema identificado.

Para el desarrollo de este proyecto se opto por utilizar la metodología de desarrollo Iterativo e Incremental.

Para la evaluación del sistema a desarrollar se usara la metodología de caso de estudio cuyo resultado permitirá determinar la utilidad del software desarrollado.

1. Introducción

Este capítulo tiene como objetivo presentar el contexto, el problema, la propuesta de solución, los objetivos y el alcance del proyecto. Así como también, entregar antecedentes sobre los trabajos relacionados al tema.

1.1. Contexto del proyecto

La escasez de agua potable es sin duda una problemática a nivel mundial y la optimización de los sistemas que permiten su distribución es cada día más relevante. Existe una serie de problemáticas asociadas a la determinación de las condiciones óptimas de operaciones y las características adecuadas para su construcción. Las RDA son redes que pueden ser muy extensas y complejas. Forman parte de la estructura principal de cualquier ciudad. Deben ser capaces de adaptarse a los cambios y asegurar niveles mínimos de servicios durante las 24 horas del día [17]. Adicionalmente, dependiendo de su topología, las RDA integran sistemas de bombeo que requieren gran cantidad de energía en horarios determinados.

La optimización de estos sistemas, a la vez, involucra la participación de múltiples criterios que deben ser tomados en cuenta a la hora de decidir. Sin embargo, la incorporación de éstos, involucra la generación de modelos cada vez más complejos [24].

Por lo anteriormente mencionado, esta área, ha llamado la atención de muchos investigadores que han creado diversos métodos para resolver la problemática desde diferentes enfoques. Sin embargo, aún existen muy pocas aplicaciones computacionales que permitan emplear los nuevos modelos y técnicas de forma práctica. Esto supone un gran problema para los interesados en aplicar estos conocimientos en un

contexto real. Generalmente se trata de personas instruidas en temáticas relacionadas con la hidráulica pero que poseen un escaso manejo de técnicas computacionales de optimización.

En este trabajo se pretende dar respuesta a esa necesidad creciente a través del diseño e implementación de una aplicación de escritorio. Este nuevo sistema, permitirá a a los usuarios resolver dos de los principales problemas en la optimización de RDA. Para ello utilizará un algoritmo genético, en el caso de los problemas con solo un objetivo y NSGA-II para resolver problemas del tipo multiobjetivo.

1.2. Presentación del problema

Los encargados de implementar sistemas de distribución de agua potable, no cuentan con suficientes herramientas y tiempo para su correcta gestión. Por lo tanto, no es posible utilizar los recursos asociados de forma eficiente. Además, las herramientas existentes no satisfacen las necesidades de usabilidad y costo, debido a que son poco intuitivas y de pago.

El escoger las especificaciones de una red de agua potable es una tarea difícil debido a que hay que evaluar el rendimiento general del sistema en función de un conjunto de variables que se mueven en un rango muy elevado de posibilidades. Debido a esto, el uso de herramientas que optimicen la selección de estas características puede ayudar considerablemente a reducir costos operaciones y de inversión.

Finalmente, es importante destacar que la construcción de un sistema que permita realizar la optimización de RDA es compleja. Necesita del conocimiento técnico de expertos en el área de hidráulica y computación. Involucra el trabajo con programas de simulación computacional que modelan las características de los sistema de agua bajo presión y de algoritmos metaheurísticos que los subordinen.

1.3. Objetivos

Para abarcar la problemática se fijan los siguientes objetivos

Objetivo general

- Diseñar y desarrollar una aplicación de apoyo a la toma de decisiones, integrando algoritmos de optimización aplicados al problema de diseño y operación en

redes de distribución de agua potable.

Objetivos especificos

1. Generar soluciones frente al problema monoobjetivo de diseño de redes de distribución de agua potable a través de la implementación de algoritmos genéticos.

2. Generar soluciones frente al problema multiobjetivo de operación de redes de distribución de agua potable a través de la implementación de NSGA-II.

3. Diseñar he implementar la interfaz gráfica del sistema de optimización de redes de agua potable desarrollado durante este proyecto.

4. Evaluar el desempeño de los algoritmos, contrastando los resultados obtenidos en redes de benchmarking con óptimos conocidos.

1.4. Propuesta de solución

La solución que se propone para abordar el problema consiste en el desarrollo de una aplicación que permita buscar soluciones a dos de los problemas existentes en las redes de agua potable. Además, de hacer uso de una arquitectura que permita fácilmente extender el programa para abarcar nuevos problemas. Para lograr esto último la aplicación debe quedar bien documentada. Por lo tanto, lo que se tendrá al termino del desarrollo del proyecto será un software escalable que tratara con dos problemas relacionados a la distribución de agua potable y que podrá ser ampliado en futuros trabajos.

Los problemas que se abordaran en el contexto de optimización de redes de agua potable serán:

- **Problema de diseño:** Se trata este problema desde un enfoque monoobjetivo implementando un algoritmo genético que buscará la optimización de los costo de inversión variando el diámetro de las tuberías.

- **Problema de operación:** Este problema se abordara desde el enfoque multiobjetivo. Para esto se implementara el algoritmo NSGA-II y se buscará la

optimización de los objetivos de costos energéticos y el número de encendidos y apagados de las bombas.

Tanto el algoritmo genético como el NSGA-II, permiten la utilización de distintos operadores de cruzamiento y mutación que también serán implementados para ser utilizados.

La solución planteada supone además el diseño e implementación de una interfaz gráfica para ayudar al usuario en el uso de la herramienta.

1.5. Alcances

Los alcances propuestos para este proyecto serán los siguientes:

- El sistema permitirá la carga y la visualización de la red gráficamente.
- El sistema permitirá visualizar la configuración básica almacenada en el archivo inp de los elementos de la red.
- El sistema solo resolverá dos clases de problemas de optimización, uno monoobjetivo y el otro multiobjetivo. El problema monoobjetivo será el de los costos de inversión. En cuanto al problema multiobjetivo, este será el de los costos energéticos y el número de encendidos y apagado de las bombas.
- El sistema únicamente contara con dos algoritmos implementados los cuales serán el algoritmo genético y NSGA-II. El algoritmo genético será el usado para tratar el problema monoobjetivo, mientras que NSGA-II será aplicado al multiobjetivo.
- El sistema permitirá visualizar el o los resultados obtenidos al finalizar la ejecución del algoritmo.
- El sistema permitirá generar archivos .inp con la configuración de la solución obtenida a través de los algoritmos. Debido a que el archivo .inp establece una gran cantidad de configuraciones, las únicas que se permitirán modificar serán las configuraciones asociadas a los conexiones (junctions) y tuberías (pipes).
- El sistema permitirá generar archivos con las soluciones obtenidas para cada problema, es decir, el valor de los objetivos y las variables de decisión involucradas.

- El sistema permitirá graficar visualmente las soluciones en un plano cartesiano.
- La gráfica únicamente estará disponible en problemas con uno o dos objetivos.
- La comparación con las redes de benchmarking consistirá únicamente, en presentar una tabla comparativa entre los resultados presentes en la literatura y los obtenidos a través de nuestro sistema, para cada uno de los problemas de redes de distribución de agua potable de nuestro sistema.

Este proyecto no contempla la creación de la red por lo que estas deberán ser ingresadas como entradas al programa, es decir, estas deberán ser creadas usando EPANET o manualmente, pero siguiendo el formato establecido por EPANET. Además, esta herramienta únicamente podrá ser ocupada en equipos cuyo sistema operativo sea Windows debido a que se realizan llamadas a librerías nativas.

1.6. Trabajo relacionado

En este apartado se mostrarán distintas herramientas y el enfoque utilizado para resolver el problema con ellas. En general, los enfoques consisten en usar software ya disponible o crear un software personalizado.

- **Magmoredes**: En [9] se describe la existencia de un software de diseño basado en micro-algoritmos genéticos multiobjetivos, que de acuerdo al autor, tiene un mejor rendimiento y es más eficiente que el algoritmo NSGA-II. Esto se debe a que requiere una menor cantidad de memoria y tiene un mejor tiempo de cómputo. Este programa puede cargar cualquier red y realiza los cálculos utilizando librerías de java. Las funciones objetivos que este sistema intenta resolver son la optimización de los costo y la confiabilidad final de la red.
- **WaterGEMS**: Software comercial que permite la construcción de modelos geoespaciales; optimización de diseño, ciclos de bombeo y calibración automática del modelo; y la gestión de activos. Este software a sido usado en [14] para llevar a cabo las simulaciones necesarias para su estudio. La metodología seguida para la utilización de este sistema consiste en ingresar los datos a WaterGEMS para correr las simulaciones. La limitación de este programa es que no permite la adición de nuevos algoritmos por parte del usuario, en el caso

de que se quiera probar o mejorar algún algoritmo ya existente. Sin embargo, este sistema también tiene sus ventajas, porque ya incorpora algunos algoritmos predefinidos para resolver ciertos problemas. Además, de que cuenta con diversas funcionalidades como conexión con datos externos, operaciones de análisis espaciales, intercambio de datos con dispositivos o programas de administración, entre otros.

- **EPANET**: El enfoque seguido con la utilización de esta herramienta consiste en la automatización al momento de ejecutar los algoritmos y la posterior evaluación de los resultados utilizando la librería EPANET Toolkit. Este es usado en [16] y consiste en la creación de un programa el cual implementa algoritmos metaheurísticos y los resultados obtenidos por estos son enviados a la EPANET Toolkit para evaluar la solución y determinar la factibilidad de ésta. La ventaja de este enfoque es que permite una mayor flexibilidad en los algoritmos metaheuristicos utilizados y los problemas que se quieren resolver. Sin embargo, debido a que se necesita implementar los problemas y los algoritmos, este enfoque toma mucho tiempo.

Para el desarrollo de este proyecto se usa el enfoque basado en EPANET, puesto que es una librería de simulación hidráulica ampliamente utilizada y permite enfocarnos en la resolución de los problemas usando algoritmos metaheurísticos. Tanto Magmoredes como WaterGEMS buscan resolver temas concretos en los sistemas de distribución de agua potable y puesto que el código de estos programas no esta disponible públicamente o son un sistema que se comercializa sin permitir la modificación del sistema por parte de terceros, se busca con nuestro proyecto incorporar esta capacidad para que en futuros trabajos se pueda abarcar una mayor cantidad de problemas con nuestro sistema.

2. Marco Teórico

En este capítulo se presentan los conceptos que serán necesarios para la realización del proyecto.

2.1. Metodología de desarrollo

Para el desarrollo de este proyecto se opto por la metodología iterativo e incremental. La elección de esta metodología se debe al hecho de que este genera suficiente documentación para ser utilizada en el desarrollo de futuros proyectos derivados de este trabajo.

2.1.1. Metodología iterativo e incremental

El desarrollo iterativo e incremental es una metodología la cual lleva a cabo el desarrollo de un proyecto de software dividiéndolo en iteraciones que generan un incremento, el cual contribuye en el desarrollo del producto final.

Cada iteración se compone de las fases de análisis, diseño, implementación y pruebas como se muestra en la Figura 2.1. La fase de análisis se encarga de llevar a cabo la obtención y definición de los requerimientos del software. Durante la etapa de diseño se realiza la conceptualización del software basado en los requerimientos definidos anteriormente. Durante la implementación se codifican las funcionalidades siguiendo las directivas establecidas durante el diseño, con el fin de satisfacer los requerimientos. Y finalmente, durante la fase de pruebas, se valida y verifica la correctitud de las funcionalidades implementadas, así como el cumplimiento de los requisitos.

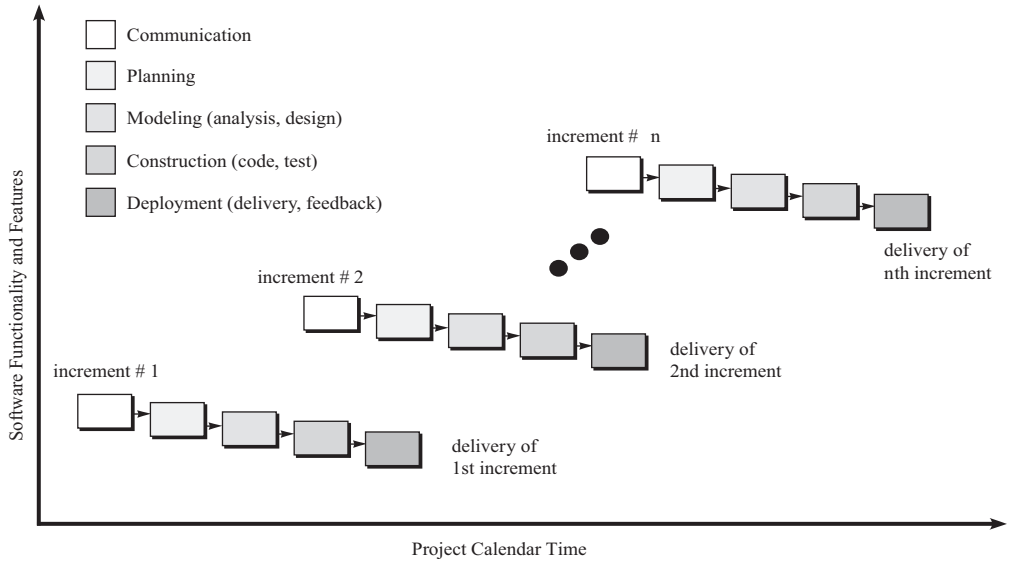

Figura 2.1: Metodología iterativo e incremental [18]

El hecho de llevar a cabo un desarrollo iterativo permite la obtención de retroalimentación del producto que se esta desarrollando tempranamente y de esta manera poder refinar el trabajo en etapas posteriores del desarrollo. [26, 15, 13, 1].

2.2. Metodología de evaluación

2.2.1. Casos de estudio

Un caso de estudio [22] es una metodología de investigación la cual en ingeniería de software permite analizar un proyecto, un grupo de personas, un producto, etc, en un contexto real con el objetivo de responder la pregunta de investigación planteada. Esta metodología de evaluación considera aspectos formales para obtener evidencia. Los principales aspectos son:

- Describir el contexto de aplicación del caso: Consiste en establecer sobre que elemento se aplicara el caso de estudio a desarrollar.
- Definición de objetivos experimentales: Consiste en indicar cual es el objetivo de la investigación a realizar, si es describir, evaluar o explicar algún suceso.
- Definición de un protocolo para conducir el caso de estudio: Consiste en escoger las pautas para llevar a cabo el caso de estudio, que instrumentos serán

utilizados para recolectar datos y como se realizaran el análisis de estos.

- Definición de características a evaluar: Consiste en establecer que es lo que estamos interesados en evaluar del elemento sobre el que se aplica el caso de estudio.
- Definición de sujetos de prueba: Consiste en indicar cual sera la fuente de datos a ser utilizada para el caso de estudio, estas pueden ser personas, datos ya recolectados, etc.
- Aplicación de caso de estudio en un conjunto de sesiones no controladas: Consiste en llevar a cabo el caso de estudio sobre los sujeto de prueba definidos.
- Aplicación de herramientas de obtención de evidencia empírica: Consiste en la utilización de métodos o técnicas con el fin de obtener evidencia empírica a partir de los datos recolectados.
- Análisis y evaluación de datos empíricos: Consiste en analizar la evidencia y evaluar la validez de los resultados obtenidos.

2.3. Herramientas para la implementación del software

Java

Java es un lenguaje de programación de alto nivel orientado a objetos y de propósito general. Un programa java se ejecuta sobre la maquina virtual llamada la Java Virtual Machine, la cual le da a este lenguaje la característica de ser multi-plataforma. Adicionalmente, java incorpora el soporte para multi-hilos, una poderosa herramienta que permite la ejecución de distintas instrucciones de código al mismo tiempo [10]. Ademas. este lenguaje también incorpora una característica conocida como el recolector de basura, el cual se encarga de limpiar la memoria de objetos que ya no están siendo utilizados. Fue anunciado por Sun Microsystems en Mayo de 1995 [23].

Java Reflection

Característica de java que permite que un programa se auto examine. Esta característica está disponible a través de la Java Reflection API, la cual cuenta con

métodos para obtener los meta-object de las clases, métodos, constructores, campos o parámetros. Esta API también permite crear nuevos objetos cuyo tipo era desconocido al momento de compilar el programa [7].

Java Annotation

Característica de java para agregar metadatos a elementos de java (clases, métodos, parámetros, etc.) [7]. Las anotaciones no tienen efecto directo sobre el código, pero cuando son usadas junto con otras herramientas pueden llegar a ser muy útiles. Estas herramientas pueden analizar estas anotaciones y realizar acciones en base a estas, por ejemplo, generar archivos adicionales como clases de java, archivos xml, entre otras; ser analizadas durante la ejecución del programa vía Java Reflection, para crear objetos cuyo tipo no conocemos en tiempo de compilación; etc.

2.4. Red de distribución de agua potable

Conjunto de elementos enlazados de tal manera que permite suministrar cierta cantidad de agua a una presión establecida [16].

Caudal

Cantidad de agua que se mueve a través de un segmento de la red.

2.4.1. Componentes físicos de una red

A continuación se define los componentes que conforman una red de agua potable [20], los cuales se aprecian en la Figura 2.2:

Nudos de caudal

Son los puntos o extremos de una tubería, los cuales también permiten que estas se unan. Estos nudos pueden actuar como nudos de demanda a través de los cuales el flujo abandona la red.

Embalse

Es una fuente de alimentación externa.

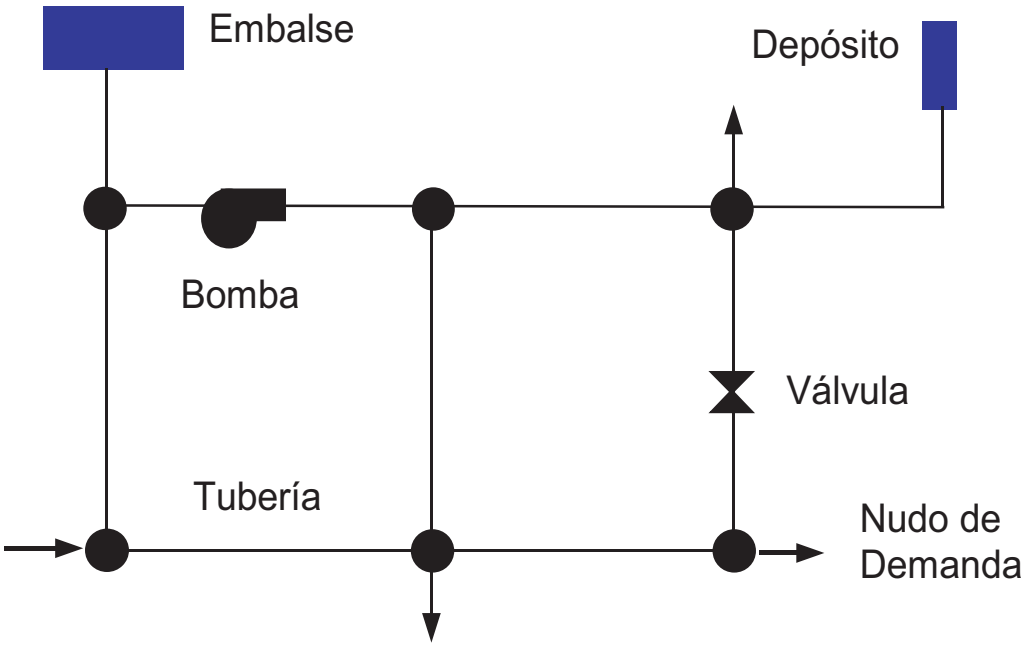

Figura 2.2: Componentes físicos de un sistema de distribución de agua [20]

Deposito

Son elementos con la capacidad de almacenar agua.

Tubería

Son los elementos a través de los cuales transita el agua de un nudo a otro.

Bomba

Elementos que permiten impulsar el liquido con el fin de elevarlo a una posición superior.

Válvula

Elementos que limitan la presión o el caudal que transita en un punto de la red.

2.4.2. Epanet

Software que permite simular el comportamiento hidráulico y la calidad del agua en redes de distribución de aguas compuesta por tuberías, nodos, bombas, válvulas

y tanques de almacenamiento [20]. Este software cuenta también con una librería dinámica conocida bajo el nombre de Epanet Programming Toolkit, la cual cuenta con un conjunto de funciones para realizar simulaciones desde diferentes entornos de desarrollo como C, C++, VB, Java, etc [21].

2.5. Optimización

La optimización consiste en maximizar o minimizar un conjunto de funciones que matemáticamente pueden ser expresadas de la siguiente forma:

$$f_1(x), f_2(x), ..., f_N(x),\ x = (x_1, ..., x_d)|x \in X$$

sujeto a una serie de condiciones

$$h_j(x) = 0, j = 1, 2, ..., J$$

$$g_k(x) \leq 0, k = 1, 2, ..., K$$

siendo $f_1, ..., f_N$ funciones objetivos; $x_1, ..., x_d$ variables de decisión, pertenecientes al espacio de búsqueda X; y h_j junto con g_k, una serie de restricciones [27]. De acuerdo a la cantidad de funciones objetivos que se tenga, se establece que si $N = 1$ la optimización es **monoobjetivo**, mientras que para $N \geq 2$ se conoce como **multiobjetivo** [27]. En este punto se debe tener en cuenta que los objetivos planteados deben encontrarse en contradicción.

Debido a la definición de las restricciones es posible dividir el espacio de búsqueda en dos regiones [6]:

- Soluciones factibles: Compuesto por los elementos pertenecientes al espacio de búsqueda que satisfacen todas las restricciones.
- Soluciones no factibles: Integrado por aquellos elementos que no complen todas las restricciones.

2.6. Heurística

Es el conocimiento que se tiene del problema el cual permite acotar la búsqueda de las soluciones en espacios de búsqueda de gran tamaño que hacen inviable la

aplicación de técnicas deterministas por el costo de tiempo que implican. Con la utilización de estas técnicas se espera encontrar soluciones buenas en un tiempo razonable, pero esto no esta garantizado [27, 19].

2.7. Metaheurística

Algoritmos que permiten resolver un amplio rango de problemas de optimización empleando técnicas con algún grado de aleatoriedad para encontrar soluciones a un problema. Estos algoritmos no garantizan que la solución encontrada sea la óptima, pero permiten obtener generalmente aproximaciones a esta. La diferencia entra heurísticas y metaheurísticas, es que esta ultima puede ser aplicado a un amplio conjunto de problemas sin necesidad de realizar grandes cambios en el algoritmo, mientras que las heurísticas generalmente son aplicadas a un dominio especifico [27, 5, 12].

2.7.1. Algoritmos Evolutivos

Conjunto de algoritmos inspirado en la teoría de la evolución de Darwin acerca de la capacidad de la naturaleza para evolucionar seres vivos bien adaptados a su entorno. Estos algoritmos hacen uso de diversos mecanismos entre los que se encuentra la selección, mutación y cruzamiento sobre los individuos de una población con el fin de generar una nueva generación de individuos [5]. En la Figura 2.3 muestra un un pseudocodigo de los pasos generales de un algoritmo evolutivo.

Algorithm 1: Algoritmo Evolutivo

```
1 población ← crearPoblaciónInicial()
2 evaluarPoblación(población)
3 while la condición de termino no ha sido alcanzada do
4 |   poblacionSeleccionada ← selección(población)
5 |   poblaciónDecendiente ← cruzamiento(poblacionSeleccionada)
6 |   poblaciónDecendiente ← mutación (poblaciónDecendiente)
7 |   poblaciónDecendiente ← evaluarPoblación (poblaciónDecendiente)
8 |   población ← remplazar (población, poblaciónDecendiente)
9 end
```

Figura 2.3: Pseudocodigo algoritmo evolutivo

Primero, se crea la población inicial. Luego, se evalúan los objetivos de dicha población y se itera hasta que la condición de termino haya sido alcanzada. La condición de término puede ser, por ejemplo un máximo numero de evaluaciones o evaluaciones sin mejoras en los resultados. Dentro del ciclo se realiza la selección sobre la población con el fin de determinar las soluciones que serán usadas en los operadores de cruzamiento y mutación. Finalmente, se remplaza la población inicial con la descendiente.

Población

Conjunto de soluciones candidatas sobre las cual opera el algoritmo. Durante cada iteración del algoritmo se generan nuevas soluciones que son agregadas a la población a la vez que se remueven otras. Una solución en la población se conoce como individuo [11].

Los individuos pueden ser representados de diversas maneras, entre ellas se encuentra la representación binaria (1 y 0), la real(Los números reales), etc. Para la representación binaria cada variable se codifica como un conjunto de bits, lo cual forma una cadena binaria. Por ejemplo, la representación binaria de la solución $(2, 4, 6, 8)$ formada por enteros de 4 bits correspondería a

$$0010\ 0100\ 0110\ 1000$$

En cambio, para la representación real esta se presenta como un vector, en donde cada valor que forma este vector pertenece a los números reales, es decir,

$$v = (x1, x2, \ldots, x_n), \text{en donde } v \in \mathbb{R}^n$$

Selección

La selección es un mecanismo utilizado por los algoritmos evolutivos para escoger a los individuos mas aptos los cuales serán usados para la reproducción [11]. Existen numerosos algoritmos de selección que pueden ser utilizados en los algoritmos evolutivos.

Cruzamiento

El cruzamiento es un mecanismo usado para generar nuevos soluciones a partir de dos o mas individuos seleccionados [4, 5].

Mutación

La mutación es un operador el cual permite mantener la diversidad en la descendencia [5] realizando modificaciones en ciertas partes de la solución.

2.7.2. Algoritmo Genético

El algoritmo genético es una estrategia de búsqueda de soluciones. Para realizar esto, el algoritmo parte desde un conjunto de soluciones denominada población he iterativamente, lleva a cabo un proceso de reproducción, generando nuevas soluciones [11]. Este algoritmo pertenece a la categoría de algoritmos evolutivos y por lo tanto puede usar el mismo esquema presentado en la Figura 2.3.

Los individuos en el contexto del algoritmo genético son llamados cromosomas, los cuales como se menciono en la sección de los algoritmos evolutivos pueden ser representados de diversas maneras.

2.7.3. Conceptos para la optimización multiobjetivo

Como resultado de un proceso de optimización multiobjetivo no existe una única solución a un problema, sino que se tiene un conjunto de soluciones. Es por ello que a continuación se presentaran una serie de criterios que permitirán analizar y determinar el conjunto de soluciones optimas. Los criterios son los siguientes [25]:

Dominancia de Pareto

Sean u y v vectores pertenecientes a $\Re^n$, se dice que u domina a v (se denota como $u \preceq v$) si, y sólo si (en el caso de minimización): $\forall i \in \{1, 2, ..., n\} | f_i(u) \leq f_i(v) \wedge \exists j \in \{1, 2, ..., n\} | f_j(u) < f_j(v)$

Es decir, para que una solución domine a otra, cada uno de sus objetivos debe ser mejores o iguales y al menos en uno de ellos este debe ser mejor.

En el ejemplo de la Figura 2.4 se muestra los vectores A,B,C,D de los cuales C y B dominan a D, y A domina a C B y D. Nótese que C y B no son dominantes entre

sí.

Optimo de Pareto

Una solución u es un optimo de Pareto si no hay otra solución v en el espacio Ω, tal que v domine a u, es decir, para $u, v \in \Re^n \quad \nexists v \preceq u$. Los óptimos de pareto también se conocen bajo el nombre de solución no-dominada. En la Figura 2.4 el vector A sería un optimo de Pareto.

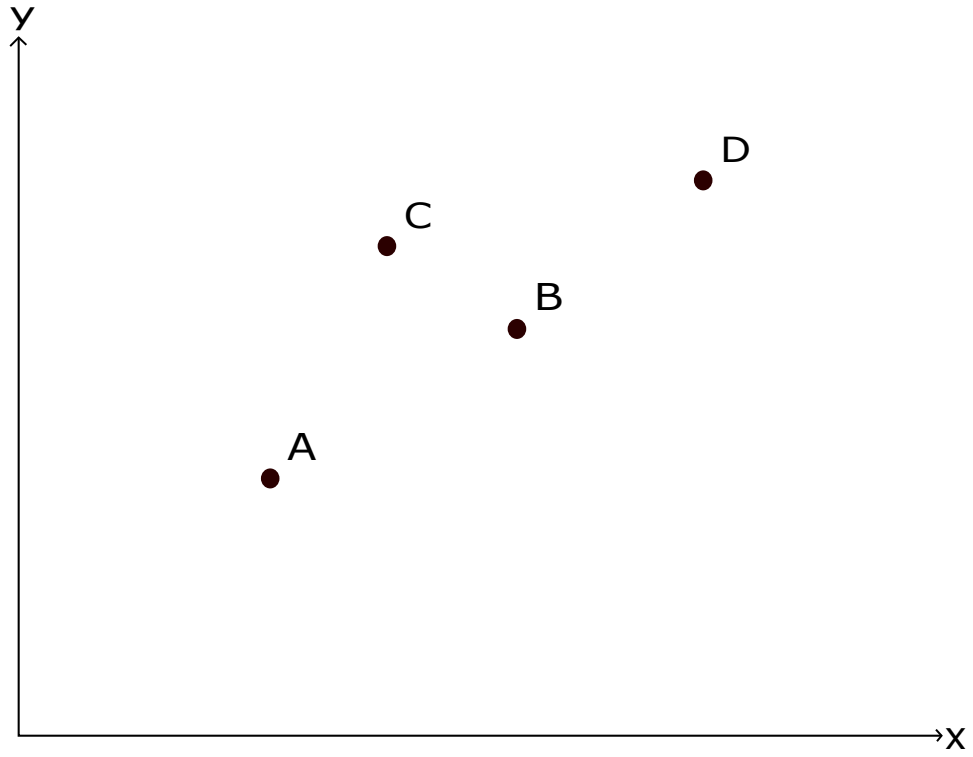

Figura 2.4: Ejemplo de dominancia y óptimo de Pareto

Frontera de Pareto

La frontera de Pareto es el conjunto de todas las soluciones no dominadas las cuales componen las soluciones óptimas al problema multiobjetivo. En la Figura 2.5 los puntos rojos componen la frontera de Pareto.

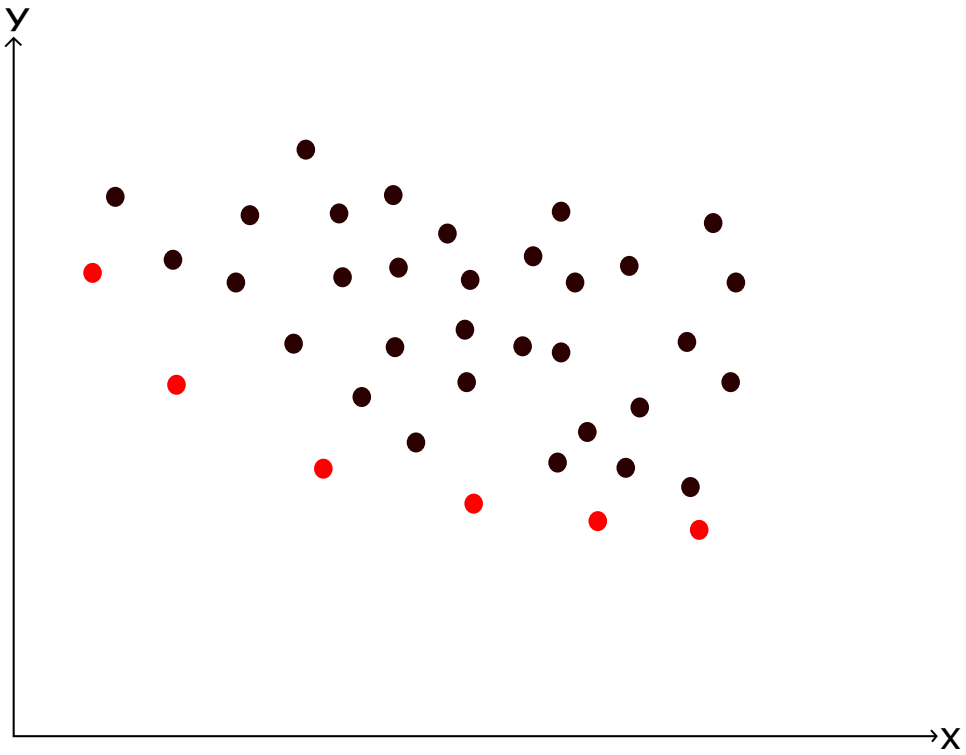

Figura 2.5: Ejemplo frente de Pareto

2.7.4. Algoritmo NSGA-II (Nondominated Genetic Algorithm)

El algoritmo NSGA-II [8] pertenece a la categoría de algoritmo evolutivo multi-objetivo (MOEA). Este algoritmo al igual que el algoritmo genético hace uso de los operadores de selección, cruzamiento y mutación para encontrar un conjunto de soluciones optimas a problemas que cuentan con más de un objetivo. Adicionalmente, NSGA-II añade conceptos y operadores adicionales los cuales permiten mejorar su rendimiento y la calidad de las soluciones obtenidas. NSGA-II puede ser implementado siguiendo los mismos pasos de el algoritmo evolutivo mostrados en la Figura 2.3, utilizando la función de remplazo mostrada en la Figura 2.6.

En la Figura 2.6 se puede ver que el proceso de remplazo dentro del algoritmo genético consiste en unir la población actual y la población descendiente (formada por los elementos resultantes del operador de selección y sobre la que se han aplicado el cruzamiento y la mutación) en un solo elemento llamada unionPoblación. Luego, esta es enviada a un procedimiento el cual ordena y categoriza la población en diversos frentes de acuerdo al concepto de dominancia de Pareto. Después, de que la población

a sido categorizada se procede a iterar sobre los frentes y añadir sus elementos a una nueva población con el cuidado de no sobrepasar el tamaño de la población deseada (N). En caso de que uno de los frentes no pueda ser añadido en su totalidad por sobrepasar dicho tamaño, se llevará a cabo un proceso por el cual se ordenaran las soluciones en dicho frente basadas en un criterio conocido como densidad de las soluciones. Una vez realizado el ordenamiento, se agregarán las mejores soluciones a la nueva población hasta alcanzar el tamaño deseado N. Se puede ver un ejemplo de este procedimiento gráficamente en la Figura 2.7.

Algorithm 2: Función de remplazo para el algoritmo NSGA-II

```
1  Function remplazar(población, poblaciónDescendiente)
2      unionPoblacion ← población ∪ poblaciónDescendiente
3      /* F = (F_1, F_2, ...)                                   */
4      F ← ordenarPorFrentesNoDominados (unionPoblacion)
5      nuevaPoblacion ← ∅
6      i = 1
7      /* Hasta que nuevaPoblacion este lleno                   */
8      while (|nuevaPoblacion| + |F_i|) ≤ N do
9          /* Calcular y asignar la densidad a cada solución del
               frente F_i                                       */
10         asignarDensidad (F_i)
11         /* Añadir a nuevaPoblacion las soluciones del frente F_i  */
12         nuevaPoblacion ← nuevaPoblacion ∪ F_i
13         i = i + 1
14     end
15     /* Ordenar el frente F_i usando el comparador de densidad  */
16     ordernar (F_i, ≺_n)
17     /* Elegir los primeros N − |nuevaPoblacion|]              */
18     nuevaPoblacion ← nuevaPoblacion ∪ F_i[1 : N − |nuevaPoblacion|]
19     retornar nuevaPoblacion
20 fin
```

Figura 2.6: Pseudocódigo de la función de remplazo utilizada en el algoritmo NSGA-II [8]

A continuación se procederá a explicar los operadores adicionales presentados en

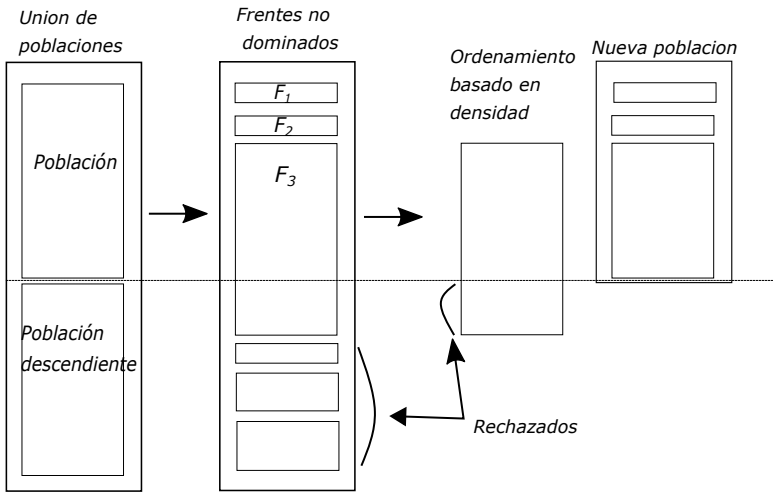

Figura 2.7: Procedimiento NSGA-II [8]

[8] y que son utilizados por la función de remplazo del algoritmo NSGA-II en la Figura 2.6.

Ordenamiento de soluciones en frentes no dominados

Uno de los procedimientos presentados en la Figura 2.6 consiste en ordenar las soluciones en frentes no dominados. Los frentes no dominados son conjuntos en los que se almacenan las soluciones que no se dominan entre si. En la Figura 2.8 se muestra un ejemplo con tres frentes.

El algoritmo para llevar a cabo esto se presenta en la Figura 2.9 y consiste en lo siguiente:

Primero, se debe comparar todas las soluciones entre si utilizando el concepto de dominancia de Pareto. Para ello, cada solución p cuenta con un atributo S_p en el que guarda todas las soluciones a las que domina y un contador n_p que almacena el numero de soluciones que lo dominan a él. Cada vez que se termina de comparar una solución con todas las otras, si su contador n_p es igual a 0, se le asigna a la solución el rango que indica el frente al que pertenece, y se guarda esta en un conjunto que contiene a todas las soluciones de dicho frente F_1.

Una vez que se han identificado todas las soluciones no dominadas del primer frente se procede a generar el siguiente, para lo cual, por cada solución q almacenada en el conjunto $S_p \in p$ del frente ya conocido se disminuye en uno su contador n_q. Si el contador n_q de la solución q llega a 0, entonces se le asigna a dicha solución el

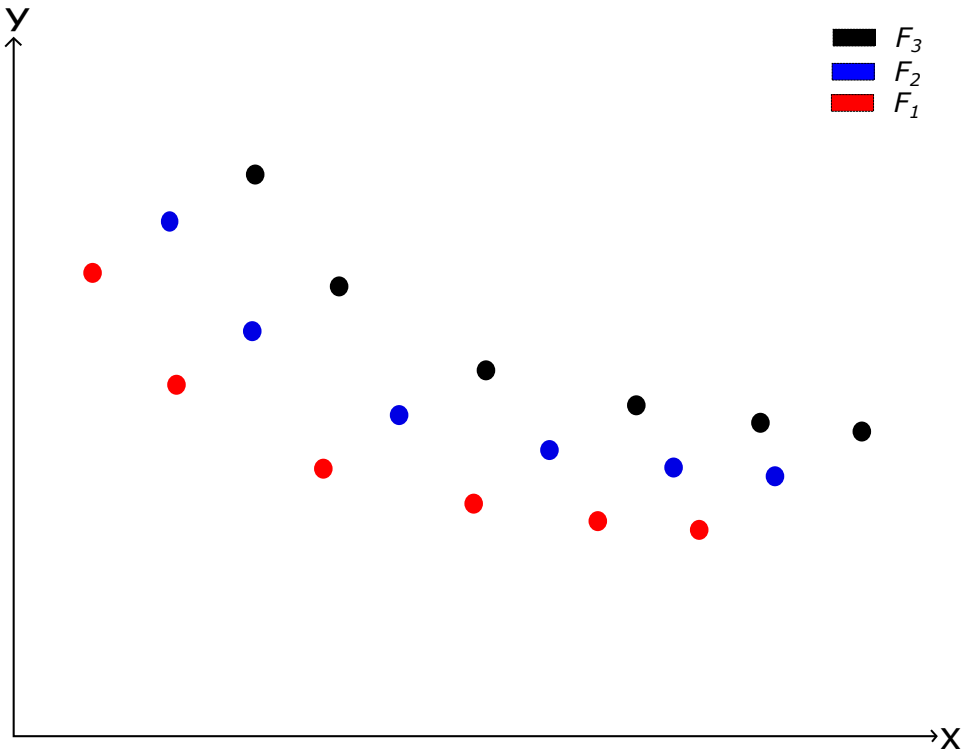

Figura 2.8: Frentes no dominados

rango correspondiente y se guarda esta en un conjunto temporal Q con el resto de las soluciones en dicho frente. Cuando se tengan identificadas todas las soluciones, estas se asignan al frente correspondiente F_i.

Finalmente, se repite el procedimiento anterior sobre el nuevo frente hasta haberlos generado todos.

Densidad de estimación (Crowing Distance)

Otra función presente en la Figura 2.6 es la asignación de las densidades sobre cada solución, la cual corresponde a la distancia promedio entre la solución anterior y la siguiente a partir de cada uno de los objetivos. En la Figura 2.10 la densidad de la solución i corresponde a la suma de la longitud del lado mayor y del lado menor del rectángulo.

El procedimiento para esta función se muestra en la Figura 2.10 y consiste en:

Primero, crear e inicializar un arreglo $\mathcal{I}_{distancia}$, con el valor 0, en donde guardar las distancias para cada solución a medida que se van calculando. Luego, por cada

uno de los objetivos se ordena el frente $\mathcal{I}$ y dentro $\mathcal{I}_{distancia}$ se le asigna al primer y ultimo elemento el valor infinito. Finalmente, se recorre desde la segunda hasta la penúltima solución calculando la distancia $\mathcal{I}[i]_{distancia}$. Notar que f_m^{max} y f_m^{min} corresponden al valor del objetivo m de la primera y ultima solución.

Comparador de densidad (Crowing Distance comparator)

Este operador compara las soluciones basados en dos conceptos los cuales son el rango de dominación y la densidad de soluciones. Estos fueron calculados al momento de generar los frentes y asignar la densidad a las soluciones. De acuerdo a Deb [8], se define el orden dado por el operador de densidad ($\prec_n$) como: $i \prec_n j$, si ($i_{rango} < j_{rango}$) o (($i_{rango} = j_{rango}$) o ($i_{distancia} > j_{distancia}$)).

Algorithm 3: Función de ordenación en frentes no dominados

```
1  Function ordenarPorFrentesNoDominados (población)
2      F ← ∅
3      foreach p ∈ población do
4          /* Conjunto de soluciones dominadas por p        */
5          S_p = ∅
6          /* Numeros de soluciones que dominan a p         */
7          n_p = 0
8          foreach q ∈ población do
9              if p ≺ q then /* p domina a q                 */
10                 S_p = S_p ∪ {q}
11             else if q ≺ p then /* q domina a p            */
12                 n_p = n_p + 1
13             end
14             if n_p = 0 then
15                 p_rango = 1
16                 F_1 = F_1 ∪ {p}
17             end
18         end
19     end
20     i = 1
21     while F_i ≠ 0 do
22         Q = ∅
23         foreach p ∈ F_i do
24             foreach q ∈ S_p do
25                 n_q = n_q − 1
26                 if n_q = 0 then
27                     q_rango = i + 1
28                     Q = Q ∪ {q}
29                 end
30             end
31         end
32         i = i + 1
33         F_i = Q
34     end
35 fin
```

Figura 2.9: Pseudocódigo de la función de ordenamiento utilizada en NSGA-II [8]

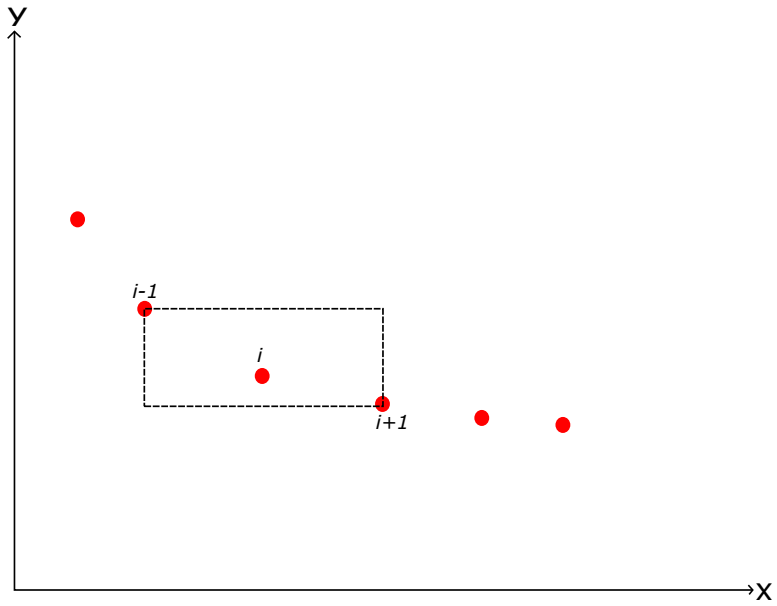

Figura 2.10: Calculo de la densidad de estimación al rededor de la solución i [8].

Algorithm 4: Función de calculo de densidades

```
1  Function asignarDensidad (ℐ) /* ℐ : frente de soluciones no
     dominadas                                                    */
2      l = |ℐ| /* Obtiene el tamaño del frente                    */
3
4      /* Inicializa la distancia para cada solución             */
5      foreach i ← 1 to l do
6          ℐ[i]_distancia ← 0
7      end
8      foreach m ← 1 to numero de objetivos do
9          ℐ ← sort(ℐ, m) /* Ordena por el objetivo m            */
10         /* Asignar a la primera y ultima solucion el valor ∞  */
11         ℐ[1]_distancia ← ℐ[l]_distancia ← ∞
12         for i ← 2 to l − 1 do
13             /* Asigna la distancia a las soluciones restantes */
14             ℐ[i]_distancia ← ℐ[i]_distancia + (ℐ[i+1].m − ℐ[i−1].m)/(f_m^max − f_m^min)
15         end
16     end
17 fin
```

Figura 2.11: Pseudocódigo de la función de asignación de densidad [8]

3. Metodología

Durante este capitulo se explicara la razón por la que fue escogida esta metodología sobre todas las demás existentes y como esta sera adaptada para ser llevada a cabo durante el desarrollo del proyecto.

Debido a que la metodología esta pensada para ser llevada a cabo por un equipo de trabajo se adaptará la metodología para poder ser aplicada en el desarrollo llevado a cabo por una sola persona. Esta adaptación consiste en la disminución de la cantidad de la documentación generada, permitir llevar a cabo más de una fase de la iteración al mismo tiempo y los roles de analista, diseñador, implementador y tester sera realizado por una sola persona. Los documentos a generar por cada fase serán:

Análisis: El producto generado por esta fase sera un documento de especificación de requisitos que constara de:

- Introducción: En este apartado del documento se hará una introducción al problema que se ha identificado.
- Requisitos de usuario: Consiste en la recopilación de lo requisitos de los usuarios que deben ser cumplidos al final del periodo de desarrollo.
- Requisito de sistema: Son los requisitos, desde un punto de vista mas técnico, que son necesarios para satisfacer los requisitos de usuario.
- Matriz de trazado requisitos de usuario vs sistema: Matriz que permite ver la trazabilidad de los requisitos de usuario con los de sistema.

Diseño: Esta fase generara como producto un documento de diseño que contara con los siguientes diagramas:

- Casos de uso: Serie de diagramas que permiten ver la interacción que el usuario tendrá con el sistema.
- Arquitectura lógica: Descripción a alto nivel del software y los componentes que lo componen.
- Diagrama de componentes: Permite ver la división del sistema y la interacción entre los distintos componentes [3].
- Diagrama de clases: Describe la relación entre las distintas clases presentes en la solución propuesta.

Implementación: Esta fase generara el código fuente de la aplicación y un manual de usuario de la aplicación.

Pruebas: Durante esta fase de realizaran la documentación y la realización de las pruebas sobre la aplicación.

- Se documentará las pruebas unitarias realizadas y sobre que componentes.
- Se documentará las pruebas de integración y que caso de uso cubren.

La razón por la que se utilizará esta metodología sobre otras es porque el producto resultante de este proyecto esta pensado para servir como base para futuros trabajos. Debido a esto es necesario documentar correctamente para que otros programadores puedan continuar con su desarrollo más adelante. Aunque existen otras metodologías como cascada u otras tradicionales, estas son difíciles de llevar a cabo por la cantidad de documentación que se requiere, mientras que metodologías de desarrollo ágil carecen en cuanto a la documentación que se necesita para el sistema a desarrollar. Además, esta metodología nos permite obtener una retroalimentación al final de la iteración, obtener nuevos requisitos que no hayan quedado definidos en etapas anteriores o refinar los requisitos y el diseño ya existente, permitiendo así mejorar la calidad del producto final.

La implementación de esta metodología para el desarrollo del proyecto se llevará a cabo repartiendo las tareas necesarias para el cumplimiento de los objetivos en

iteraciones. De este modo al final de cada iteración se contará con un prototipo funcional de la aplicación sobre el que se agregaran las nuevas funcionalidades en las iteraciones siguientes.

4. Evaluación De La Solución

5. Conclusiones Y Trabajos Futuros

Glosario

El primer término: Este es el significado del primer término, realmente no se bien lo que significa pero podría haberlo averiguado si hubiese tenido un poco mas de tiempo.

El segundo término: Este si se lo que significa pero me da lata escribirlo...

Bibliografía

[1] Adel Alshamrani and Abdullah Bahattab. A comparison between three SDLC models waterfall model, spiral model, and incremental/iterative model. *IJCSI International Journal of Computer Science Issues*, 12(1):106–111, 2015.

[2] H. A. Basha and B. G. Kassab. Analysis of water distribution systems using a perturbation method. *Applied Mathematical Modelling*, 20(4):290–297, 1996.

[3] Donald Bell. UML basics : The component diagram. *IBM developerWorks*, (December):1–10, 2004.

[4] Christian Blum and Andrea Roli. Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison. *ACM Computing Surveys*, 35(3):268–308, 2003.

[5] Ilhem Boussaïd, Julien Lepagnot, and Patrick Siarry. A survey on optimization metaheuristics. *Information Sciences*, 237:82–117, 2013.

[6] Omid Bozorg-Haddad, Mohammad Solgi, and Hugo A Loáiciga. *Meta-heuristic and evolutionary algorithms for engineering optimization*. John Wiley & Sons, Incorporated, Newark, United States, 2017.

[7] Mathias Braux and Jacques Noyé. Towards partially evaluating reflection in Java. *ACM SIGPLAN Notices*, 34(11):2–11, 1999.

[8] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 2002.

[9] Pino V. Edwin, Valle C. Angely, Condori P. Franz, Mejia M. Jesus, Chavarri V. Eduardo, and Alfaro R. Luis. Diseño óptimo de redes de distribución de agua usando un software basado en microalgoritmos genéticos multiobjetivos. *Ribagua*, 4(1):6–23, 2017.

[10] James Gosling, Bill Joy, Guy Steele, Gilad Bracha, and Alex Buckley. The Java® Language Specification Java SE 8 Edition, 2015.

[11] Dorothea Heiss-Czedik. An introduction to genetic algorithms. *Artificial Life*, 3(1):63–65, 1997.

[12] Sean Luke. *Essentials of metaheuristics*. Lulu, second edition, 2013.

[13] Robert C Martin. Iterative and incremental development (IID). *Design*, (Iid), 1999.

[14] Darshan Mehta, Vipin Yadav, Keyur Prajapati, and Sahita Waikhom. Design of optimal water distribution systems using WaterGEMS: A case study of Surat city. *E-proceedings of the 37th IAHR World Congress*, (December):1–8, 2017.

[15] Susan M. Mitchell and Carolyn B. Seaman. A comparison of software cost, duration, and quality for waterfall vs. iterative and incremental development: A systematic review. *2009 3rd International Symposium on Empirical Software Engineering and Measurement, ESEM 2009*, (February 2008):511–515, 2009.

[16] Daniel Mora. Diseño de redes de distribución de agua mediante algoritmos evolutivos. análisis de eficiencia. 2012.

[17] Gabriel Pereyra, Daniel Pandolfi, and Andrea Villagra. Diseño y optimización de redes de distribución de agua utilizando algoritmos genéticos. *Informes Científicos Técnicos - UNPA*, 9(1):37–63, 2017.

[18] Roger S Pressman. *Software Engineering A Practitioner's Approach*. 7 edition, 2009.

[19] Marc H.J. Romanycia and Francis Jeffry Pelletier. What is a heuristic? *Computational Intelligence*, 1(1):47–58, 1985.

[20] Lewis Rossman. *EPANET 2.0 en español.Analisis hidraulico y de calidad del agua en redes de distribución de agua. manual del usuario.* 2017.

[21] Lewis A. Rossman. The EPANET Programmer's Toolkit for Analysis of Water Distribution Systems. In *WRPMD'99*, pages 1–10, Reston, VA, jun 1999. American Society of Civil Engineers.

[22] Per Runeson and Martin Höst. Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering*, 14(2):131–164, 2009.

[23] C L Sabharwal. Java, Java, Java. *IEEE Potentials*, 17(3):33–37, 1998.

[24] Yamisleydi Salgueiro, Jimmy Gutiérrez-Bahamondes, Yamisleydi Salgueiro, Sergio Silva-Rubio, Marco Alsina, Daniel Mora-Meliá, and Vicente Fuertes-Miquel. jHawanet: An Open-Source Project for the Implementation and Assessment of Multi-Objective Evolutionary Algorithms on Water Distribution Networks. *Water*, (September), 2019.

[25] David a. Van Veldhuizen and Gary B Lamont. Evolutionary Computation and Convergence to a Pareto Front. *Late Breaking Papers at the Genetic Programming 1998 Conference*, pages 221–228, 1998.

[26] R Victor. Iterative and incremental development: A brief history. 2003.

[27] Xin She Yang. Metaheuristic optimization. *Scholarpedia*, 6(2011):11472, 2015.

ANEXOS

A. Documento de especificación de requisitos

B. Documento de diseño

C. Documento de casos de prueba

D. Cuestionario para la evaluación de la aplicación