Documento de Requisitos

de Usuario / Software

Herramienta para la Optimización de Redes de Distribución de Agua Potable.

Fecha: 21/09/2019

Versión: 0.1

Equipo de Desarrollo:

Nombre	Rol	Contacto
Gabriel Sanhueza Fuentes	Administrador, Analista, Diseñador, Implementador y Tester.	gsanhueza15@alumnos.utalca.cl

Contraparte:

Nombre	Rol	Contacto
Jimmy H. Gutiérrez-Bahamondes	Cliente/Profesor guía	
Daniel Mora-Meliá	Cliente/Profesor co-guía	

Historia del Documento

Versión	Fecha	Razón del Cambio	Autor(es)
0.1	07/09/2019	Primer borrador	

Índice

1 Introducción

El proyecto que se desarrollara consiste en la creación de una herramienta que haga uso de algoritmos metaheurísticos para tratar y minimizar problemas existentes en redes de distribución de agua potable.

Este proyecto solo abarcara dos problemas existentes dentro de las redes de distribución de agua potable.

Para el desarrollo de este proyecto se usarán librerías ya existentes con el fin de reducir el tiempo de desarrollo. Estas librerías son epanet2.dll, desarrollada en c, y JNA, librería en java para funciones nativas.

1.1 Propósito del Sistema

El proyecto consiste en el desarrollo de un sistema que permita simular y buscar soluciones a problemas presentes en las redes de distribución de agua potable haciendo uso de algoritmos metaheurísticos. Adicionalmente, el sistema será diseñado de tal forma que pueda ser extendido añadiendo nuevos problemas, algoritmos u operadores.

1.2 Alcance del Proyecto

Al final del periodo de desarrollo la herramienta contara con las siguientes prestaciones.

- El sistema permitirá la carga y la visualización de la red gráficamente.
- El sistema solo resolverá dos clases de problemas de optimización, uno mono-objetivo y el otro multiobjetivo. El problema mono-objetivo 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 mono-objetivo, mientras que NSGA-II será aplicado al multiobjetivo.
- El sistema permitirá visualizar y guardar las soluciones de los algoritmos en un archivo.
- El sistema permitirá que el usuario agregue nuevos algoritmos, operadores o problemas sin tener que modificar la interfaz de usuario.

Este proyecto no contempla la creación de la red por lo que estas deberán ser ingresadas como entradas al programa. Además, esta herramienta únicamente podrá ser ocupada en equipos cuyo sistema operativo sea Windows debido a que se realizan llamadas a librerías nativas.

1.3 Contexto

Este sistema será desarrollado utilizando el lenguaje de programación java. Debido a que este es un lenguaje ampliamente utilizado y que cuenta con un gran soporte y comunidad que lo utilizan.

Como motor de cálculo para llevar a cabo las simulaciones se utilizará una librería desarrollada en c, que cuenta con funciones para llevar a cabo simulaciones de redes de agua potable. El nombre de esta librería es epanet2.dll. Las funciones que incorpora esta librería se encuentran explicadas en [1]

Desde lenguaje se realizarán llamadas a librerías nativas usando la librería JNA existente en java. Esta librería cuenta con las clases y métodos necesarios para poder acoplar este sistema a la librería epanet2.dll desarrollada en c y que será usada como motor de calculo para llevar a cabo las simulaciones.

Puesto que una de las funcionalidades del sistema es permitir la ejecución de algoritmos metaheurísticos, se toma como base la arquitectura presentada por el framework JMetal.

JMetal es un framework para la optimización multiobjetivo con metaheurísticas. Su arquitectura inicial [2] involucraba una serie de problemas y dificultaban la realización de ciertas acciones que eran recurrentes. Además, esta no hacia uso de las novedades incorporadas por Java como los genéricos. Es por esto, que posteriormente fue rediseñada, haciendo uso de patrones de diseño, principios de la programación orientada a objetos y aprovechando las características del lenguaje Java. Este rediseño se presenta en [3].

El contexto en el que se desenvolverá este sistema será en ambientes universitario, de investigación y en el ambiente laboral.

1.4 Definiciones, Acrónimos y Abreviaturas

NSGA-II: Non-dominated Sorting Genetic Algorithm

1.5 Referencias

[1] L. Rossman, *EPANET 2.0 en Español. Analisis Hidraulico y de Calidad del Agua en Redes de Distribución de Agua. Manual del Usuario.* 2017.

[2] J. J. Durillo, A. J. Nebro, and E. Alba, “The jMetal framework for multi-objective optimization: Design and architecture,” *2010 IEEE World Congr. Comput. Intell. WCCI 2010 - 2010 IEEE Congr. Evol. Comput. CEC 2010*, pp. 1–8, 2010.

[3] A. J. Nebro, J. J. Durillo, and M. Vergne, “Redesigning the jMetal multi-objective optimization framework,” in *GECCO 2015 - Companion Publication of the 2015 Genetic and Evolutionary Computation Conference*, 2015.

2 Descripción General

En esta sección se describirán las características de los usuarios que harán uso del sistema. Además, se mencionará el ambiente operacional de la solución y la relación que este proyecto tiene con otros proyectos. Finalmente, también se mencionará las restricciones generales que existe en la realización de esta herramienta.

2.1 Características de los Usuarios

Este sistema solo cuenta con un tipo de usuario el cual tendrá acceso a todas las funcionalidades. Se espera que el usuario que este sistema sean ingenieros, investigadores u personas cuenten con el conocimiento básico acerca de redes de distribución de agua potable.

2.2 Ambiente Operacional de la Solución

El ambiente operacional en que el sistema desarrollado es el siguiente:

- Intel(R) Core(TM) i7-7700HQ CPU @ 2.80Ghz 2.8Ghz
- RAM 16GB DDR4
- HDD 7200rpm 1T
- SSD 256GB PCIe NVME M.2
- Windows 10 x64
- NVIDIA GeForce GTX 1050

2.3 Relación con Otros Proyectos

El sistema depende de la librería nativa epanet2_64bit.dll, ya que usara esta librería como motor de cálculo. La librería cuenta con 54 funciones dentro de las cuales se encuentran funciones para correr las simulaciones, modificar y obtener configuraciones de la red, modificar los elementos que conforman la red y generar reportes.

Las llamadas desde java a la librería nativa serán realizadas a través de la librería JNA.

Adicionalmente, el sistema toma la arquitectura utilizada por JMETAL como base para agregar los algoritmos, operadores y problemas. Esta arquitectura será modificada según se necesite para satisfacer los requisitos del sistema.

2.4 Restricciones Generales

- La red será ingresada como entrada al programa a través de un archivo .inp.
- La herramienta solo estará disponible para el sistema operativo Window.

3 Requisitos del Sistema

En esta sección se presentan los requisitos que pueden están sujetos a posibles cambios en las siguientes iteraciones.

3.1 Requisitos de Usuario

A continuación, se presentarán los requisitos de usuarios que han sido obtenidos para el desarrollo de este proyecto

RU001 – Cargar una red		
Descripción	**:**	La red que es representada por el archivo .inp debe ser cargada en el programa para poder llevar a cabo la simulación.
Fuente	**:**	Jimmy Gutiérrez
Prioridad	**:**	
Estabilidad	**:**	Intransable
Fecha actualización :		09/09/2019
Estado	**:**	Cumple
Incremento	**:**	1
Tipo	**:**	Funcional

RU002 – Aplicar algoritmo genético al problema mono objetivo		
Descripción	**:**	El algoritmo genético debe ser aplicado para resolver el problema mono objetivo que tiene como función objetivo el costo de inversión y como variable de decisión el diámetro de las tuberías.
Fuente	**:**	Jimmy Gutiérrez
Prioridad	**:**	Critica
Estabilidad	**:**	Intransable
Fecha actualización :		09/09/2019
Estado	**:**	Cumple
Incremento	**:**	2
Tipo	**:**	Funcional

RU003 – Aplicar algoritmo NSGA-II al problema multiobjetivo		
Descripción	**:**	El algoritmo NSGA-II debe ser aplicado al problema multiobjetivo cuyas funciones a optimizar son los costos energéticos y el número de encendido y apagado de las bombas (Pumping Schedule)
Fuente	**:**	Jimmy Gutiérrez
Prioridad	**:**	Critica
Estabilidad	**:**	Intransable
Fecha actualización :		09/09/2019
Estado	**:**	No_Cumple
Incremento	**:**	3
Tipo	**:**	Funcional

RU004 – Agregar operadores para usar con el algoritmo NSGA-II		
Descripción	**:**	Se deben agregar distintos operadores de cruza y de mutación para aplicarlos en el algoritmo genético
Fuente	**:**	Jimmy Gutiérrez
Prioridad	**:**	
Estabilidad	**:**	Intransable
Fecha actualización	**:**	09/09/2019
Estado	**:**	No_Cumple
Incremento	**:**	2
Tipo	**:**	Funcional

RU005 – Agregar la funcionalidad de la DLL Epanet		
Descripción	**:**	Epanet DLL es una librería programada en C, la cual debe ser usada para realizar las simulaciones de las redes de distribución de agua potable en este proyecto.
Fuente	**:**	Jimmy Gutiérrez
Prioridad	**:**	
Estabilidad	**:**	Intransable
Fecha actualización	**:**	09/09/2019
Estado	**:**	No_Cumple
Incremento	**:**	3
Tipo	**:**	Funcional

RU006 – Visualizar red en una interfaz grafica		
Descripción	**:**	La red cargada desde un archivo .inp ser mostrada en la interfaz gráfica de la aplicación.
Fuente	**:**	Jimmy Gutiérrez
Prioridad	**:**	
Estabilidad	**:**	
Fecha actualización	**:**	09/09/2019
Estado	**:**	No_Cumple
Incremento	**:**	4
Tipo	**:**	Funcional

RU007 – Almacenar los resultados de los algoritmos aplicados		
Descripción	**:**	Los resultados generados por la ejecución de los algoritmos deben poder ser guardados en un archivo local.
Fuente	**:**	Jimmy Gutiérrez
Prioridad	**:**	
Estabilidad	**:**	
Fecha actualización	**:**	09/09/2019

Estado	**:**	No_Cumple
Incremento	**:**	4
Tipo	**:**	Funcional

RU008 – Implementar el operador IntegerSBXCrossover		
Descripción	**:**	El operador IntegerSBXCrossover es uno de los operadores de cruzamiento. En base a cálculos probabilísticos combina dos soluciones para crear unas dos nuevas soluciones hijas.
Fuente	**:**	Jimmy Gutiérrez
Prioridad	**:**	
Estabilidad	**:**	
Fecha actualización	**:**	01/10/2019
Estado	**:**	Cumple
Incremento	**:**	2
Tipo	**:**	Funcional

RU009 – Implementar el operador IntegerSinglePointCrossover		
Descripción	**:**	El operador IntegerSinglePointCrossover es un operador de cruzamiento. Viendo la solución como un vector, este operador toma dos soluciones y elige un punto a partir del cual los valores de una solución se intercambiarán con los valores de otra solución. Este operador usa una probabilidad de cruzamiento y solamente realiza el intercambio de los valores en la solución cuando un numero generado aleatoriamente es menor que la probabilidad de cruzamiento.
Fuente	**:**	Jimmy Gutiérrez
Prioridad	**:**	
Estabilidad	**:**	
Fecha actualización	**:**	01/10/2019
Estado	**:**	Cumple
Incremento	**:**	2
Tipo	**:**	Funcional

RU010 – Implementar el operador IntegerPolynomialMutation		
Descripción	**:**	El operador IntegerPolynomialMutation es un operador de mutación. Este operador de mutación usa cálculos probabilísticos para mutar algunos variables de decisión que forman parte de la solución.
Fuente	**:**	Jimmy Gutiérrez
Prioridad	**:**	
Estabilidad	**:**	
Fecha actualización	**:**	01/10/2019

Estado	**:**	Cumple
Incremento	**:**	2
Tipo	**:**	Funcional

RU011 – Implementar el operador IntegerSimpleRandomMutation		
Descripción	**:**	El operador IntegerSimpleRandomMutation es un operador de mutación. Este operador muta una variable de decisión cuando un numero generado aleatoriamente es menor que la probabilidad de mutación establecida. El operador recorre cada variable de decisión realizando lo descrito anteriormente. La mutación realizada por este operador consiste en cambiar el valor de la variable de decisión por otro valor aleatorio.
Fuente	**:**	Jimmy Gutiérrez
Prioridad	**:**	
Estabilidad	**:**	
Fecha actualización	**:**	01/10/2019
Estado	**:**	Cumple
Incremento	**:**	2
Tipo	**:**	Funcional

RU012 – Implementar el operador IntegerRangeRandomMutation (Asi lo llame)		
Descripción	**:**	El operador IntegerRangeRandomMutation es un operador de mutación. Este operador muta una variable de decisión cuando un numero generado aleatoriamente es menor que la probabilidad de mutación establecida. El operador recorre cada variable de decisión realizando lo descrito anteriormente. La mutación realizada por este operador consiste en cambiar el valor de la variable de decisión por otro valor aleatorio que se encuentre entre un rango establecido. Ejemplo: Variable de decisión: 3 Rango: 2 La variable de decisión después de aplicado el operador puede tomar un valor entre [1, 5].
Fuente	**:**	Jimmy Gutiérrez
Prioridad	**:**	
Estabilidad	**:**	
Fecha actualización	**:**	01/10/2019
Estado	**:**	Cumple
Incremento	**:**	2
Tipo	**:**	Funcional

RU013 – Implementar el operador UniformSelection		
Descripción	**:**	El operador UniformSelection es un operador de selección. Este operador de selección ordena la población y asigna una probabilidad máxima y mínima a la mejor y peor solución respectivamente. A las soluciones que se encuentran entre la mejor y la peor solución se le asigna una probabilidad que se encuentra entre la probabilidad máxima y mínima. Si la probabilidad de la solución es mayor a 1.5 entonces la solución se duplica en la nueva población. Si la probabilidad esta entre 0.5 y 1.5, entonces en la nueva población se agrega la solución solo una vez. Las soluciones cuya probabilidad es menor que 0.5 no aparecen en la nueva población.
Fuente	**:**	Jimmy Gutiérrez
Prioridad	**:**	
Estabilidad	**:**	
Fecha actualización	**:**	01/10/2019
Estado	**:**	Cumple
Incremento	**:**	2
Tipo	**:**	Funcional

RU014 – Crear archivo .inp de la solución generada		
Descripción	**:**	Al ejecutar un algoritmo metaheurístico este devuelve una solución al problema, a partir de esta solución se debe crear y guardar un archivo .inp.
Fuente	**:**	Jimmy Gutiérrez
Prioridad	**:**	
Estabilidad	**:**	
Fecha actualización	**:**	15/10/2019
Estado	**:**	No_Cumple
Incremento	**:**	4
Tipo	**:**	Funcional

RU015 – Mostrar las soluciones y poder guardarlas		
Descripción	**:**	Mostrar los resultados de la ejecución del algoritmo, es decir las variables y los valores objetivos resultantes. Adicionalmente, estas deben poder ser guardadas en el equipo. Las variables de decisión se guardarán en un archivo (VAR), mientras que los valores de los objetivos se guardaran en otro (FUN).
Fuente	**:**	Jimmy Gutiérrez
Prioridad	**:**	
Estabilidad	**:**	Transable
Fecha actualización	**:**	30/11/2019
Estado	**:**	No_Cumple
Incremento	**:**	4

Tipo	**:**	Funcional

RU016 – Mostrar las características de la red		
Descripción	**:**	Mostrar las características que posee la red. Esto puede ser realizado cuando se presiona el elemento de la red o agregando algún componente que muestre los elementos que conforman la red.
Fuente	**:**	Jimmy Gutiérrez
Prioridad	**:**	
Estabilidad	**:**	Transable
Fecha actualización	**:**	30/11/2019
Estado	**:**	No_Cumple
Incremento	**:**	4
Tipo	**:**	Funcional

RU017 – Graficar las soluciones		
Descripción	**:**	Mostrar en un plano cartesiano las soluciones que se van obteniendo a medida que se ejecuta el algoritmo. Solo considerar hasta 2 objetivos.
Fuente	**:**	Jimmy Gutiérrez
Prioridad	**:**	
Estabilidad	**:**	Transable
Fecha actualización	**:**	30/11/2019
Estado	**:**	No_Cumple
Incremento	**:**	4
Tipo	**:**	Funcional

RU018 – Hacer el programa fácil de ampliar		
Descripción	**:**	El programa debe poder fácilmente agregar nuevos algoritmos, operadores y problemas.
Fuente	**:**	Jimmy Gutiérrez
Prioridad	**:**	
Estabilidad	**:**	Transable
Fecha actualización	**:**	30/11/2019
Estado	**:**	No_Cumple
Incremento	**:**	
Tipo	**:**	No Funcional

3.2 Requisitos de Sistema

En esta sección se presentarán los requisitos de sistema obtenido a partir de los requisitos de usuarios.

RS001 – Leer red .inp	
Descripción	**:** Leer un archivo inp desde la aplicación.
Fuente	**:** Jimmy Gutiérrez
Prioridad	**:**
Estabilidad	**:**
Fecha actualización	**:** 09/09/2019 – 12:14
Estado	**:** No_Cumple
Incremento	**:** 1
Tipo	**:** Funcional

RS002 – Cargar red dentro del programa	
Descripción	**:** Generar una representación de la red en el programa a partir del archivo leído.
Fuente	**:** Jimmy Gutiérrez
Prioridad	**:**
Estabilidad	**:**
Fecha actualización	**:** 09/09/2019 – 12:14
Estado	**:** No_Cumple
Incremento	**:** 1
Tipo	**:** Funcional

RS003 – Implementar algoritmo genético	
Descripción	**:** Implementar el algoritmo genético. La versión del algoritmo genético a ser implementado consiste en el algoritmo genético generacional.
Fuente	**:** Jimmy Gutiérrez
Prioridad	**:**
Estabilidad	**:**
Fecha actualización	**:** 09/09/2019 – 12:14
Estado	**:** No_Cumple
Incremento	**:** 2
Tipo	**:** Funcional

RS004 – Implementar el operador IntegerSBXCrossover	
Descripción	**:** El operador IntegerSBXCrossover es uno de los operadores de cruzamiento. En base a cálculos probabilísticos combina dos soluciones para crear

unas dos nuevas soluciones hijas.		
Fuente	**:**	Jimmy Gutiérrez
Prioridad	**:**	
Estabilidad	**:**	
Fecha actualización	**:**	01/10/2019 – 15:00
Estado	**:**	Cumple
Incremento	**:**	2
Tipo	**:**	Funcional

RS005 – Implementar el operador IntegerSinglePointCrossover		
Descripción	**:**	El operador IntegerSinglePointCrossover es un operador de cruzamiento. Viendo la solución como un vector, este operador toma dos soluciones y elige un punto a partir del cual los valores de una solución se intercambiarán con los valores de otra solución. Este operador usa una probabilidad de cruzamiento y solamente realiza el intercambio de los valores en la solución cuando un numero generado aleatoriamente es menor que la probabilidad de cruzamiento.
Fuente	**:**	Jimmy Gutiérrez
Prioridad	**:**	
Estabilidad	**:**	
Fecha actualización	**:**	01/10/2019 – 15:00
Estado	**:**	Cumple
Incremento	**:**	2
Tipo	**:**	Funcional

RS006 – Implementar el operador IntegerPolynomialMutation		
Descripción	**:**	El operador IntegerPolynomialMutation es un operador de mutación. Este operador de mutación usa cálculos probabilísticos para mutar algunos variables de decisión que forman parte de la solución.
Fuente	**:**	Jimmy Gutiérrez
Prioridad	**:**	
Estabilidad	**:**	
Fecha actualización	**:**	01/10/2019 – 15:00
Estado	**:**	Cumple
Incremento	**:**	2
Tipo	**:**	Funcional

RS007 – Implementar el operador IntegerSimpleRandomMutation		
Descripción	**:**	El operador IntegerSimpleRandomMutation es un operador de mutación. Este operador muta una variable de decisión cuando un numero generado aleatoriamente es menor que la probabilidad de mutación establecida. El operador

recorre cada variable de decisión realizando lo descrito anteriormente. La mutación realizada por este operador consiste en cambiar el valor de la variable de decisión por otro valor aleatorio.

Fuente	**:**	Jimmy Gutiérrez
Prioridad	**:**	
Estabilidad	**:**	
Fecha actualización	**:**	01/10/2019 – 15:00
Estado	**:**	Cumple
Incremento	**:**	2
Tipo	**:**	Funcional

RS008 – Implementar el operador IntegerRangeRandomMutation (Asi lo llame)
Descripción **:** El operador IntegerRangeRandomMutation es un operador de mutación. Este operador muta una variable de decisión cuando un numero generado aleatoriamente es menor que la probabilidad de mutación establecida. El operador recorre cada variable de decisión realizando lo descrito anteriormente. La mutación realizada por este operador consiste en cambiar el valor de la variable de decisión por otro valor aleatorio que se encuentre entre un rango establecido. Ejemplo: Variable de decisión: 3 Rango: 2 La variable de decisión después de aplicado el operador puede tomar un valor entre [1, 5].

Fuente	**:**	Jimmy Gutiérrez
Prioridad	**:**	
Estabilidad	**:**	
Fecha actualización	**:**	01/10/2019 – 15:00
Estado	**:**	Cumple
Incremento	**:**	2
Tipo	**:**	Funcional

RS009 – Implementar el operador UniformSelection
Descripción **:** El operador UniformSelection es un operador de selección. Este operador de selección ordena la población y asigna una probabilidad máxima y mínima a la mejor y peor solución respectivamente. A las soluciones que se encuentran entre la mejor y la peor solución se le asigna una probabilidad que se encuentra entre la probabilidad máxima y mínima. Si la probabilidad de la solución es mayor a 1.5 entonces la solución se duplica en la nueva población. Si la probabilidad

esta entre 0.5 y 1.5, entonces en la nueva población se agrega la solución solo una vez. Las soluciones cuya probabilidad es menor que 0.5 no aparecen en la nueva población.
Fuente : Jimmy Gutiérrez
Prioridad :
Estabilidad :
Fecha actualización : 01/10/2019 – 15:00
Estado : Cumple
Incremento : 2
Tipo : Funcional

RS010 – Evaluar soluciones al problema monoobjetivo usando Epanet
Descripción : Las soluciones generadas por el algoritmo genético deben ser evaluadas para ver si cumple con la presión mínima en cada nodo.
Fuente : Jimmy Gutiérrez
Prioridad :
Estabilidad :
Fecha actualización : 01/10/2019 – 15:00
Estado : No_Cumple
Incremento : 2
Tipo : Funcional

RS011 – Implementar NSGA-II
Descripción : Implementar el algoritmo NSGA-II. El cual es usado para tratar con problemas multiobjetivo.
Fuente : Jimmy Gutiérrez
Prioridad :
Estabilidad :
Fecha actualización : 01/10/2019 – 15:00
Estado : No_Cumple
Incremento : 3
Tipo : Funcional

RS012 – Evaluar soluciones al problema multiobjetivo usando Epanet.
Descripción : Los resultados generados por el algoritmo deben ser evaluados para medir su factibilidad.
Fuente : Jimmy Gutiérrez

Prioridad	**:**	
Estabilidad	**:**	
Fecha actualización	**:**	01/10/2019 – 15:00
Estado	**:**	No_Cumple
Incremento	**:**	3
Tipo	**:**	Funcional

RS013 – Modificar una red de acuerdo con los resultados		
Descripción	**:**	Cuando el algoritmo haya generado soluciones para el problema, hay que guardar estas soluciones en algún objeto para posteriormente volcarlas en un archivo inp.
Fuente	**:**	Implementador
Prioridad	**:**	
Estabilidad	**:**	
Fecha actualización	**:**	15/10/2019
Estado	**:**	No_Cumple
Incremento	**:**	4
Tipo	**:**	Funcional

RS014 – Crear archivo .inp de la solución generada		
Descripción	**:**	Al ejecutar un algoritmo metaheurístico este devuelve una solución al problema, a partir de esta solución se debe crear y guardar un archivo .inp.
Fuente	**:**	Jimmy Gutiérrez
Prioridad	**:**	
Estabilidad	**:**	
Fecha actualización	**:**	15/10/2019
Estado	**:**	No_Cumple
Incremento	**:**	4
Tipo	**:**	Funcional

RS015 – Abrir una ventana que permita visualizar los resultados		
Descripción	**:**	Cuando el algoritmo haya terminado de ejecutar se debe mostrar los resultados en una nueva ventana.
Fuente	**:**	Jimmy Gutiérrez
Prioridad	**:**	
Estabilidad	**:**	Transable
Fecha actualización	**:**	30/11/2019
Estado	**:**	No_Cumple

Incremento	**:**	4
Tipo	**:**	Funcional

RS016 – Mostrar los resultados		
Descripción	**:**	En la ventana abierta se deben mostrar los resultados.
Fuente	**:**	Jimmy Gutiérrez
Prioridad	**:**	
Estabilidad	**:**	Transable
Fecha actualización	**:**	30/11/2019
Estado	**:**	No_Cumple
Incremento	**:**	4
Tipo	**:**	Funcional

RS017 – Guardar las soluciones mostradas		
Descripción	**:**	Se deben poder guardar las soluciones mostradas en la ventana de presentación de resultados, la cuales corresponden a las soluciones entregadas por el algoritmo ejecutado para un problema en particular. Para guardar las soluciones se generarán dos archivos. Un archivo VAR el cual contiene los valores de la variable de decisión. Y un archivo FUN que contienen los valores de los objetivos.
Fuente	**:**	Jimmy Gutiérrez
Prioridad	**:**	
Estabilidad	**:**	Transable
Fecha actualización	**:**	30/11/2019
Estado	**:**	No_Cumple
Incremento	**:**	4
Tipo	**:**	Funcional

RS018 – Implementar un componente que permita mostrar los elementos de la red		
Descripción	**:**	El componente permitirá filtrar por el tipo de elemento de la red y al seleccionar uno de estos elementos abrirá una nueva interfaz que muestre las características de ese elemento.
Fuente	**:**	Jimmy Gutiérrez
Prioridad	**:**	
Estabilidad	**:**	Transable
Fecha actualización	**:**	30/11/2019
Estado	**:**	No_Cumple
Incremento	**:**	4
Tipo	**:**	Funcional

RS019 – Implementar los componentes que permitan mostrar las características de cada elemento de la red.		
Descripción	**:**	En la ventana en que se muestran los elementos de la red, seleccionar alguno, se debe mostrar una ventana que describa ese elemento y las configuraciones que posee.
Fuente	**:**	Jimmy Gutiérrez
Prioridad	**:**	
Estabilidad	**:**	Transable
Fecha actualización	**:**	30/11/2019
Estado	**:**	No_Cumple
Incremento	**:**	4
Tipo	**:**	Funcional

RS020 – Implementar un componente que muestre un plano cartesiano		
Descripción	**:**	Este componente mostrara un plano cartesiano y debe ser mostrado a medida que se ejecuta el algoritmo para ver cómo se están comportando las soluciones obtenidas.
Fuente	**:**	Jimmy Gutiérrez
Prioridad	**:**	
Estabilidad	**:**	Transable
Fecha actualización	**:**	30/11/2019
Estado	**:**	No_Cumple
Incremento	**:**	4
Tipo	**:**	Funcional

RS021 – Mostrar las soluciones de los algoritmos en el plano cartesiano		
Descripción	**:**	El componente de gráficos debe permitir agregar el valor de la solución al plano cartesiano. Se deben poder mostrar las soluciones para problemas de un objetivo y para los de 2 objetivos.
Fuente	**:**	Jimmy Gutiérrez
Prioridad	**:**	
Estabilidad	**:**	Transable
Fecha actualización	**:**	30/11/2019
Estado	**:**	No_Cumple
Incremento	**:**	4
Tipo	**:**	Funcional

3.3 Matriz de Trazado Requisitos de Usuario vs. Requisitos de Software

La matriz de trazabilidad de los requisitos de usuario y de sistema que se presenta a continuación permite ver la relación y dependencia que un requisito de sistema tiene con los requisitos de usuario.

	RU001	RU002	RU003	RU004	RU005	RU006	RU007	
RS001								
RS002								
RS003								
RS004								
RS005								
RS006								
RS007								

4 Pruebas de Sistema

4.1 Pruebas de Usuario

En esta sección se especificarán las pruebas que se harán sobre el sistema, para determinar que se cumplen los requisitos de usuario. Una prueba puede dar lugar a muchos casos de prueba.

4.2 Matriz de Trazado Requisitos de Usuario vs. Pruebas

Tabla 1: Matriz de requisitos de usuario versus las pruebas

	RP1	RP2	RP3	RP4	RP5	RP6	RP7	RP8	RP9	RP10
RU1	x									X
RU2			x		X				x	
RU3										
RU4					x					