UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

ALGORITMO DE OPTIMIZACIÓN MULTIOBJETIVO PARA EL POSICIONAMIENTO DE ENRUTADORES EN REDES INALÁMBRICAS DE SENSORES Y ACTUADORES

Tesis presentada en opción al título de Máster en Informática Aplicada

Autor: **Ing. Jorge Martínez Padrón**

Tutor: **MSc. Ismael Armando Nodarse Mora**

La Habana, Julio de 2014

A mi mamá, a mi papá y a maire.

DECLARACIÓN JURADA DE AUTORÍA Y AGRADECIMIENTOS

Yo Jorge Martínez Padrón, con carné de identidad 86050605141, declaro que soy el autor principal del resultado que expongo en la presente memoria titulada “Algoritmo de optimización multiobjetivo para el posicionamiento de enrutadores en Redes Inalámbricas de Sensores y Actuadores”, para optar por el título de Máster en Informática Aplicada.

El presente trabajo fue desarrollado individualmente en el transcurso de los años 2012-2014.

En especial deseo agradecer al MSc. Ismael A. Nodarse Mora, tutor de esta investigación. Además, deseo agradecer a los integrantes del grupo de investigación Andrómeda y del proyecto Multisaber-El Navegante por su apoyo y comprensión. A todos ellos, así como a colegas y amigos que no he mencionado por razones de espacio, mi más sincero agradecimiento.

Finalmente declaro que todo lo anteriormente expuesto se ajusta a la verdad, y asumo la responsabilidad moral y jurídica que se derive de este juramento profesional.

Y para que así conste, firmo la presente declaración jurada de autoría en La Habana a los ____ días del mes de _____ del año 2014.

_____

Firma del Maestrante

Resumen

El despliegue de redes inalámbricas de sensores y actuadores (WSANs, por sus siglas en inglés) en entornos interiores representa un reto para los diseñadores; requiere experiencia en el tema y varias iteraciones de ensayo y error para encontrar un diseño optimizado. Como parte del grupo de investigación Andrómeda, se está desarrollando una herramienta que sugiera el diseño optimizado de WSANs; tomando como criterios de optimización el número de enrutadores, el consumo de energía por concepto de transmisión/recepción y la tolerancia a fallos. Uno de los principales problemas identificados en el desarrollo de la herramienta es optimizar, siguiendo los criterios definidos, el posicionamiento de los enrutadores a partir de las posiciones candidatas de los mismos; garantizando además la presencia de los dispositivos terminales (sensores y actuadores) y el coordinador, así como la conectividad entre ellos. El objetivo de esta investigación es desarrollar un algoritmo de optimización multiobjetivo que dé solución al problema planteado. Para lograrlo, luego de analizar soluciones similares, definir indicadores para la estimación de los criterios de optimización y analizar la complejidad del problema, se propusieron cinco algoritmos de optimización multiobjetivo: cuatro algoritmos evolutivos (NSGA-II, SPEA2, PAES y PESA-II) y un algoritmo memético (mNSGA-II) desarrollado como parte de esta investigación. Teniendo en cuenta los indicadores Épsilon y Radio de error se seleccionó el algoritmo mNSGA-II por sus resultados frente a una muestra de instancias reales del problema. Para validar la efectividad del mNSGA-II se compararon los valores de los indicadores de los criterios de optimización y las restricciones antes y después de aplicado el algoritmo a la muestra.

Palabras clave: redes inalámbricas de sensores y actuadores, posicionamiento de enrutadores, NP-Hard, algoritmos evolutivos, algoritmos meméticos

Índice general

Índice de figuras

Índice de tablas

Introducción

El término “red inalámbrica de sensores y actuadores” (WSAN, por sus siglas en inglés) hace referencia a un grupo de sensores y actuadores conectados por un medio inalámbrico de manera que puedan observar el entorno, procesar la información obtenida, tomar decisiones en base a esta información y llevar a cabo las acciones adecuadas [1]. Los sensores son dispositivos diseñados para medir una variable física o química de interés, como puede ser la temperatura, la humedad, la iluminación, la presión, la radiación, entre otras. Por otro lado, un actor o actuador, es un dispositivo que convierte una señal eléctrica en una acción física con repercusión en el entorno [2]. De manera general, ambos dispositivos están compuestos por un controlador, un transceptor, un módulo de alimentación y una memoria; aunque frecuentemente los actuadores son dispositivos más complejos, con fuentes de alimentación más duraderas y mayor capacidad de procesamiento y transmisión.

Los enrutadores (también conocidos como *relays* o repetidores) son dispositivos opcionales que tienen como objetivo propiciar las comunicaciones que ocurren en la red. Utilizado fundamentalmente en WSANs de mediano y gran tamaño, los enrutadores favorecen aspectos como la conectividad, la escalabilidad, el balance de cargas y el tiempo de vida de la red [3].

Otro dispositivo presente en las WSANs es el coordinador. Dependiendo de la arquitectura de la red, puede funcionar como *gateway* entre la WSAN y una red externa (internet, por ejemplo) o como el dispositivo central que procesa la información recibida de los sensores y decide qué acción deben llevar a cabo los actuadores [4].

Estas redes pueden ser utilizadas como parte integral de sistemas de vigilancia, de control de edificios [5], de automatización del hogar [6], de monitoreo ambiental [7], entre otros. Por ejemplo, como parte de un sistema de control de edificios, una WSAN dotada de sensores de movimiento e iluminación puede detectar la presencia de seres humanos en una habitación y dependiendo de la iluminación natural, encender las luces artificiales.

Un punto clave para el adecuado funcionamiento de una WSAN es la ubicación física de los dispositivos que la conforman. Aspectos vitales como la conectividad, la ausencia de zonas muertas (sin cobertura), la disminución del costo, la reducción del consumo eléctrico, la latencia, la interferencia y la tasa de error por paquete están estrechamente relacionados con el posicionamiento de los elementos de la red [8]. En consecuencia, el

posicionamiento óptimo de los dispositivos de una WSAN es un problema de interés para la industria y la comunidad científica.

El grupo de investigación Andrómeda, producto de una colaboración entre la Universidad de las Ciencias Informáticas y la Universidad de Málaga, se encuentra desarrollando una herramienta que proponga el diseño de una WSAN basada en el estándar ZigBee [9]. Para ello debe tener en cuenta las características del entorno donde será desplegada y otras restricciones especificadas por el diseñador de la red.

Entre las principales características de la herramienta se encuentran que: 1) está basada en un modelo de despliegue propuesto específicamente para WSAN [10] y 2) le permite al diseñador especificar la ubicación física de los dispositivos terminales (sensores y actuadores) y del coordinador.

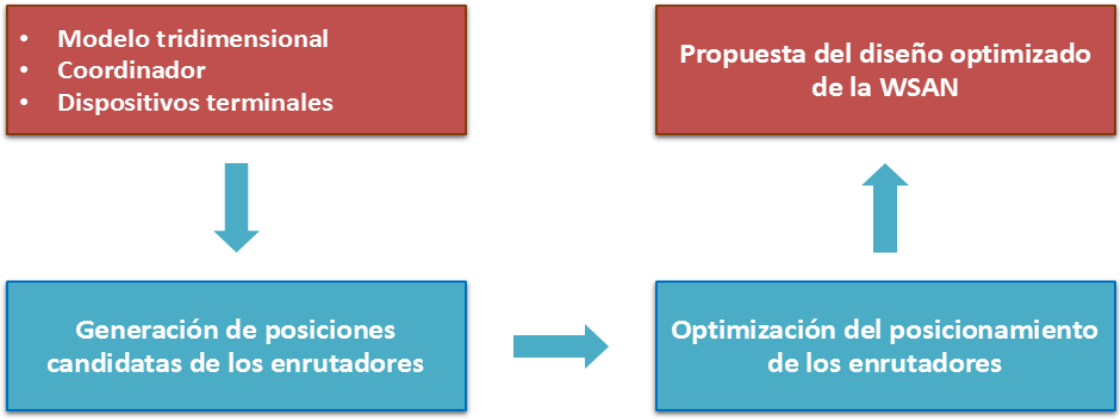


Figura 1. Diagrama de bloques que explica el funcionamiento de la herramienta.

Como entrada, dicha herramienta recibirá el modelo tridimensional del edificio donde se desea desplegar la red, la ubicación del coordinador (*ZigBee Coordinator*) y las ubicaciones de los dispositivos terminales (*ZigBee End Devices*). A partir de estos elementos se generarán las posiciones candidatas que pueden ocupar los enrutadores (*ZigBee Routers*); de manera que quede garantizada la conectividad entre los dispositivos de la red y un posicionamiento adecuado con relación al entorno. Posteriormente se optimiza el posicionamiento de los enrutadores utilizando como criterios: 1) minimizar el número de enrutadores, 2) minimizar el consumo de energía por transmisión/recepción y 3) maximizar la tolerancia de la red al fallo de sus dispositivos. Se incluyen además, como restricciones: 1) que todos los dispositivos terminales y el coordinador estén incluidos en la red optimizada y 2) que todos los dispositivos que la

componen estén conectados entre sí. Finalmente, el resultado es presentado al diseñador como una propuesta optimizada de la WSAN.

El objetivo del proceso de Generación de posiciones candidatas de los enrutadores es garantizar la conectividad entre los dispositivos terminales y el coordinador; ubicando enrutadores en posiciones válidas de acuerdo al modelo del edificio. Para inferir la conectividad entre los dispositivos se utilizan los modelos de propagación *Multi-Wall* [11], *Standard Ray Tracing* [12] o *Dominant Path* [13] según las características del entorno. Por las características de este proceso, la WSAN resultante por lo general presenta un gran número de enrutadores y un excesivo consumo de energía por concepto de transmisión/recepción; aunque, vale señalar, posee una elevada tolerancia a fallos. Tomando esta WSAN resultante, el proceso de Optimización del posicionamiento de los enrutadores debe encontrar una WSAN que mejore los criterios de optimización descritos y al mismo tiempo cumpla las restricciones especificadas.

Teniendo en cuenta los elementos anteriormente planteados, se identifica el siguiente **problema de investigación**: ¿Cómo minimizar el número de enrutadores y el consumo de energía de una WSAN, maximizando la tolerancia a fallos y garantizando la presencia de los dispositivos terminales y del coordinador, así como la conectividad entre ellos?

En ese sentido, se define como **objeto de estudio** el diseño de WSANs y como **campo de acción** el posicionamiento de enrutadores en WSANs.

El **objetivo general** de esta investigación es desarrollar un algoritmo de optimización multiobjetivo que permita minimizar el número de enrutadores y el consumo de energía de una WSAN, maximizando la tolerancia a fallos y garantizando la presencia de los dispositivos terminales y del coordinador, así como la conectividad entre ellos.

Se plantea como **hipótesis** que si se desarrolla un algoritmo de optimización multiobjetivo para el posicionamiento de los enrutadores de una WSAN, es posible minimizar el número de enrutadores y el consumo de energía, maximizando la tolerancia a fallos y garantizando la presencia de los dispositivos terminales y del coordinador, así como la conectividad entre ellos.

Del objetivo general se desglosan los siguientes **objetivos específicos**:

- Establecer los fundamentos teóricos y metodológicos relacionados con el diseño de WSANs.

- Desarrollar un algoritmo de optimización multiobjetivo para el posicionamiento de los enrutadores en WSANs teniendo en cuenta los criterios de optimización y las restricciones definidas.
- Comparar, teniendo en cuenta los criterios de optimización y las restricciones definidas, el diseño de las WSANs antes y después de aplicar el algoritmo desarrollado.

Aporte práctico

El buen funcionamiento de la WSAN está en gran medida relacionado con la optimización del posicionamiento de los enrutadores. Además de darle solución a una de las problemáticas fundamentales que enfrenta el grupo de investigación Andrómeda, el resultado de esta investigación puede ser extrapolado fácilmente al posicionamiento de enrutadores en redes inalámbricas de sensores (WSN, pos sus siglas en inglés).

Novedad científica

La optimización del posicionamiento de los enrutadores en WSANs no ha sido abordada en la bibliografía consultada con los criterios de optimización y las restricciones planteados en esta investigación. En ese sentido, la solución pretende apoyar la validación de una novedosa estrategia de despliegue propuesta específicamente para WSANs [10] que pudiera ser utilizada con éxito en la práctica y/o servir como banco de pruebas para próximas investigaciones.

Métodos de investigación

- Analítico-sintético: Para descomponer el problema de investigación en varios elementos; de esta forma será más sencillo profundizar en el estudio de cada elemento, para luego sintetizarlos en la solución de la propuesta.
- Histórico-lógico: Para llevar a cabo el estudio crítico del proceso de diseño de una WSAN, así como de las estrategias para la resolución de problemas de optimización multiobjetivo.
- Inductivo-deductivo: Para identificar la técnica con que se desarrollará el algoritmo.
- Experimento: Para la verificación de la hipótesis.

Estructura del documento

El presente documento está estructurado en 3 capítulos cuyos contenidos se resumen a continuación:

Capítulo 1: Se exponen los principales conceptos relacionados con el diseño de WSANs y sus similitudes con las WSNs. Se analizan dos grupos de soluciones similares, destacándose las estrategias de resolución y las estructuras que utilizan para la modelación. Se describen los indicadores que se utilizarán para estimar los criterios de optimización y las restricciones del problema. Se describe formalmente el problema y se clasifica de acuerdo a su complejidad computacional. También se analizan las estrategias identificadas para la resolución de problemas de optimización multiobjetivo y se valoran varios *frameworks* para la implementación y comparación de algoritmos de optimización multiobjetivo.

Capítulo 2: Se propone la modelación del problema especificando la codificación de los individuos y los algoritmos para el cálculo de los indicadores de los criterios de optimización y las restricciones. Además, se proponen cinco algoritmos de optimización multiobjetivo: cuatro algoritmos evolutivos (NSGA-II, SPEA2, PAES y PESA-II) y un algoritmo memético (mNSGA-II) desarrollado como parte de esta investigación. Por último se describen los operadores evolutivos utilizados por los algoritmos propuestos.

Capítulo 3: Se especifica el diseño de los experimentos efectuados para la selección del algoritmo de optimización multiobjetivo y para la validación del algoritmo seleccionado. Se presentan, además, los resultados de la aplicación de estos experimentos y el análisis estadístico realizado.

Capítulo 1. Preliminares

En este capítulo se introducen conceptos relacionados con el diseño de WSANs, su relación con las redes inalámbricas de sensores (WSNs, por sus siglas en inglés) y los métodos de estimación más utilizados para los criterios de optimización y las restricciones que se toman en cuenta en esta investigación. Además, se define formalmente el problema y se hace un análisis de su complejidad computacional. Por otro lado, también se analizan problemas similares y las estrategias utilizadas para su resolución. Por último, se consideran varios *frameworks* para la implementación y comparación de algoritmos de optimización multiobjetivo.

1.1. Diseño de WSANs

Las WSANs son consideradas una evolución natural de las WSNs ante la necesidad de automatizar ciertos procesos en sistemas con requerimientos especiales [14]. Aunque tienen muchos puntos en común, el uso de actuadores para responder a los eventos detectados por los sensores ha incluido características particulares a las WSANs.

En las WSNs tienen lugar comunicaciones sensor/sensor y sensor/coordinador; en las WSANs ocurren además comunicaciones sensor/sensor, sensor/coordinador, sensor/actuador, actuador/actuador y coordinador/sensor. Otras de las particularidades de las WSANs con relación a las WSNs son [1]:

- Heterogeneidad de los dispositivos: Por lo general los sensores presentan menos capacidad de cómputo y de comunicación inalámbrica que los actuadores.
- Respuesta en tiempo real: En muchos casos es necesario que las acciones que llevarán a cabo los actuadores ocurran tan pronto como los sensores detecten el estímulo que las dispara.
- Despliegue de sensores y actuadores: Según el caso, la red puede necesitar cientos de sensores y unos pocos actuadores en los puntos del entorno que lo requieran, o cientos de actuadores y unos pocos sensores.

Los enrutadores tienen la misma función en ambas redes; sin importar qué dispositivos se comuniquen, su objetivo es garantizar la conectividad, mejorar la escalabilidad, el balance de carga y el tiempo de vida.

Desde el punto de vista de su arquitectura, las WSANs se dividen en 2 grupos: las de arquitectura automática y las de arquitectura semiautomática. Cuando los sensores de

una WSAN de arquitectura automática detectan un fenómeno, transmiten su lectura a los actuadores; estos procesan la información y ejecutan la acción apropiada.

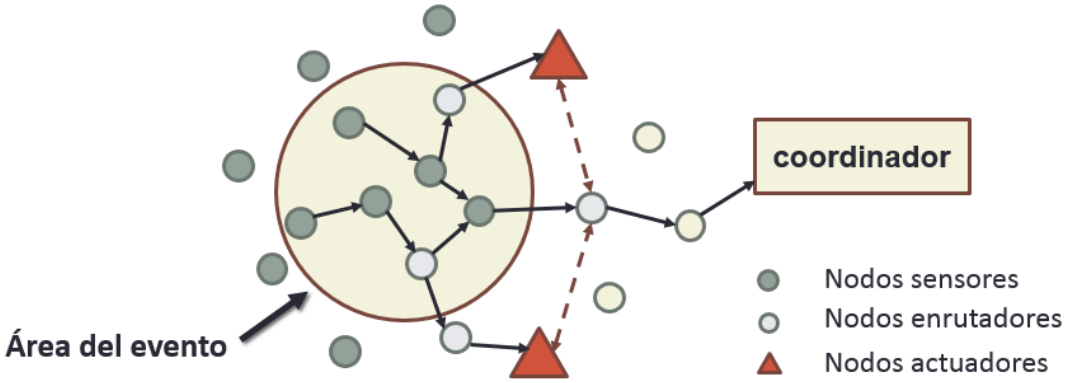


Figura 2. WSAN con arquitectura automática.

Por otro lado, en las de arquitectura semiautomática, los sensores transmiten su lectura hacia el coordinador; este la procesa y se encarga de comunicar a los actuadores la acción que deben ejecutar.

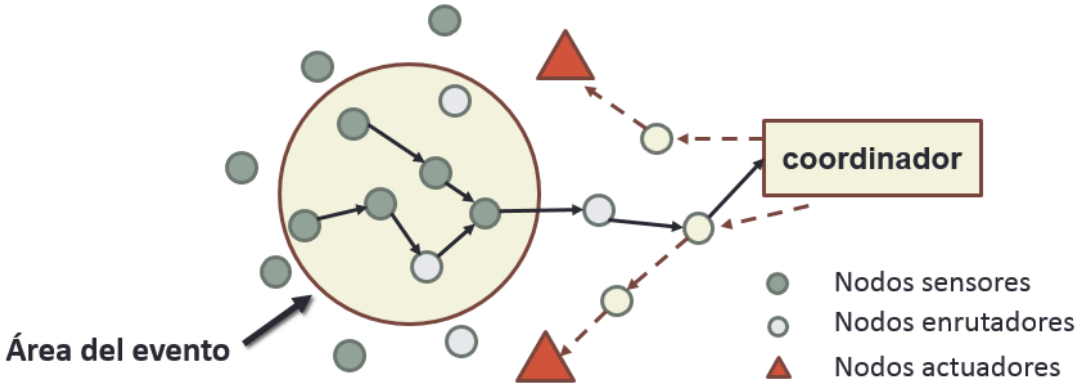


Figura 3. WSAN con arquitectura semiautomática.

Las WSANs de arquitectura automática presentan tasas de transferencias más bajas, menor consumo de energía y un tiempo de vida de la red más elevado. Por otro lado, las de arquitectura semiautomática requieren algoritmos de comunicación y coordinación más simples y su arquitectura es similar a la utilizada por las WSNs [1]. Teniendo en cuenta estas características y la naturaleza del sistema donde se va a utilizar la WSAN, los diseñadores seleccionan una u otra arquitectura para el diseño de la red a desplegar.

1.2. Soluciones similares

En la bibliografía consultada no se encontró ninguna referencia a la optimización del posicionamiento de los enrutadores a partir de una WSAN existente. Con el objetivo de identificar una técnica adecuada para la resolución de este problema, se analizan dos grupos de soluciones de problemas similares que comparten, sobre todo, la modelación de sus instancias y los criterios de optimización utilizados.

1.2.1. Optimización del posicionamiento de enrutadores en WSNs

Este problema consiste en encontrar una distribución de enrutadores que garantice cierto grado de conectividad de la WSN, utilizando un número fijo de enrutadores o la menor cantidad posible. El hecho de que los enrutadores son dispositivos opcionales en las WSNs ha condicionado que no sea un problema tan abordado por la comunidad científica. Más allá de la modelación del dominio, a este problema se le asocian criterios de optimización similares a los que se manejan en esta investigación.

Tabla 1. Soluciones analizadas para la optimización del posicionamiento de enrutadores en WSNs.

Año	Título	Estrategia	Modelación
2007	Fault-tolerant Relay Node Placement in Wireless Sensor Networks [15]	Algoritmo de aproximación	Grafo
2007	Fault-tolerant Relay Node Placement in Wireless Sensor Networks: Problems and Algorithms [16]	Algoritmo de aproximación	Grafo
2010	Fault-Tolerant Relay Node Placement in Heterogeneous Wireless Sensor Networks [17]	Algoritmo de aproximación	Grafo
2011	A multiobjective Approach to the Relay Placement Problem in WSNs [18]	Algoritmo evolutivo (memético)	Grafo
2012	Optimal Relay Placement for Indoor Sensor Networks [19]	Algoritmo de aproximación	Cuadrícula

En [15] se analizan 2 variantes del problema: 1) cantidad mínima de enrutadores garantizando la existencia de al menos un camino de vértices disjuntos y 2) cantidad mínima de enrutadores garantizando la existencia de al menos 2 caminos de vértices disjuntos. En ambos casos se proponen algoritmos con tiempo de ejecución polinomial, con factores de aproximación $(6 + \varepsilon)$ y $(24 + \varepsilon)$ respectivamente; donde ε representa el factor de aproximación del algoritmo que utilizan para calcular la cobertura de disco mínimo (*min-disk-cover*) [20]. La WSN es modelada como un grafo.

La propuesta presentada en [16] incluye además los conceptos de *single-tiered* (un nivel) y *two-tiered* (dos niveles), para referirse a las WSN en que los enrutadores y los sensores

participan en el reenvío de paquetes y a las que solo los enrutadores intervienen respectivamente. En todos los casos se proponen algoritmos con tiempo de ejecución polinomial, con factores de aproximación de $(14 + \varepsilon)$ y $(20 + \varepsilon)$; esta solución también está basada en el algoritmo propuesto en [20] para calcular la cobertura de disco mínimo (*min-disk-cover*), con factor de aproximación ε. En este caso la WSN es modelada también como un grafo.

El estudio presentado en [17] toma en cuenta que los sensores pueden ser heterogéneos; por tanto, sus radios de transmisión pueden variar. Otra característica es que no se restringe el problema a una cantidad k de caminos de vértices disjuntos, sino que se plantean algoritmos con tiempo de ejecución polinomial con factores de aproximación en función de k. Por otro lado, se separa el problema según el nivel de tolerancia a fallo en: completo) cantidad mínima de enrutadores garantizando una cantidad k de caminos de vértices disjuntos entre dos pares de dispositivos cualesquiera (sensores o enrutadores) y parcial) cantidad mínima de enrutadores garantizando una cantidad k de caminos de vértices disjuntos entre dos pares de enrutadores cualesquiera. Para cada caso se obtienen factores de aproximación de $O(\sigma k^3)$ y y $O(\sigma k^2)$ respectivamente; donde σ es el factor de aproximación del algoritmo propuesto en [21] para la búsqueda del grafo de expansión mínimo. La WSN es modelada como un grafo.

Tomando como criterios el número de enrutadores y el consumo de energía, en [18] se propone un algoritmo memético [22] para la resolución de esta problemática. En este trabajo se modela la WSN como un grafo en el que se ubican los sensores, los enrutadores y un conjunto de puntos críticos que deben quedar dentro del área de cobertura de la red optimizada. El algoritmo propuesto utiliza dos heurísticas de búsqueda local, una relacionada con la conectividad y la reducción de ciclos en el grafo resultante y otra asociada a las distancias entre dispositivos modelando el grafo resultante como un espacio euclídeo.

En [19] se aborda la optimización del posicionamiento de enrutadores en WSNs en entornos interiores. Para ello utilizan un modelo de propagación que predice el radio de transmisión de los enrutadores. En este caso, se utiliza para la modelación de la WSN una cuadrícula y se propone para la resolución del problema un algoritmo voraz (greedy) con un factor de aproximación de $\ln(n) + 1$.

1.2.2. **Optimización del posicionamiento de sensores en WSNs**

Este problema ha sido ampliamente estudiando por la comunidad científica y la industria por más de 10 años. En términos generales, consiste en encontrar una distribución con el número mínimo de sensores, de manera que quede cubierta el área a sensar y exista conectividad entre ellos y el coordinador de la red. Por lo general se le asocian otros criterios de optimización como la tolerancia a fallos y el consumo de energía. Comparte con el problema planteado en esta investigación el dominio y los criterios de optimización.

Tabla 2. Soluciones analizadas para la optimización del posicionamiento de sensores en WSNs.

Año	Título	Estrategia	Modelación
2007	Adaptive design optimization of wireless sensor networks using genetic algorithms [23]	Algoritmo evolutivo	Cuadrícula
2009	Genetic algorithm based node placement methodology for wireless sensor networks [24]	Algoritmo evolutivo	Cuadrícula
2008	Optimal sensor network layout using multi-objective metaheuristics [25]	Algoritmo evolutivo	Cuadrícula
2009	Multi-objective optimization for coverage control in wireless sensor network with adjustable sensing radius [26]	Algoritmo evolutivo	Grafo
2010	A multi-objective evolutionary algorithm for the deployment and power assignment problem in wireless sensor networks [27]	Algoritmo evolutivo	Plano bidimensional
2010	Multi-Objective WSN Deployment: Quality of Monitoring, Connectivity and Lifetime [28]	Algoritmo evolutivo (memético)	Grafo
1012	Multiobjective Sensor Node Deployement in Wireless Sensor Networks [29]	Optimización de enjambres de partículas	Grafo
2012	Application of Multiobjective Particle Swarm Optimization to maximize Coverage and Lifetime of wireless Sensor Network [30]	Optimización de enjambres de partículas	Cuadrícula
2013	A PSO-Optimized Minimum Spanning Tree-Based Topology Control Scheme for Wireless Sensor Networks [31]	Optimización de enjambres de partículas	Grafo
2009	A Novel Approach For Optimal Wireless Sensor Network Deployment [32]	Recocido simulado	Plano bidimensional

En [23] y [24] se propone abordar este problema utilizando algoritmos genéticos. En ambos casos se toman en cuenta un gran número de criterios de optimización (7 y 5 criterios respectivamente) dentro de los que se encuentran el número de sensores, el consumo de energía y el tiempo de vida de la WSN. En las dos soluciones la WSN es modelada mediante una cuadrícula (*grid*).

En [25] se comprueba la factibilidad de varios algoritmos genéticos para la resolución de este problema; se utilizan como criterios de optimización la cobertura y el tiempo de vida, como restricción se establece la conectividad de la WSN. Los algoritmos implementados son: MOEA, NSGA-II y dos variaciones de IBEA; desde el punto de vista estadístico no se encontraron diferencias significativas pero el MOEA mostró resultados ligeramente mejores. La WSN es modelada como una cuadrícula.

Partiendo de un conjunto potencial de sensores ubicados aleatoriamente, en [26] se propone la utilización del algoritmo NSGA-II para encontrar un subconjunto de sensores teniendo en cuenta como criterios de optimización el número de elementos, el consumo de energía y la cobertura del área a sensar. En este caso se toma en cuenta que los radios de transmisión y sensado de los sensores pueden ser variables. A modo de demostración, se realiza una comparación del comportamiento de los criterios de optimización al aplicar el algoritmo propuesto y el OGDC [33]; la propuesta presentada arroja mejores resultados. En este caso la WSN es modelada como un grafo.

Teniendo en cuenta como criterios de optimización la cobertura y el tiempo de vida de la WSN, en [27] se propone el uso de un algoritmo evolutivo multiobjetivo basado en descomposición (MODE/D) para la optimización del posicionamiento de los enrutadores. Para su resolución, el problema original es dividido en varios sub-problemas que son resueltos en paralelo. En este caso, se realiza un estudio estadístico del comportamiento de los criterios de optimización al aplicar el algoritmo propuesto y el NSGA-II; demostrándose la factibilidad de la propuesta presentada. La WSN es modelada en un plano bidimensional.

En [28] se propone una solución basada en algoritmos evolutivos y búsqueda por vecindad. En este caso, luego de encontrar las soluciones dominantes (frente de Pareto) con el algoritmo evolutivo propuesto, se utiliza una Búsqueda tabú [34] para seleccionar una solución individual. La WSN es modelada como un grafo y los criterios de optimización tomados en cuenta son la cobertura y el tiempo de vida asociado al agotamiento de la energía; la conectividad queda especificada como una restricción. La solución es comparada con otras estrategias utilizadas para resolver este problema y arroja resultados positivos.

Otra estrategia híbrida es propuesta en [29], en este caso basada en Optimización de enjambres de partículas (PSO, por sus siglas en inglés) y Lógica difusa (*Fuzzy Logic*).

Utilizando una regla, se clasifican los sensores en buenos, normales o malos de acuerdo a indicadores relevantes para los criterios de optimización utilizados; los sensores buenos son fijados en la solución y mediante PSO se seleccionan el resto de los sensores necesarios para cubrir el área a sensar. Los criterios de optimización utilizados en este caso son la cobertura, la conectividad y el tiempo de vida asociado al agotamiento de la energía. La WSN es modelada como un grafo.

En [30] se propone un algoritmo basado en PSO utilizando como criterios de optimización la cobertura y el tiempo de vida de la WSN. La red es modelada en un plano bidimensional donde la posición de los sensores está representada por una coordenada x, y; esto facilita el cálculo del área de acción de cada sensor. En este caso no se garantiza la conectividad de los sensores ni se comparan los resultados del algoritmo propuesto con otras soluciones.

Otra solución basada en PSO es propuesta en [31] utilizando como criterios de optimización la conectividad y el consumo de energía; la cobertura es considerada una restricción. El problema original es transformado en la búsqueda del árbol de expansión mínima con restricción en los grados (*degree constrained minimum spanning tree*); consecuentemente la WSN es modelada como un grafo. Los resultados obtenidos son comparados con los algoritmos NSGA-II, PSEA 2 y MOGA.

Utilizando como criterios de optimización la cobertura, la conectividad y el consumo de energía, en [32] se propone una solución basada en Recocido Simulado (SA, por sus siglas en inglés). En este caso no se toma en cuenta la cantidad de sensores necesarios para cubrir el área de interés ni se comparan los resultados del algoritmo presentado. La WSN es modelada en un plano bidimensional.

1.3. Indicadores para los criterios de optimización y las restricciones

Como se puede apreciar en las Tablas 1 y 2, la estructura más utilizada para la representación de WSNs en las soluciones similares analizadas son los grafos. Además de ser una estructura natural para la modelación de redes, emplear grafos para representar las instancias del problema permite aprovechar los elementos de teoría de grafos; tornándose especialmente útil para el cálculo de los indicadores con que se estiman los criterios de optimización y las restricciones.

Tabla 3. Indicadores para los criterios de optimización y restricciones en soluciones modeladas como grafos.

Referencia	Criterios de optimización	Indicadores	Restricciones	Indicadores
[15]	Minimizar el número de enrutadores	Número de vértices	Conectividad	Al menos un camino entre cada par de vértices
[16]	Minimizar el número de enrutadores	Número de vértices	Conectividad	Al menos un camino entre cada par de vértices
[17]	Minimizar el número de enrutadores	Número de vértices	Conectividad	Al menos un camino entre cada par de vértices
	Maximizar la tolerancia a fallos	Caminos de vértices disjuntos		
[18]	Minimizar el número de enrutadores	Número de vértices		
	Minimizar el consumo de energía por transmisión/recepción	Costo del camino mínimo entre los sensores y el coordinador		
[26]	Minimizar el número de sensores	Número de vértices		
	Minimizar el consumo de energía del proceso de sensado	Sumatoria del consumo de energía en cada vértice		
	Maximizar cobertura	Unión de las áreas sensadas por cada vértice		
[28]	Maximizar la cobertura	Unión de las áreas sensadas por cada vértice	Conectividad	Al menos un camino entre cada par de vértices
	Minimizar el consumo de energía por transmisión/recepción	Grado máximo del grafo		
[29]	Maximizar el área a sensar	Unión de las áreas sensadas por cada vértice		
	Maximizar la conectividad	Caminos de vértices disjuntos		

	Minimizar el consumo de energía en el proceso de sensado	Sumatoria del consumo de energía en cada vértice		
[31]	Maximizar la conectividad	Distancia entre los vértices	Cobertura	Grado mínimo del grafo
	Minimizar el consumo de energía en el proceso de sensado	Sumatoria del consumo de energía en cada vértice		

A pesar de las diferencias analizadas en el epígrafe 1.1 entre las WSNs y las WSANs, varios de los indicadores mostrados en la Tabla 3 pueden ser utilizados directamente para la estimación de los criterios de optimización y las restricciones del problema analizado en esta investigación. Por ejemplo, para la estimación del **número de enrutadores**, en todos los casos analizados se utiliza como indicador el número de vértices presentes en la solución.

En el caso del **consumo de energía por transmisión/recepción**, en [18] se propone la sumatoria del costo del camino mínimo entre cada vértice que representa un sensor y el vértice que representa el coordinador. Si bien en el dominio de las WSNs este indicador tiene sentido, en las WSANs no puede ser aplicado porque no en todos los casos los sensores deben enviar su lectura al coordinador; por ejemplo en las WSANs con arquitectura automática. En el enfoque presentado en [28] se establece una relación entre las transmisiones y recepciones del sensor y el grado del mismo; de manera que disminuyendo el grado máximo del grafo se reduce el consumo de energía por concepto de transmisión/recepción. Por su generalidad, este indicador resulta más adecuado en el contexto de las WSANs.

Para estimar la **tolerancia a fallos**, en [17] se propone como indicador la cantidad k de caminos de vértices disjuntos (k *vertex-disjoint paths*) que existen entre cada uno de sus dispositivos; de manera tal que la red siga siendo conexa ante el fallo de $k-1$ dispositivos. La factibilidad de este indicador para estimar la tolerancia a fallos es analizada en [35].

Con relación a las restricciones, para comprobar la **presencia de los dispositivos terminales y del coordinador** se propone la definición de un conjunto de vértices D donde estén representados los mismos; de manera que el indicador en este caso es la presencia de los vértices en D como parte de los vértices de la solución representada.

Para comprobar la **conectividad**, en todas las soluciones estudiadas que la definen como restricción se propone como indicador la existencia de al menos un camino entre cada par de vértices.

Tabla 4. Relación entre los criterios de optimización y las restricciones del problema de esta investigación y sus indicadores.

Criterios de optimización y restricciones	**Indicador**
Minimizar el número de enrutadores	Número de vértices
Minimizar el consumo de energía	Grado máximo del grafo
Maximizar la tolerancia a fallos	Número de caminos de vértices disjuntos
Presencia de los dispositivos terminales y el coordinador	Presencia de los vértices que representan a los dispositivos terminales y el coordinador en los vértices de la solución
Conectividad	Al menos un camino entre cada par de vértices

1.4. Definición formal y complejidad computacional del problema

Teniendo en cuenta la modelación de las instancias del problema y los indicadores definidos, el problema de esta investigación se define formalmente como:

Sea $G = (V, E)$ *un grafo conexo no dirigido, un conjunto* $R \subsetneq V$ *y un conjunto* $D := \{N \cup \{c\}\}(\subsetneq V)$, *de manera que* $V = (R \cup D)$, *encontrar un subgrafo conexo* $G' = (V', E')$ *que satisfaga las siguientes funciones objetivo:*

(1) Min $|V'|$

(2) Min $\Delta(G')$

(3) Max $k - connected(G')$

Sujeto a:

$D \subseteq V'$

$k - connected(G') \geq 1$

Donde el grafo $G = (V, E)$ representa la WSAN obtenida de la Generación de posiciones candidatas de los enrutadores. El conjunto de vértices V está compuesto por los dispositivos terminales N, el coordinador c y las posiciones candidatas de los enrutadores R; las aristas E, constituyen las conexiones posibles entre ellos. El subgrafo G', inducido por G, representa una posible solución del problema. $|V'|$ denota la cantidad de vértices en G', $\Delta(G')$ el grado máximo en G' y $k - connected(G')$ el número de caminos de vértices disjuntos en G'. Como referencia al problema planteado con

anterioridad se utilizarán las siglas OPE (Optimización del posicionamiento de enrutadores).

Incluso sin tener en cuenta la función objetivo relacionada con el grado máximo del grafo, (2), el problema descrito en la presente investigación pertenece a la clase de problemas NP-Hard. Para demostrarlo se utilizará el problema Conectividad externa a un conjunto (*Outconnectivity to a Subset Problem*, OSP), clasificado como NP-Hard de acuerdo a su complejidad computacional en [36] a partir del problema del ciclo hamiltoniano.

OSP: Dado un grafo ponderado no dirigido $\Omega = (Ü, Ë)$, un vértice especial $r \subsetneq Ü$, un conjunto $S \subsetneq Ü$ y un número $p \geq 1$; encontrar el subgrafo inducido de costo mínimo $\Omega' = (Ü', Ë')$ de manera que $\{r \cup S\}(\subseteq Ü')$ y existan al menos p caminos de vértices disjuntos entre cada par de vértices en Ω'.

El problema de decisión asociado a OSP puede ser definido como:

Instancia:

- Un grafo conexo ponderado $\Omega = (Ü, Ë)$
- Un vértice $r \subsetneq Ü$
- Un conjunto de vértices $S \subsetneq Ü$
- Un número $q > 0$
- Un número $p \geq 1$

Interrogante:

¿Existe un subgrafo inducido $\Omega' = (Ü', Ë')$ donde $r \subsetneq Ü'$, $S \subsetneq Ü'$, $\{r \cup S\}(\subseteq Ü')$, el costo de Ω' es menor o igual que q y existen al menos p caminos de vértices disjuntos entre los vértices de Ω'?

El problema de decisión asociado a la relajación del problema de la optimización del posicionamiento de los enrutadores descrito con anterioridad (Relaxed-OPE en lo adelante) puede ser definido como:

Instancia:

- Un grafo conexo no ponderado $G = (V, E)$
- Un vértice $c \subsetneq V$
- Un conjunto de vértices $N \subsetneq V$
- Un número $m \geq 2$

- Un número $k \geq 1$

Interrogante:

¿Existe un subgrafo inducido $G' = (V', E')$ donde $c \subsetneq V', N \subsetneq V', \{N \cup \{c\}\} (\subseteq V'), |V'| \leq m$ y existan al menos k caminos de vértices disjuntos?

Sea $I = (\Omega = (\ddot{U}, \ddot{E}), r, S, q, p)$ una instancia de OSP, el siguiente algoritmo produce una instancia de Relaxed-OPE $J = (G = (V, E), c, N, m, k)$ tomando I como entrada:

```
reduction_algorithm (I = (Ω = (Ü,Ë),r,S,q,p)):
  c ← r
  N ← S
  m ← q
  k ← p
  for i = 1 to |Ü| do:
    neighbors ←adjacent vertexes of Ü[i]
    for j = 1 to |neighbors| do:
      weight ←weight of edge between Ü[i] and neighbors[j]
      Remove edge between Ü[i] and neighbors[j]
      for w = 1 to weight -1 do:
        Add a new vertex between Ü[i] and neighbors[j] linked by an edge of weight 1
      end
    end
  end
  G ← Ω
  return J = (G,c,N,m,k)
```

Figura 4. Algoritmo para generar una instancia del problema Relaxed-OPE a partir de una instancia del OSP.

Aunque ambos problemas de decisión son similares, es necesario realizar un ajuste con relación a los vértices y las aristas de Ω debido a que la función de costo de OSP está dada por la sumatoria de los pesos de las aristas incluidas en el subgrafo Ω'. Por tanto, cada arista de peso x se sustituye por $x-1$ vértices y x aristas de peso 1, de manera que los vértices continúen conectados. El algoritmo propuesto para la reducción tiene un tiempo de ejecución polinomial de $O(w|\ddot{U}|^2)$, donde w es la cota superior del peso asociado a las aristas de Ω.

Una vez encontrado un algoritmo de reducción que produce una instancia del Relaxed-OPE a partir de una instancia del OSP en tiempo polinomial, es posible afirmar que Relaxed-OPE es computacionalmente al menos tan costoso de resolver como el OSP. Por tanto, es posible afirmar también que el problema de la optimización del posicionamiento de los enrutadores planteado pertenece a la clase de problemas NP-Hard; siendo al menos tan complejo como su versión relajada Relaxed-OPE.

1.5. Estrategias identificadas para la resolución de problemas de optimización multiobjetivo

Los problemas de optimización multiobjetivo (MOPs, por sus siglas en inglés), también conocidos como de optimización multicriterio, tratan de encontrar valores para las variables de decisión de manera que cumplan con las restricciones del problema y ofrezcan valores óptimos para más de una función objetivo. En los problemas de optimización con una sola función objetivo, las soluciones óptimas están asociadas a un único valor de la función objetivo. Sin embargo, en los MOPs, varias soluciones con valores distintos en las funciones objetivo pueden ser consideradas óptimas.

Un MOP general se define formalmente como: *minimizar (o maximizar)* $F(x) = \big(f_1(x), \dots, f_k(x)\big)$ *sujeto a* $g_i(x) \leq 0, i = \{1, \dots, m\}$ *y* $h_j(x) = 0, j = \{1, \dots, p\}$ $x \in \Omega$. *Una solución del MOP minimiza (o maximiza) los componentes del vector* $F(x)$ *donde la variable de decisión* x *es un vector de dimensión* n $x = (x_1, \dots, x_n)$ *perteneciente a un universo Ω. Nótese que* $g_i(x) \leq 0$ *y* $h_j(x) = 0$ *son restricciones mientras se minimiza (o maximiza)* $F(x)$*; además,* Ω *contiene todas las posibles* x *que pueden ser usadas para satisfacer una evaluación de* $F(x)$. [37]

En los MOPs el concepto de óptimo varía; al tener en cuenta varias funciones objetivo, en lugar de buscar un óptimo global se buscan soluciones que ofrezcan buenos resultados en todas las funciones objetivo. En este entorno, el concepto de óptimo más utilizado es el de Optimalidad de Pareto; definido formalmente como: *Una solución* $x \in \Omega$ *se dice que es óptima con respecto a* Ω *si y solo si no existe una* $x' \in \Omega$ *para la cual* $v = F(x') = (f_1(x'), \dots, f_k(x'))$ *es dominada por* $u = F(x) = (f_1(x), \dots, f_k(x))$. Por otro lado, el concepto de dominancia en este ámbito puede ser definido como: Un vector $u = (u_1, \dots, u_k)$ se dice que domina a otro vector $v = (v_1, \dots, v_k)$ si y solo si u es parcialmente menor que v. Es decir, $\forall i \in \{1, \dots, k\}, u_i \leq v_i \wedge \exists i \in \{1, \dots, k\} : u_i < v_i$. [38]

A continuación se analizan las estrategias para la resolución de MOPs identificadas en el estudio de soluciones similares.

1.5.1. Algoritmos evolutivos

De manera general, los algoritmos evolutivos son métodos de búsqueda de soluciones inspirados en los procesos y mecanismos de la evolución biológica propuesta por Darwin, teniendo en cuenta la variedad de los organismos vivos y su adaptación al medio [37]. En la Figura 5 se describe el esquema general de un algoritmo evolutivo.

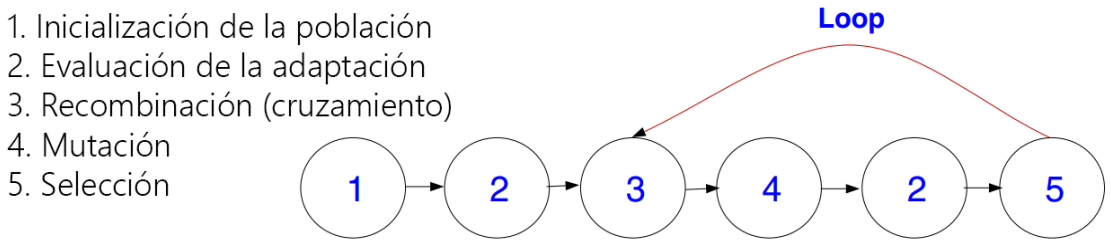


Figura 5. Esquema general de un algoritmo evolutivo. [37]

Esta técnica es utilizada típicamente para encontrar buenas soluciones aproximadas a problemas que no pueden resolverse con otras técnicas más exactas. Debido a su naturaleza aleatoria, los algoritmos evolutivos no garantizan encontrar soluciones óptimas para los problemas, pero en muchos casos pueden encontrar, si existen, buenas soluciones.

Los algoritmos evolutivos en la práctica resultan particularmente adecuados para resolver problemas de optimización multiobjetivo, debido a que consideran simultáneamente varias soluciones [39]. Esta característica les da la oportunidad de encontrar varias soluciones no dominadas en una sola corrida. Además, los algoritmos evolutivos son menos susceptibles a la forma y la continuidad del frente de Pareto; posibilitándole lidiar con frentes de Pareto cóncavos o discontinuos [37].

Existen dos prerrequisitos para utilizar algoritmos evolutivos en un problema: 1) que sea posible codificar las soluciones candidatas del problema para poder realizar las operaciones de cruzamiento y mutación; para la evaluación se necesita decodificar la solución y 2) debe ser posible evaluar las soluciones parciales del problema de manera cuantitativa con el objetivo de guiar el proceso evolutivo.

1.5.2. **Algoritmos meméticos**

Los algoritmos meméticos (MA, por sus siglas en inglés) están inspirados en la interacción de evolución genética y la evolución memética. El Darwinismo universal es la generalización de la evolución genética a cualquier sistema donde unidades discretas de información pueden ser heredadas y están sujetas a la selección y la variación. El término *meme* se refiere a esas unidades discretas de información cultural propuestas por la relación de la evolución genética y cultural. [40]

De manera general, los MA aplican una técnica de búsqueda global para identificar áreas prometedoras en el espacio de búsqueda y utilizan repetidamente una búsqueda local

sobre soluciones individuales para encontrar óptimos locales. En teoría, los MA reúnen los aspectos positivos de la evolución genética y cultural; permitiendo la transmisión, selección, herencia y variación tanto de los *memes* como de los genes. En la Figura 6 se muestra el esquema general de un MA.[41]

Con relación a la resolución de MOPs, los MA presentan características similares a los algoritmos evolutivos. La ventaja fundamental de los MA sobre los algoritmos evolutivos es que por lo general requieren menos tiempo de ejecución para explorar el espacio de búsqueda. Además, en la búsqueda local se pueden incorporar heurísticas específicas para el problema a resolver. Por otro lado, existen aspectos que no han sido suficientemente estudiados para los MA como la frecuencia con que debe aplicarse la búsqueda local, si se debe aplicar sobre todos los individuos o la relación que existe entre las posibles variantes para el cruzamiento y la mutación y la búsqueda local utilizada. [37]

```
Input: ProblemSize, Pop_size, MemePop_size
Output: S_best
Population ← InitializePopulation(ProblemSize, Pop_size);
while ¬StopCondition() do
    foreach S_i ∈ Population do
        Si_cost ← Cost(S_i);
    end
    S_best ← GetBestSolution(Population);
    Population ← StochasticGlobalSearch(Population);
    MemeticPopulation ← SelectMemeticPopulation(Population,
    MemePop_size);
    foreach S_i ∈ MemeticPopulation do
        S_i ← LocalSearch(S_i);
    end
end
return S_best;
```

Figura 6. Esquema general de un MA. [37]

1.5.3. **Algoritmos de aproximación**

Según [42] un algoritmo A es un algoritmo de aproximación para un problema de optimización Π si dada una instancia de entrada I, este calcula una solución factible S para cada entrada I. La calidad de estos algoritmos está dada por proximidad de esta solución S a la solución real; para ello se utiliza una métrica teórica estándar denominada coeficiente de aproximación. El coeficiente de aproximación se define formalmente como $C_a = min\left\{\frac{V(A(I))}{OPT(I)}, \frac{OPT(I)}{V(A(I))}\right\}$, donde $V(A(I))$ denota el valor obtenido al evaluar la solución encontrada por el algoritmo A en la función objetivo y $OPT(I)$ denota el valor óptimo de

la función objetivo para la instancia I. Otra definición importante es el factor de garantía que puede ofrecer el algoritmo A para el problema Π, expresado como $r_a = min\{C_a | \, para \, toda \, I \, de \, \Pi\}$.

Los algoritmos de aproximación tienen la ventaja de garantizar, con un tiempo de ejecución polinomial, la calidad de la solución encontrada con un coeficiente de aproximación específico en el peor de los casos. Son, por tanto, muy utilizados para la resolución de problemas de optimización complejos en los que una relajación específica de la calidad de la solución es aceptable.

Esta técnica no tiene un carácter general de aplicación, ya que utiliza características específicas del problema para su resolución. Por tanto, hay muchos problemas para los que no se ha demostrado la existencia de un coeficiente de aproximación. Otra característica de esta técnica, relevante para esta investigación, es que no se utiliza usualmente para problemas de optimización multiobjetivo por la complejidad que esto presupone.

1.5.4. **Optimización de enjambres de partículas**

Optimización por enjambre de partículas (PSO, por sus siglas en inglés) está inspirado en el movimiento en grupos de animales que buscan alimentos, como los pájaros o los peces. Estos grupos se mueven por el entorno de búsqueda siguiendo a un miembro más adelantado y generalmente reajustan su movimiento hacia ciertas zonas que históricamente les han dado buenos resultados. [43]

Simulando el movimiento del grupo, las posiciones de los individuos son asignadas aleatoriamente de manera inicial con poca velocidad. La dirección de cada individuo está determinada por su velocidad, la posición del mejor individuo en su vecindad y la mejor posición global conocida hasta el momento. Luego de actualizar cada posición, cada individuo es evaluado de acuerdo a la función objetivo. Con el tiempo, mediante la exploración y explotación de las mejores posiciones en el espacio de búsqueda, los individuos tienden a agruparse o converger alrededor de un área óptima. Nótese que existen puntos en común entre esta técnica y los algoritmos evolutivos como los conceptos de población, función objetivo (*fitness*), cruzamiento (con el ajuste de los individuos) y mutación (turbulencias). [41]

Aunque la versión original de PSO no contemplaba la resolución de MOPs, recientemente se han propuesto algunas versiones diseñadas específicamente para

este tipo de problemas. En [44] aparece una lista detallada de estas versiones y sus resultados. Las principales ventajas de PSO en la resolución de MOPs son su simplicidad (tanto en la teoría como en la implementación) y su eficiencia. De ello, algunos autores como [45] lo proponen como soluciones ultra eficientes para MOPs.

Dentro de las desventajas de PSO frente a los MOPs se encuentra su aparente dificultad para mantener la diversidad y evitar la convergencia. Existen, además, pocas investigaciones sobre la influencia que tienen distintas configuraciones sobre el comportamiento de esta técnica. [37]

```
   Input: ProblemSize, Population_size
   Output: P_g_best
 1 Population ← ∅;
 2 P_g_best ← ∅;
 3 for i = 1 to Population_size do
 4 |   P_velocity ← RandomVelocity();
 5 |   P_position ← RandomPosition(Population_size);
 6 |   P_cost ← Cost(P_position);
 7 |   P_p_best ← P_position;
 8 |   if P_cost ≤ P_g_best then
 9 |   |   P_g_best ← P_p_best;
10 |   end
11 end
12 while ¬StopCondition() do
13 |   foreach P ∈ Population do
14 |   |   P_velocity ← UpdateVelocity(P_velocity, P_g_best, P_p_best);
15 |   |   P_position ← UpdatePosition(P_position, P_velocity);
16 |   |   P_cost ← Cost(P_position);
17 |   |   if P_cost ≤ P_p_best then
18 |   |   |   P_p_best ← P_position;
19 |   |   |   if P_cost ≤ P_g_best then
20 |   |   |   |   P_g_best ← P_p_best;
21 |   |   |   end
22 |   |   end
23 |   end
24 end
25 return P_g_best;
```

Figura 7. Esquema general del PSO. [37]

1.5.5. **Recocido simulado**

Recocido simulado (SA, por sus siglas en inglés) está inspirado en el proceso metalúrgico para el fortalecimiento de metales (*annealing*). En este procedimiento se calienta el metal y luego se va enfriando poco a poco; aumentando el tamaño de los cristales en el metal y reduciendo sus imperfecciones. El calor inicial aumenta la energía

de los átomos permitiéndoles moverse libremente; luego, mediante un enfriamiento regulado, surgen nuevas configuraciones en el metal con características específicas. A medida que el metal se va enfriando se hace más estricto el criterio para la aceptación de una nueva configuración; de manera que cuando el metal se enfría mantiene una configuración óptima o cercana a la óptima. [46]

```
Input: ProblemSize, iterations_max, temp_max
Output: S_best
S_current ← CreateInitialSolution(ProblemSize);
S_best ← S_current;
for i = 1 to iterations_max do
    S_i ← CreateNeighborSolution(S_current);
    temp_curr ← CalculateTemperature(i, temp_max);
    if Cost(S_i) ≤ Cost(S_current) then
        S_current ← S_i;
        if Cost(S_i) ≤ Cost(S_best) then
            S_best ← S_i;
        end
    else if Exp( (Cost(S_current) − Cost(S_i)) / temp_curr ) > Rand() then
        S_current ← S_i;
    end
end
return S_best;
```

Figura 8. Esquema general del algoritmo SA. [37]

SA funciona, básicamente, como una simulación del proceso metalúrgico descrito anteriormente; donde cada posible configuración representa una solución en el espacio de búsqueda del problema. Cada perturbación de la temperatura provoca una nueva configuración que se evalúa con relación a las funciones objetivo; si la nueva configuración es mejor (o pertenece al frente de Pareto) es aceptada, si no, puede ser aceptada con una probabilidad que es inversamente proporcional a la temperatura. En la Figura 8 se muestra el esquema general de un SA.

Aunque existe un grupo de variaciones del SA original específicas para MOPs, esta técnica no presenta ventajas importantes para su resolución. La más llamativa está relacionada con la idea de que en SA está demostrado teóricamente que son capaces de encontrar las soluciones pertenecientes al frente de Pareto, sin embargo estos mismos análisis plantean que en el peor de los casos el número de evaluaciones pudiera ser mayor que el número de soluciones en el espacio de búsqueda. Dentro de sus características negativas para la resolución de MOPs, en [37] se señalan la dificultad de encontrar un cronograma de enfriamiento (*cooling schedule*) adecuado para el problema

a resolver y la complejidad de definir una estrategia que mantenga la diversidad y el frente de Pareto al mismo tiempo.

1.6. Frameworks para el desarrollo de algoritmos de optimización multiobjetivo

Implementar y comparar un algoritmo de optimización multiobjetivo es una tarea compleja. En consecuencia, es usual utilizar un *framework* que permita la reutilización de componentes genéricos y sirva de marco de pruebas en la resolución del problema planteado. A continuación se analizan varias opciones liberadas sin restricciones legales con el objetivo de seleccionar un *framework* adecuado para esta investigación.

1.6.1. JMetal

Desarrollado por el Departamento de Ciencias de la Computación de la Universidad de Málaga (España), este *framework* está disponible desde el 2006 para la comunidad internacional. JMetal está desarrollado en *Java* con un diseño orientado a objetos e implementa algoritmos basados en diferentes meta-heurísticas entre las que se destacan las estrategias evolutivas y las híbridas [47]. En [48] se puede consultar la lista de algoritmos disponibles, así como las posibles codificaciones que se pueden utilizar para representar las variables de decisión.

Con relación a los experimentos, jMetal incluye funcionalidades para ejecución en paralelo de algoritmos, así como para la comparación y análisis estadístico de los resultados. Algunos de los indicadores implementados para la comparación de los algoritmos son el Hipervolumen, la Distancia generacional, Épsilon y la Dispersión [48].

La versión 4.5 (última versión) fue liberada en enero del 2014 y los plazos de entrega son generalmente anuales; esto ratifica que el equipo de desarrollo está activo y el *framework* debe tener soporte al menos en un futuro cercano. La documentación y los ejemplos disponibles están actualizados y tienen un nivel técnico adecuado [49].

1.6.2. ParadisEO

ParadisEO es un *framework* desarrollado por el Instituto Nacional de Investigaciones de Informática y Automática de Francia con más de 8 años de existencia. Los algoritmos implementados se basan en dos meta-heurísticas: algoritmos evolutivos y búsquedas locales; también están disponibles versiones híbridas, paralelas y distribuidas de los algoritmos propuestos [50]. En [51] se puede consultar la lista de algoritmos disponibles. Llama la atención la amplia documentación con ejemplos y ejercicios disponibles así

como la existencia de una comunidad con correo de contacto para preguntas y soporte. Utiliza *C++* como lenguaje de programación.

1.6.3. **Opt4J**

Opt4J está desarrollado por el Departamento de Ciencias de la Computación de la Universidad de Erlangen-Nuremberg (Alemania). Desarrollado en *Java*, *Opt4J* se destaca por el amplio grupo de meta-heurísticas utilizadas como base para sus algoritmos (Algoritmos Evolutivos, Recocido Simulado, Optimización de Enjambres de Partículas, Evolución Diferencial, entre otras) y por su interfaz gráfica para la configuración y representación de los experimentos [52]. El código fuente contiene además los *javadocs* donde aparece información sobre los paquetes, las clases y las interfaces implementadas. A consideración del autor de la presente investigación, la documentación disponible (al menos en inglés) en algunos temas no es lo suficientemente profunda y los ejemplos propuestos no son de mucha utilidad en casos no triviales.

1.7. **Conclusiones del capítulo**

A partir de los elementos expuestos anteriormente se arriban a las siguientes conclusiones:

- Las WSANs presentan varios puntos en común con las WSNs. Sus principales particularidades están en la naturaleza de los dispositivos y en las conexiones necesarias para la coordinación.
- La mayoría de las soluciones similares utilizan algoritmos evolutivos y modelan las instancias del problema como grafos.
- Modelar las instancias del problema de esta investigación como grafos permite utilizar los elementos de teoría de grafos para el cálculo de los indicadores con que se estiman los criterios de optimización y las restricciones.
- El número de vértices puede utilizarse como indicador del número de enrutadores.
- El grado máximo del grafo puede utilizarse como indicador del consumo de energía por concepto de transmisión/recepción.
- La cantidad de caminos de vértices disjuntos entre los vértices puede utilizarse como indicador para la tolerancia a fallos.

- Para demostrar la presencia de los dispositivos terminales y el coordinador en una posible solución, debe probarse la presencia de los vértices que representan a los dispositivos terminales y el coordinador entre los vértices de la posible solución.
- Para demostrar la conectividad de una posible solución, basta con demostrar la existencia de al menos un camino entre cada par de vértices de la posible solución.
- El problema planteado en esta investigación se clasifica, de acuerdo a su complejidad computacional, como NP-Hard.
- Si bien es imposible seleccionar *a priori* una técnica por encima de otra, teniendo en cuenta la complejidad computacional del problema y el análisis realizado sobre las técnicas utilizadas en problemas similares, se considera que los algoritmos evolutivos y meméticos pueden encontrar soluciones no dominadas cercanas al frente de Pareto real en este caso.
- Teniendo en cuenta la documentación, los algoritmos implementados y las facilidades que brinda para la experimentación, el *framework* jMetal es el que más se ajusta a las necesidades de la investigación.

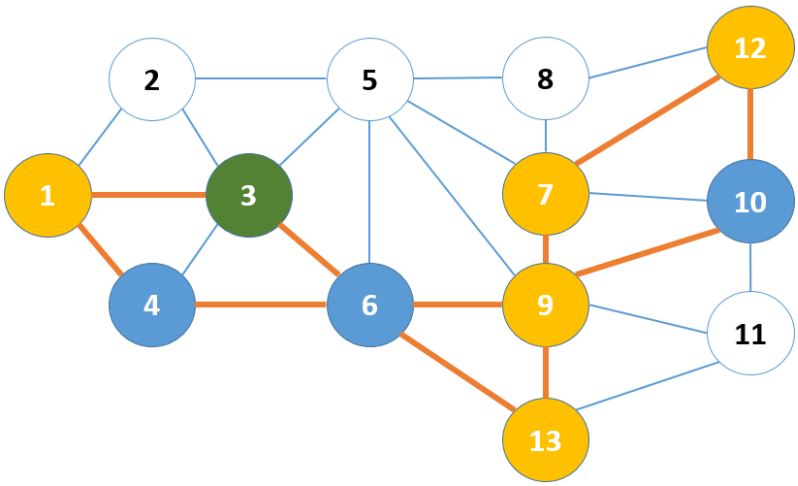
Capítulo 2. Propuesta de solución

En este capítulo se explica la estrategia utilizada para la representación del problema y los algoritmos para el cálculo de los indicadores definidos para las funciones objetivo y las restricciones. Además, se describen cinco algoritmos de optimización multiobjetivo propuestos para la identificación de soluciones no dominadas. Se analizan también los operadores evolutivos utilizados en los problemas propuestos.

2.1. Modelación del problema

En el contexto de los algoritmos evolutivos, la modelación del problema es un paso trascendental para el adecuado funcionamiento de los mismos. En la modelación se garantizan los dos prerrequisitos asociados a la utilización de algoritmos evolutivos: 1) las posibles soluciones deben ser codificadas para poder aplicar los operadores de cruzamiento y mutación y 2) las posibles soluciones deben ser evaluables para guiar el proceso de búsqueda.

Como se especifica en la definición formal del problema, el grafo G representa la WSAN resultante de la Generación de posiciones candidatas de los enrutadores y cada subgrafo inducido G' constituye una posible solución al problema. Para codificar las posibles soluciones se utilizó en este caso una codificación binaria conformada a partir de la lista de aristas del grafo G; donde el valor de cada bit de la cadena de codificación especifica si la arista está presente (1) o no (0) en el individuo representado.



Func. Objetivo	Valor
$\|V'\|$	9
$\Delta(G')$	4
$k-connected(G')$	1

L1	v1	v1	v1	v2	v2	v3	v3	v3	v4	v5	v5	v5	v5	v6	v6	v7	v7	v7	v7	v8	v9	v9	v9	v10	v10	v11
L2	v2	v3	v4	v3	v5	v4	v5	v6	v6	v6	v7	v8	v9	v9	v13	v8	v9	v10	v12	v12	v10	v11	v13	v11	v12	v13
Ind.	0	1	1	0	0	0	0	1	1	0	0	0	0	1	1	0	1	0	1	0	1	0	1	0	1	0

Figura 9. Codificación de un individuo.

La codificación binaria es una de las más utilizadas y estudiadas para la resolución de problemas mediante algoritmos evolutivos [37]. Utilizando la lista de aristas para representar los grafos G y G', la codificación binaria resulta natural para representar las posibles soluciones. Utilizando esta codificación, el espacio de búsqueda crece de manera proporcional al número de aristas en G. Cada elemento del espacio de búsqueda es un subconjunto del conjunto de aristas en G, por tanto, el tamaño del espacio de búsqueda es igual al número de elementos en el conjunto potencia del conjunto de aristas en G. Por ejemplo, para el grafo de la WSAN propuesta en la Figura 9 el número de posibles soluciones sería $2^{26} = 67108864$.

Para calcular el indicador **número de vértices**, asociado al criterio de optimización número de enrutadores, es necesario identificar los vértices presentes en el individuo. En la Figura 10 se propone un algoritmo para el cálculo de este indicador de acuerdo a la codificación seleccionada. El algoritmo propuesto tiene una complejidad $O(ms)$ en el peor de los casos, siendo m la cantidad de aristas en G y s la cantidad de vértices en G'. Sin embargo, la utilización de un diccionario (*hash*) para almacenar los vértices encontrados, puede disminuir la complejidad promedio del algoritmo; ya que en la práctica las operaciones sobre diccionarios son por lo general eficientes ($O(1)$ amortizado) [53]. El diccionario de vértices con sus grados será utilizado para el cálculo de otros criterios.

```
getVertexes ():
    hashMap_vertexes ← HashMap<vertex, degree>
    for i ← 0 to m do
        if individual[i] > 0 then:
            vertex1 ← VertexesList1[i]
            vertex2 ← VertexesList2[i]
            if hashMap_vertexes.containsKey(vertex1) then:
                current_degree ← hashMap_vertexes.get(vertex1)
                hashMap_vertexes.put(vertex1, current_degree + 1)
            else:
                hashMap_vertexes.put(vertex1, 1)
            end
            if hashMap_vertexes.containsKey(vertex2) then:
                current_degree ← hashMap_vertexes.get(vertex2)
                hashMap_vertexes.put(vertex2, current_degree + 1)
            else:
                hashMap_vertexes.put(vertex2, 1)
            end
        end
    end
    return hashMap_vertexes
```

Figura 10. Pseudocódigo para el cálculo del número de vértices.

Por otro lado, en el problema se utiliza el **grado máximo** de G' para la estimación del consumo de energía por concepto de transmisión/recepción. Para ello se puede utilizar el siguiente algoritmo de complejidad $O(s^2)$ en el peor de los casos, siendo s la cantidad de vértices en G'.

```
getMaximumDegree (hashMap_vertexes):
    maximumDegree ← 0
    vertexes ← hashMap_vertexes.keySet
    s ← vertexes.size
    for i ← 0 to s do:
        currentVertex ← vertexes[i]
        currentDegree ← hashMap_vertexes.get(currentVertex)
        if currentDegree > maximumDegree then:
            maximumDegree ← currentDegree
        end
    end
    return maximumDegree
```

Figura 11. Pseudocódigo para el cálculo del grado máximo.

Con relación al indicador de la tolerancia a fallos, el **número de caminos de vértices disjuntos**, se utilizó el algoritmo propuesto por Even y Tarjan en [54]. Este algoritmo utiliza como subrutina el cálculo del flujo máximo, para lo cual se empleó el algoritmo de Edmonds-Karp, con complejidad $O(sd^2)$; siendo s el número de vértices y d el número de aristas en G' [55]. El algoritmo propuesto por Even y Tarjan tiene una complejidad

$O(k(s - \delta - 1))$ [54]; donde k representa el número de llamadas al algoritmo para el cálculo del flujo máximo, s el número de vértices en G' y δ el grado mínimo de G'.

```
get_k_connected (hashMap_vertexes):
    i ← 0
    vertexes ← hashMap_vertexes.keySet
    s ← vertexes.size
    N ← (s > 0) ? s-1 : 0
    while i <= N :
        for j ← i + 1 to s do:
            vi ← vertexes[i]
            vj ← vertexes[j]
            if are_not_neighbors(vi, vj) then:
                current_k ← max_flow(vi, vj)
                N ← min(N, current_k)
                if current_k = 0 then:
                    k_connected ← N
                    return k_connected
                end
            end
        end
        i++
    end
    k_connected ← N
    return k_connected
```

Figura 12. Pseudocódigo para el cálculo del número de caminos de vértices disjuntos.

Para comprobar la **presencia de los dispositivos terminales y del coordinador** se utilizó el algoritmo representado en Figura 13 de complejidad $O(es)$ en el peor de los casos; siendo e la cantidad de dispositivos terminales más el coordinador y s la cantidad de vértices en G'. En el caso de la **conectividad**, una vez calculado el número de caminos de vértices disjuntos como indicador de la tolerancia a fallos, si existe al menos un camino de vértices disjuntos en G' entre cada par de vértices es posible afirmar que el subgrafo G' es conexo.

```
check_presence_restriction(end_devices_and_controller, hashMap_vertexes):
    e ← end_devices_and_controller.size
    for i ← 0 to e do:
        current ← end_devices_and_controller[i]
        if not hashMap_vertexes.containsKey(current) then:
            return false
        end
    end
    return true
```

Figura 13. Pseudocódigo para la comprobación de la presencia de los dispositivos terminales y el coordinador.

En los algoritmos evolutivos y meméticos la evaluación se repite en cada generación para determinar la selección de los individuos mejor adaptados. Por tanto, es recomendable mantener la complejidad de esta actividad lo más bajo posible. Utilizando la modelación propuesta, la evaluación de los indicadores de los criterios de optimización y las restricciones tiene en este caso una complejidad $O(sd^2(s - \delta - 1))$; siendo s el número de vértices en G', d la cantidad de aristas en G' y δ el grado mínimo de G'.

2.2. Algoritmos de optimización multiobjetivo propuestos

Para la identificación de las soluciones no dominadas se proponen cuatro algoritmos evolutivos: NSGA-II, SPEA2, PAES, PESA-II y un algoritmo memético: mNSGA-II. Con excepción del mNSGA-II, estos algoritmos son recomendados en [37] por sus resultados demostrados en la resolución de MOPs de manera general. Todos los algoritmos propuestos utilizan, de una manera u otra, el concepto de dominancia de Pareto para determinar la probabilidad de reproducción de los individuos. A continuación se detallan algunas de sus características y sus pseudocódigos.

Nondominated Sorting Genetic Algorithm (NSGA-II)

En el caso de la selección, en este algoritmo inicialmente la población es dividida y ordenada de acuerdo a su nivel de dominancia. Se calcula entonces la distancia grupal (*crowding distance*) entre los individuos y se seleccionan los no dominados que mantienen cierta *distancia* entre ellos; esto garantiza la diversidad de la población. Los individuos mejor adaptados pasan a la siguiente generación (elitismo), garantizando que se reproduzcan las mejores características. [56]

```
1: procedure NSGA-II(𝒩′, g, f_k(x_k))    ▷ 𝒩′ members evolved g generations to solve f_k(x)
2:     Initialize Population ℙ′
3:     Generate random population - size 𝒩′
4:     Evaluate Objective Values
5:     Assign Rank (level) Based on Pareto dominance - sort
6:     Generate Child Population
7:       Binary Tournament Selection
8:       Recombination and Mutation
9:     for i = 1 to g do
10:        for each Parent and Child in Population do
11:            Assign Rank (level) based on Pareto - sort
12:            Generate sets of nondominated vectors along PF_known
13:            Loop (inside) by adding solutions to next generation starting from the first front until 𝒩′ individuals found determine crowding distance between points on each front
14:        end for
15:        Select points (elitist) on the lower front (with lower rank) and are outside a crowding distance
16:        Create next generation
17:            Binary Tournament Selection
18:            Recombination and Mutation
19:    end for
20: end procedure
```

Figura 14. Pseudocódigo del NSGA-II. [37]

Strength Pareto Evolutionary Algorithm (SPEA2)

Dentro de las características representativas de este algoritmo está la utilización de un archivo externo para mantener las soluciones no dominadas encontradas. Por cada individuo no dominado se calcula el “grado” de su dominancia en base a la cantidad de soluciones que domina y a la cantidad de soluciones que lo dominan; este indicador determina su probabilidad de pasar a la siguiente generación. El tamaño del archivo donde se mantienen las soluciones no dominadas debe ser controlado en cada generación para mantener la calidad del proceso de selección y no afectar la eficiencia de la búsqueda. [57]

```
1: procedure SPEA2(𝒩′, g, f_k(x))
2:     Initialize Population ℙ′
3:     Create empty external set 𝔼′
4:     for i=1 to g do
5:         Compute fitness of each individual in ℙ′ and 𝔼′
6:         Copy all individual evaluating to nondominated vectors ℙ′ and 𝔼′ to 𝔼′
7:         Use the truncation operator to remove elements from E when the capacity
   of the file has been extended
8:         If the capacity of 𝔼′ has not been exceeded then use dominated individuals
   in ℙ′ to fill 𝔼′
9:         Perform binary tournament selection with replacement to fill the mating
   pool
10:        Apply crossover and mutation to the mating pool
11:    end for
12: end procedure
```

Figura 15. Pseudocódigo del SPEA2. [37]

Pareto Archived Evolution Strategy (PAES)

Un rasgo distintivo de este algoritmo es que no utiliza operadores de cruzamiento; utiliza una estrategia evolutiva basada en la mutación de un único individuo que conforma la población. Los elementos no dominados encontrados son almacenados en un archivo externo. Para la selección del nuevo individuo de la población, PAES propone un sistema basado en la distancia entre las soluciones que garantiza la diversidad. [58]

```
1: procedure PAES(f_k(x))
2:     repeat
3:         Initialize Single Population parent, 𝒞, and add to archive, 𝔸
4:         Mutate 𝒞 to produce child 𝒞′ and evaluate fitness
5:         if 𝒞 ≻ 𝒞′ then
6:             discard 𝒞′
7:         else if 𝒞 ≻ 𝒞′ then
8:             replace 𝒞 with 𝒞′, and add 𝒞 to 𝔸
9:         else if ∃_{𝒞"∈𝔸}(𝒞" ≻ 𝒞′) then
10:            discard 𝒞′
11:        else
12:            apply test (𝒞, 𝒞′, 𝔸) to determine which becomes the new current so-
   lution and whether to add 𝒞′ to 𝔸
13:        end if
14:    until termination criteria is met
15: end procedure
```

Figura 16. Pseudocódigo del PAES. [37]

Pareto Envelope-based Selection Algorithm (PESA-II)

El diseño de este algoritmo incorpora ideas del SPEA2 y el PAES como el almacenamiento de las soluciones no dominadas en un archivo externo y el uso de la

distancia grupal como indicador de las similitudes entre los individuos. La selección de los individuos para la reproducción está basada en regiones, es decir, se selecciona la región mejor adaptada y se escogen individuos dentro de esta región. [59]

```
1: procedure PESA(𝒩′, f_k(x))
2:     Initialize Population ℙ′_i of size 𝒩′ Randomly
3:     Evaluate each member of ℙ′_i
4:     Initialize the external population ℙ′_e to the empty set
5:     repeat
6:         Incorporate individuals evaluating to nondominated vectors from ℙ′_i into
   ℙ′_e
7:         Delete the current contents of ℙ′_i
8:         repeat
9:             With probability p_c, select two parents from ℙ′_e          ▷ p_c is the
   probability of crossover
10:            Produce a single child via crossover
11:            Mutate the child created in the previous step
12:            With probability (1 − p_c), select one parent
13:            Mutate the selected parent to produce a child
14:        until ℙ′_i is filled
15:    until termination criteria is met
16:    Return(ℙ′_e)                    ▷ Return the members of ℙ′_e as the result
17: end procedure
```

Figura 17. Pseudocódigo del PESA-II. [37]

Memetic Nondominated Sorting Genetic Algorithm (mNSGA-II)

El algoritmo mNSGA-II fue diseñado específicamente para esta investigación basado en el NSGA-II. Como algoritmo memético, mNSGA-II propone la utilización de una búsqueda local para la exploración de regiones limitadas dentro del espacio de búsqueda; en este caso basada en el Escalador de colinas estocástico (*Stochastic Hill Climbing*) [41]. La Figura 18 muestra el pseudocódigo utilizado para la búsqueda local. Otra característica de este algoritmo es que no aplica la búsqueda local sobre individuos que violan más del 50% de las restricciones especificadas en el problema. La Figura 19 muestra el pseudocódigo del algoritmo mNSGA-II.

```
Input: Iter_max, ProblemSize
Output: Current
Current ← RandomSolution(ProblemSize);
foreach iter_i ∈ Iter_max do
    Candidate ← RandomNeighbor(Current);
    if Cost(Candidate) ≥ Cost(Current) then
        Current ← Candidate;
    end
end
return Current;
```

Figura 18. Pseudocódigo del Escalador de Colinas Estocástico. Utilizado para la búsqueda local en el mNSGA-II. [41]

```
procedure mNSGAII(N', g, f_k(x_k))    ▷ N' members evolved g generations to solve f_k(x)
    Initialize Population P'
    Generate random population - size N'
    Evaluate Objective Values
    Assign Rank (level) Based on Pareto dominance - sort
    Generate Child Population
      Binary Tournament Selection
      Recombination and Mutation
    for i = 1 to g do
        for each Parent and Child in Population do
            Assign Rank (level) based on Pareto - sort
            Generate sets of nondominated vectors along PF_known
            Loop (inside) by adding solutions to next generation starting from the first front until N' individuals found determine crowding distance between points on each front
        end for
        Select points (elitist) on the lower front (with lower rank) and are outside a crowding distance
        Create next generation
            Binary Tournament Selection
            Recombination and Mutation
            Apply local search except for individues who violates more
                        than 50% of constrains
    end for
end procedure
```

Figura 19. Pseudocódigo del mNSGA-II.

2.3. **Operadores evolutivos**

La selección de los operadores evolutivos es un paso trascendental para los algoritmos evolutivos y meméticos. De manera general, esta selección está muy ligada con la

codificación de las posibles soluciones utilizada en la modelación del problema. En este caso se proponen operadores de cruzamiento y mutación ampliamente utilizados para problemas con codificación binaria. Con el objetivo de que la selección de los operadores no represente una diferencia entre los algoritmos propuestos, todos utilizan los mismos operadores.

Luego del proceso de selección específico de cada uno de los algoritmos propuestos, se aplican operadores de cruzamiento y/o mutación para la obtención de nuevos individuos. Con excepción del PAES, que no utiliza cruzamiento, para todos los algoritmos propuestos se utilizó un operador de cruzamiento denominado Cruzamiento de punto único (*Single Point Crossover*) [37]. Dado un punto de cruzamiento aleatorio, este operador crea dos nuevos individuos a partir de fragmentos de dos individuos seleccionados. Ver la Figura 20 para más detalles.

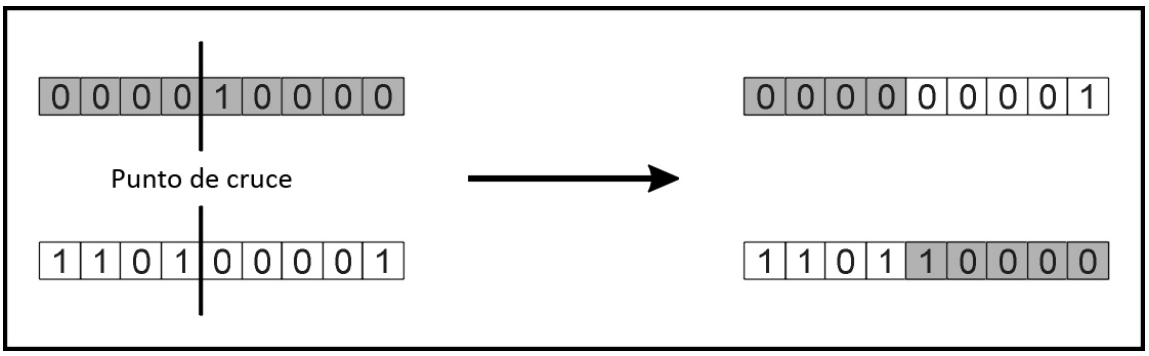


Figura 20. Cruzamiento de punto único.

Con relación a la mutación, en todos los algoritmos propuestos se utilizó el operador Mutación aleatoria bit a bit (*Flip Bit Mutation*) [37]. Por cada posición de la cadena binaria que representa al individuo, este operador genera un número aleatorio que es comparado con la probabilidad de mutación; si el número aleatorio es menor que la probabilidad de mutación cambia el valor de la cadena binaria en la posición analizada. Ver ejemplo en la Figura 21.

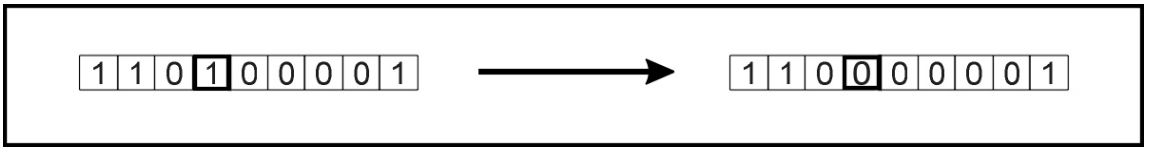


Figura 21. Mutación aleatoria bit a bit.

2.4. **Conclusiones del capítulo**

Como parte de la modelación del problema, en este capítulo se presentó una codificación binaria basada en la lista de aristas del grafo que representa la WSAN resultante de la Generación de posiciones candidatas de los enrutadores. Se definieron también los algoritmos utilizados para el cálculo de los indicadores de los criterios de optimización y las restricciones; demostrándose que, de manera general, la evaluación de cada posible solución tiene una complejidad $O(sd^2(s - \delta - 1))$.

Además, se propusieron cinco algoritmos de optimización multiobjetivo para la identificación de soluciones no dominadas: cuatro algoritmos evolutivos y un algoritmo memético diseñado específicamente como parte de esta investigación; teóricamente todos son capaces de encontrar soluciones no dominadas cercanas al frente de Pareto real. Para que no represente una diferencia, todos los algoritmos propuestos utilizan los mismos operadores evolutivos.

Capítulo 3. Selección y validación del algoritmo

En este capítulo se describe el diseño de los experimentos realizados para la selección del algoritmo de optimización multiobjetivo y para la validación del algoritmo seleccionado. Se definen también los indicadores utilizados para evaluar la calidad de las soluciones no dominadas identificadas por los algoritmos en los experimentos.

Además, se describen los resultados obtenidos en los experimentos para la selección del algoritmo de optimización multiobjetivo y se aplica el test de Wilcoxon para determinar si existen diferencias estadísticamente significativas entre los resultados. Una vez seleccionado el algoritmo, se valida su efectividad comparando los valores de los indicadores de los criterios de optimización y las restricciones antes y después de su aplicación.

3.1. Diseño de experimentos

Durante la investigación se diseñaron dos experimentos, uno relacionado con la selección del algoritmo de optimización multiobjetivo y otro con la validación de la efectividad del algoritmo seleccionado. Los dos experimentos utilizan la misma muestra: cuatro WSANs representadas como grafos, resultantes del proceso de Generación de posiciones candidatas de los enrutadores.

Tabla 5. Muestra utilizada para los experimentos.

	WSAN1	**WSAN2**	**WSAN3**	**WSAN4**
Número de vértices	19	63	88	93
Grado máximo	6	7	6	6
Número de caminos de vértices disjuntos	2	1	2	2
Presencia de los vértices que representan a los dispositivos terminales y el coordinador	Sí	Sí	Sí	Sí
Al menos un camino entre cada par de vértices	Sí	Sí	Sí	Sí

A continuación se analiza el procedimiento y las variables a considerar en ambos experimentos.

3.1.1. Selección del algoritmo de optimización multiobjetivo

Utilizando el *framework* jMetal v4.5 se implementó la modelación del problema planteada en el capítulo anterior y cada uno de los algoritmos de optimización multiobjetivo

propuestos. Cada algoritmo se ejecutó 30 veces frente a cada sujeto de la muestra para que el análisis estadístico contara con un grupo considerable de resultados.

Teniendo en cuenta que los resultados del experimento pueden variar considerablemente de acuerdo a la parametrización empleada, en los experimentos se utilizó una parametrización estándar propuesta en [37] y descrita en la Figura 22.

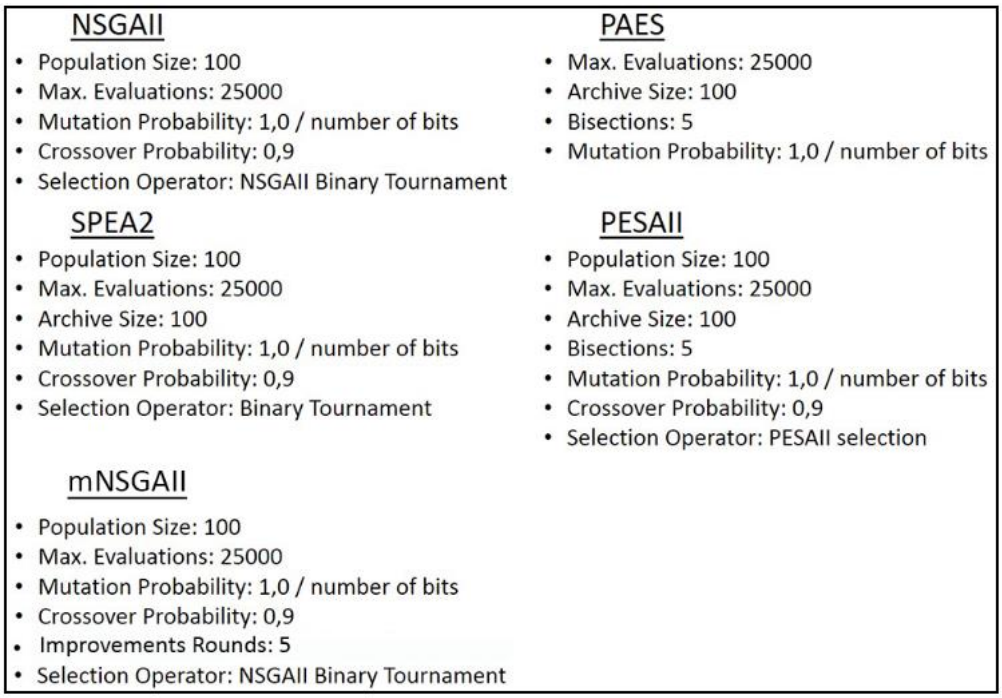


Figura 22. Parametrización de los algoritmos propuestos.

Con relación a las variables a considerar en el experimento, existen varios indicadores de calidad para la comparación de algoritmos de optimización multiobjetivo. De manera general, estos indicadores se dividen en dos clases, los consecuentes con el concepto de dominancia de Pareto (*Pareto compliant*) y los no consecuentes (*Pareto noncompliant*). Dado dos soluciones A, B y un indicador I, si A es no dominada por B, para todos los indicadores consecuentes con el concepto de dominancia de Pareto $I(A)$ obtiene mejores resultados que $I(B)$; en contraste, para los no consecuentes con el concepto de dominancia de Pareto, $I(B)$ puede obtener mejores resultados que $I(A)$. Una lista detallada de varios indicadores y su clasificación puede ser consultada en [37].

En el caso de este experimento se seleccionaron dos indicadores consecuentes con el concepto de dominancia de Pareto: Épsilon (ε) y Radio de error (ER, por sus siglas en inglés).

Épsilon

Dado dos conjuntos A y B, esta métrica determina las transformaciones que hay que realizar en A para que cada elemento en B sea dominado por al menos una solución de A. Si $\varepsilon(A,B) < 1$, todas las soluciones en B son dominadas por al menos una solución en A. Si $\varepsilon(A,B) = 1$ y $\varepsilon(B,A) = 1$, entonces A y B representan el mismo frente de Pareto. Si $\varepsilon(A,B) > 1$ y $\varepsilon(B,A) > 1$, entonces A y B son incomparables (los dos contienen soluciones no dominadas por el otro). [37]

Radio de error

Reporta el número de soluciones en el frente de Pareto a evaluar (PF_{known}) que no pertenecen al frente de Pareto real (PF_{true}). Matemáticamente se representa como $ER \triangleq \frac{\sum_{i=1}^{|PF_{known}|} e_i}{|PF_{known}|}$, donde e_i es cero cuando la i-ésima solución en PF_{known} es una solución del PF_{true} o 1 cuando no está. Si $ER = 0$, todas las soluciones en PF_{known} pertencen al PF_{true}; por otro lado, si $ER = 1$, ninguna de las soluciones en PF_{known} pertenecen al PF_{true}. Mientras menor sea el valor de ER mejor es el PF_{known}. [37]

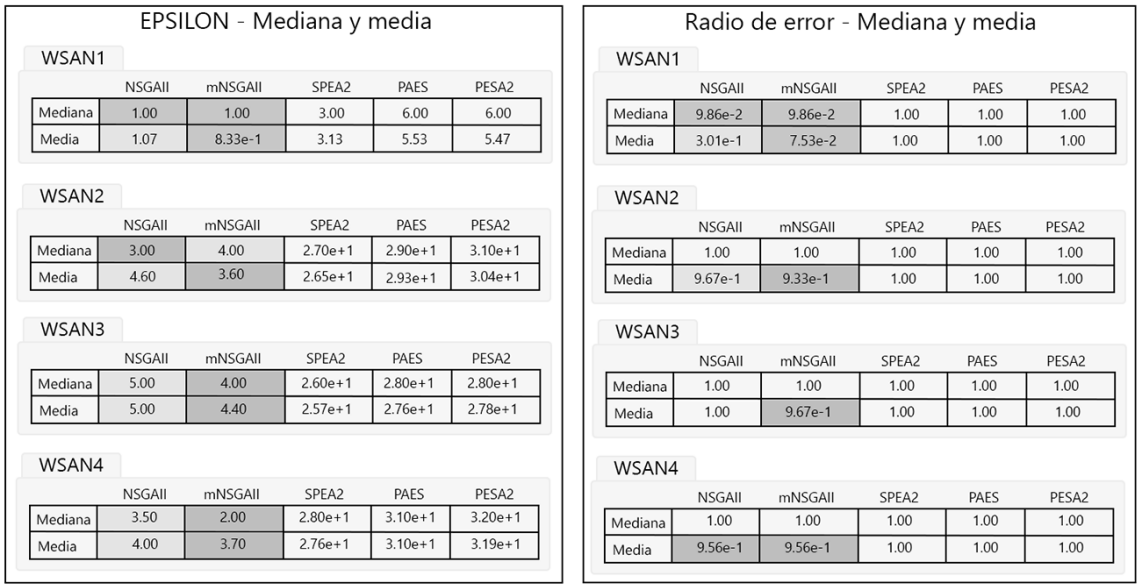
Para el cálculo del indicador ER es necesario conocer el frente de Pareto real (PF_{true}) de la instancia del problema. Identificar el PF_{true} mediante un método determinista, al menos con el hardware disponible para esta investigación, es computacionalmente imposible en la mayoría de los casos (el espacio de búsqueda se define como 2^r donde r sería el número de aristas en G). En este caso se utiliza un PF_{true} aproximado, identificado a partir de las soluciones no dominadas en todos los frentes de Pareto obtenidos por los algoritmos para una instancia del problema durante el experimento. Esta idea es sugerida en [49] para el cálculo de indicadores que necesitan el PF_{true}.

3.1.2. **Validación del algoritmo seleccionado**

Una vez seleccionado el algoritmo de optimización multiobjetivo mediante el experimento descrito con anterioridad, es necesario validar su efectividad frente a la muestra. En este experimento, el algoritmo seleccionado fue ejecutado 30 veces por cada sujeto de la muestra. Las variables consideradas en este experimento son los indicadores definidos para la estimación de los criterios de optimización y las restricciones: el número de vértices, el grado máximo, el número de caminos de vértices disjuntos, la presencia de los vértices que representan a los dispositivos terminales y al coordinador y la presencia de al menos un camino entre cada par de vértices.

3.2. Análisis de los resultados de la selección del algoritmo de optimización multiobjetivo

En la Figura 23 se presentan la mediana y la media de los resultados obtenidos al aplicar el experimento para la selección del algoritmo de optimización multiobjetivo. Los valores resaltados en gris oscuro representan los mejores resultados, los grises claros le siguen en calidad. Para los dos indicadores menor es mejor.



EPSILON - Mediana y media

WSAN1

	NSGAII	mNSGAII	SPEA2	PAES	PESA2
Mediana	1.00	1.00	3.00	6.00	6.00
Media	1.07	8.33e-1	3.13	5.53	5.47

WSAN2

	NSGAII	mNSGAII	SPEA2	PAES	PESA2
Mediana	3.00	4.00	2.70e+1	2.90e+1	3.10e+1
Media	4.60	3.60	2.65e+1	2.93e+1	3.04e+1

WSAN3

	NSGAII	mNSGAII	SPEA2	PAES	PESA2
Mediana	5.00	4.00	2.60e+1	2.80e+1	2.80e+1
Media	5.00	4.40	2.57e+1	2.76e+1	2.78e+1

WSAN4

	NSGAII	mNSGAII	SPEA2	PAES	PESA2
Mediana	3.50	2.00	2.80e+1	3.10e+1	3.20e+1
Media	4.00	3.70	2.76e+1	3.10e+1	3.19e+1

Radio de error - Mediana y media

WSAN1

	NSGAII	mNSGAII	SPEA2	PAES	PESA2
Mediana	9.86e-2	9.86e-2	1.00	1.00	1.00
Media	3.01e-1	7.53e-2	1.00	1.00	1.00

WSAN2

	NSGAII	mNSGAII	SPEA2	PAES	PESA2
Mediana	1.00	1.00	1.00	1.00	1.00
Media	9.67e-1	9.33e-1	1.00	1.00	1.00

WSAN3

	NSGAII	mNSGAII	SPEA2	PAES	PESA2
Mediana	1.00	1.00	1.00	1.00	1.00
Media	1.00	9.67e-1	1.00	1.00	1.00

WSAN4

	NSGAII	mNSGAII	SPEA2	PAES	PESA2
Mediana	1.00	1.00	1.00	1.00	1.00
Media	9.56e-1	9.56e-1	1.00	1.00	1.00

Figura 23. Mediana y media para los indicadores Épsilon y Radio de error.

De manera general, para los dos indicadores, los algoritmos NSGA-II y mNSGA-II muestran los mejores resultados. En el caso del indicador Épsilon, si bien la mediana arrojó que el mNSGA-II solo tuvo mejor comportamiento para los sujetos WSAN3 y WSAN4, la media indica que tuvo un mejor comportamiento para toda la muestra. Con respecto al indicador Radio de error, la mediana no muestra diferencias entre los algoritmos propuestos para los sujetos WSAN2, WSAN3 y WSAN4; por otro lado, la media indica mejores resultados para el mNSGA-II, excepto para el sujeto WSAN4, donde tiene el mismo desempeño que el NSGA-II. El comportamiento de la mediana para los sujetos WSAN2, WSAN3 y WSAN4 es consecuencia del aumento del espacio de búsqueda en estas instancias; a medida que disminuye la posibilidad de encontrar varias soluciones no dominadas válidas, la mediana para el indicador Radio de error tiende a ser la misma.

Para determinar si las diferencias obtenidas fueron estadísticamente significativas, se aplicó el test no paramétrico de rasgos con signo de Wilcoxon (W*ilcoxon Signed Rank*

Test) para los resultados de cada par de algoritmos. Los resultados se muestran en la Figura 24.

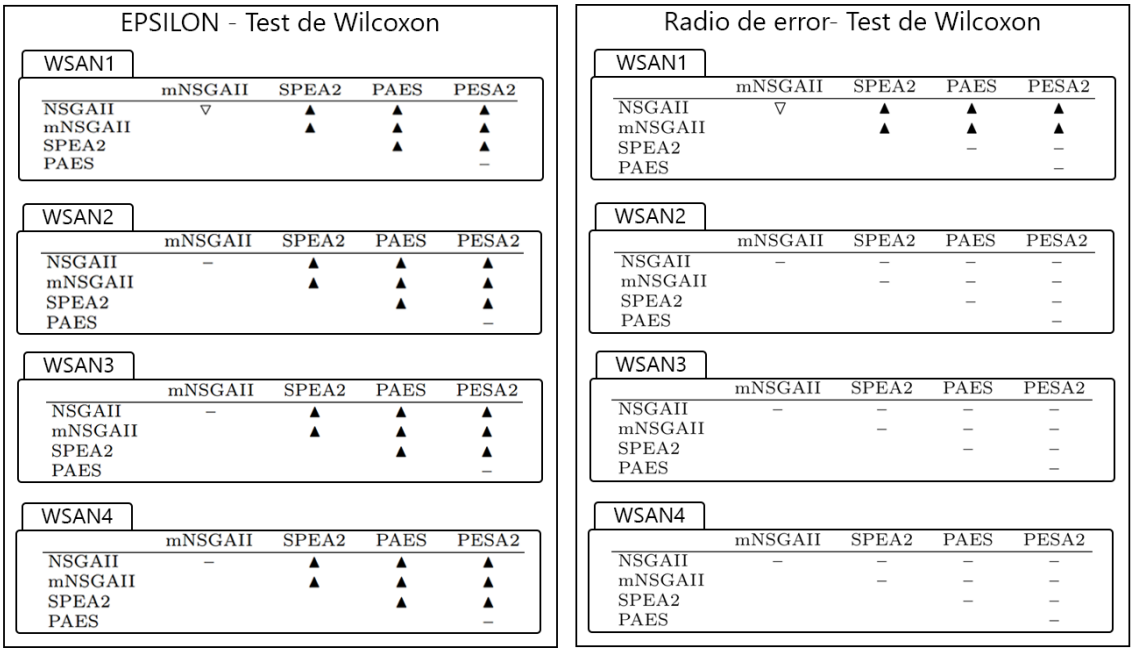


Figura 24. Resultados del test de Wilcoxon para los indicadores Épsilon y Radio de error.

En los resultados del test, el símbolo " ▲ " significa que el algoritmo de la fila mostró mejores resultados que el de la columna con una diferencia estadísticamente significativa, el símbolo " ▽ " indica que el algoritmo de la columna obtuvo mejores resultados que el de la fila con una diferencia estadísticamente significativa y el símbolo " − " significa que no hay diferencias estadísticamente significativas entre ellos.

De los resultados del test de Wilcoxon se puede deducir que, sobre todo para el indicador Épsilon, existen diferencias estadísticamente significativas en el comportamiento de los algoritmos analizados. Se evidencia también que existen diferencias estadísticamente significativas entre los algoritmos NSGA-II y mNSGA-II para instancias con pocas aristas; a medida que aumenta el número de aristas el comportamiento del algoritmo mNSGA-II tiende a igualarse al del NSGA-II.

De manera general, el algoritmo mNSGA-II obtuvo mejores resultados de acuerdo a los indicadores analizados; teniendo en el peor de los casos un comportamiento similar al NSGA-II. Teniendo en cuenta el análisis de tendencia central realizado a los valores de los indicadores y los resultados del test de Wilcoxon, se selecciona el algoritmo mNSGA-II para la identificación de las soluciones no dominadas del problema OPE.

3.3. Análisis de los resultados de la validación del algoritmo seleccionado

Con cada ejecución del algoritmo mNSGA-II se obtienen un conjunto de soluciones no dominadas o frente de Pareto. Sin embargo, según el esquema de funcionamiento de la herramienta donde se desea utilizar el algoritmo, se propone una única WSAN optimizada. Teniendo en cuenta que en este caso todos los criterios de optimización tienen la misma importancia, las soluciones que integran el frente de Pareto identificado en cada ejecución no pueden ser comparadas. Para este experimento se propone la selección de una solución aleatoria dentro de los frentes de Pareto identificados en cada una de las ejecuciones del algoritmo.

La Tabla 6 muestra la media obtenida para cada uno de los indicadores de los criterios de optimización y el comportamiento de las restricciones en los resultados del experimento. Nótese como para toda la muestra el algoritmo es capaz de encontrar soluciones válidas, es decir, soluciones en las que están presentes los vértices que representan a los dispositivos terminales y al coordinador y existe al menos un camino entre cada par de vértices.

Tabla 6. Media obtenida para cada indicador y comportamiento de las restricciones.

	WSAN1	**WSAN2**	**WSAN3**	**WSAN4**
Número de vértices	12.35	28.00	55.00	56.60
Número de caminos de vértices disjuntos	1.07	1.00	1.00	1.00
Grado máximo	2.78	3.00	3.00	3.33
Al menos un camino entre cada par de vértices	se cumple en todos los casos	se cumple en todos los casos	se cumple en todos los casos	se cumple en todos los casos
Presencia de los vértices que representan a los dispositivos terminales y el coordinador	se cumple en todos los casos	se cumple en todos los casos	se cumple en todos los casos	se cumple en todos los casos

Para comparar el comportamiento de los indicadores antes y después de la aplicación del algoritmo, la Tabla 7 muestra los porcientos que representan las medias de las soluciones de los valores iniciales de los indicadores para cada sujeto de la muestra.

Tabla 7. Porcientos que representan las medias obtenidas de los valores iniciales de los indicadores.

	WSAN1	**WSAN2**	**WSAN3**	**WSAN4**
Número de vértices	65%	44.44%	62.5%	60.21%
Grado máximo	43.33%	42.85%	50%	55.5%
Número de caminos de vértices disjuntos	53.5%	100%	50%	50%

La Tabla 7 evidencia una reducción considerable del número de vértices (entre el 35% y el 55%) y del grado máximo (entre el 45% y el 57%), manteniendo al menos el 50% del número de caminos de vértices disjuntos (entre el 50% y el 100%).

3.4. Conclusiones del capítulo

A partir del experimento diseñado para la selección del algoritmo multiobjetivo, se realizó un análisis de la tendencia central, utilizando la mediana y la media de los resultados obtenidos para los indicadores Épsilon y Radio de error. Para ambos indicadores, el algoritmo mNSGA-II se comportó mejor o igual al NSGA-II en la mayoría de los casos. Para comprobar si las diferencias obtenidas fueron estadísticamente significativas se aplicó el test de Wilcoxon; corroborando que el algoritmo mNSGA-II se comporta, en el peor de los casos, igual al NSGA-II.

Para validar la efectividad del algoritmo mNSGA-II se aplicó otro experimento que compara el comportamiento de los indicadores de los criterios de optimización y las restricciones antes y después de la aplicación del algoritmo. Este experimento mostró que el algoritmo aplicado fue capaz de encontrar soluciones válidas de acuerdo a las restricciones, disminuyendo considerablemente el número de vértices y el grado máximo de los sujetos de la muestra y manteniendo en el peor de los casos la mitad de los caminos de vértices disjuntos.

Conclusiones generales

Durante el desarrollo de esta investigación se arribaron a las siguientes conclusiones, que evidencian el cumplimiento de los objetivos propuestos:

- El diseño de WSANs presenta varios puntos en común con el de las WSNs. Sus principales particularidades están en la naturaleza de los dispositivos y en las conexiones necesarias para la coordinación.
- El problema planteado en esta investigación se clasifica, de acuerdo a su complejidad computacional, como NP-Hard.
- A partir de las soluciones similares analizadas y a la clase a la que pertenece el problema de acuerdo a su complejidad computacional, se considera que los algoritmos evolutivos y meméticos pueden encontrar soluciones no dominadas cercanas al frente de Pareto real en este caso.
- De los cinco algoritmos de optimización multiobjetivo propuestos, el algoritmo mNSGA-II obtuvo mejores resultados de acuerdo a los indicadores Épsilon y Radio de error; teniendo un desempeño, en el peor de los casos, similar al NSGA-II.
- Los resultados del test de Wilcoxon mostraron que solo existen diferencias significativas entre el NSGA-II y el mNSGA-II para WSANs pequeñas, en estos casos el mNSGA-II obtiene mejores resultados.
- Se validó la efectividad del algoritmo mNSGA-II comparando el comportamiento de los indicadores para los criterios de optimización y las restricciones antes y después de aplicar el algoritmo. Demostrando que el algoritmo propuesto es capaz de encontrar soluciones válidas de acuerdo a las restricciones, disminuyendo considerablemente el número de enrutadores y el consumo de energía, manteniendo al menos el 50% de la tolerancia a fallos.

Recomendaciones

- Utilizar otra de las estrategias identificadas en la resolución de problemas similares para diseñar el algoritmo de optimización multiobjetivo y comparar con los resultados obtenidos por el mNSGA-II.
- Evaluar el comportamiento del algoritmo mNSGA-II frente a otros problemas de optimización multiobjetivo.
- Analizar los resultados de los algoritmos propuestos para la identificación del frente de Pareto utilizando otras codificaciones para la variable de decisión y otros operadores evolutivos para el cruzamiento y la mutación.
- Estudiar el comportamiento de otros indicadores de calidad para la selección del algoritmo de optimización multiobjetivo.

Referencias bibliográficas

[1] I. A. Ismail, I. F. Akyildiz, y I. H. Kasimoglu, *Wireless Sensor and Actor Networks: Research Challenges*. 2004.

[2] «Wireless Sensor and Actor Networks (WSANs)». [En línea]. Disponible en: http://www.ece.gatech.edu/research/labs/bwn/actors/. [Accedido: 14-abr-2013].

[3] M. Younis y K. Akkaya, «Strategies and techniques for node placement in wireless sensor networks: A survey», *Ad Hoc Netw.*, vol. 6, n.º 4, pp. 621-655, jun. 2008.

[4] I. A. Ismail, I. F. Akyildiz, y I. H. Kasimoglu, *Wireless Sensor and Actor Networks: Research Challenges*. 2004.

[5] W. Guo y M. Zhou, «An emerging technology for improved building automation control», en *IEEE International Conference on Systems, Man and Cybernetics, 2009. SMC 2009*, 2009, pp. 337-342.

[6] M. Gauger, D. Minder, P. J. Marrón, A. Wacker, y A. Lachenmann, «Prototyping sensor-actuator networks for home automation», en *Proceedings of the workshop on Real-world wireless sensor networks*, New York, NY, USA, 2008, pp. 56–60.

[7] M. I. Akbas, M. R. Brust, y D. Turgut, «Local positioning for environmental monitoring in wireless sensor and actor networks», en *2010 IEEE 35th Conference on Local Computer Networks (LCN)*, 2010, pp. 806-813.

[8] Michał Marks, «A Survey of Multi-Objective Deployment in Wireless Sensor Networks», *J. Telecomunications Inf. Technol.*, 2010.

[9] «ZigBee Alliance». [En línea]. Disponible en: http://www.zigbee.org/. [Accedido: 27-mar-2013].

[10] I. A. Nodarse y M. Díaz, «Utilización de WSANs en Sistemas de Control de Edificios», Universidad de Málaga.

[11] A. Pinto, M. D'Angelo, C. Fischione, E. Scholte, y A. Sangiovanni-Vincentelli, «Synthesis of embedded networks for building automation and control», en *American Control Conference, 2008*, 2008, pp. 920-925.

[12] A. Guinard, A. McGibney, y D. Pesch, «A Wireless Sensor Network Design Tool to Support Building Energy Management», en *Proceedings of the First ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Buildings*, New York, NY, USA, 2009, pp. 25–30.

[13] G. Wölfle, R. Wahl, P. Wertz, P. Wildbolz, y F. Landstorfer, «Dominant path prediction model for indoor scenarios», en *German Microwave Conference (GeMIC)*, 2005.

[14] Z. Cai, X. Ren, G. Hao, B. Chen, y Z. Xue, «Survey on wireless sensor and actor network», en *2011 9th World Congress on Intelligent Control and Automation (WCICA)*, 2011, pp. 788-793.

[15] H. Liu, P.-J. Wan, y X. Jia, «Fault-tolerant relay node placement in wireless sensor networks», en *Computing and Combinatorics*, Springer, 2005, pp. 230–239.

[16] W. Zhang, G. Xue, y S. Misra, «Fault-tolerant relay node placement in wireless sensor networks: Problems and algorithms», en *INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE*, 2007, pp. 1649–1657.

[17] X. Han, X. Cao, E. L. Lloyd, y C.-C. Shen, «Fault-tolerant relay node placement in heterogeneous wireless sensor networks», *Mob. Comput. IEEE Trans. On*, vol. 9, n.º 5, pp. 643–656, 2010.

[18] A. J. Perez, M. A. Labrador, y P. M. Wightman, «A multiobjective approach to the relay placement problem in WSNs», en *2011 IEEE Wireless Communications and Networking Conference (WCNC)*, 2011, pp. 475-480.

[19] C. Xue, Y. Zhu, L. Ni, M. Li, y B. Li, «Optimal relay placement for indoor sensor networks», en *Distributed Computing in Sensor Systems (DCOSS), 2012 IEEE 8th International Conference on*, 2012, pp. 209–215.
[20] D. S. Hochbaum y W. Maass, «Approximation schemes for covering and packing problems in image processing and VLSI», *J. ACM JACM*, vol. 32, n.º 1, pp. 130–136, 1985.
[21] J. L. Bredin, E. D. Demaine, M. Hajiaghayi, y D. Rus, «Deploying Sensor Networks With Guaranteed Fault Tolerance», *IEEEACM Trans. Netw.*, vol. 18, n.º 1, pp. 216-228, feb. 2010.
[22] J. Knowles y D. Corne, «Memetic algorithms for multiobjective optimization: issues, methods and prospects», en *Recent advances in memetic algorithms*, Springer, 2005, pp. 313–352.
[23] K. P. Ferentinos y T. A. Tsiligiridis, «Adaptive design optimization of wireless sensor networks using genetic algorithms», *Comput. Netw.*, vol. 51, n.º 4, pp. 1031-1051, mar. 2007.
[24] A. P. Bhondekar, R. Vig, M. L. Singla, C. Ghanshyam, y P. Kapur, «Genetic algorithm based node placement methodology for wireless sensor networks», en *Proceedings of the international multiconference of engineers and computer scientists*, 2009, vol. 1, pp. 18–20.
[25] G. Molina, E. Alba, y E.-G. Talbi, «Optimal sensor network layout using multi-objective metaheuristics», *J. Univers. Comput. Sci.*, vol. 14, n.º 15, pp. 2549–2565, 2008.
[26] J. Jia, J. Chen, G. Chang, Y. Wen, y J. Song, «Multi-objective optimization for coverage control in wireless sensor network with adjustable sensing radius», *Comput. Math. Appl.*, vol. 57, n.º 11, pp. 1767–1775, 2009.
[27] A. Konstantinidis, K. Yang, Q. Zhang, y D. Zeinalipour-Yazti, «A multi-objective evolutionary algorithm for the deployment and power assignment problem in wireless sensor networks», *Comput. Netw.*, vol. 54, n.º 6, pp. 960–976, 2010.
[28] N. Aitsaadi, N. Achir, K. Boussetta, y G. Pujolle, «Multi-Objective WSN Deployment: Quality of Monitoring, Connectivity and Lifetime», en *Communications (ICC), 2010 IEEE International Conference on*, 2010, pp. 1–6.
[29] K. S. S. Rani y N. Devarajan, «Multiobjective Sensor Node Deployement in Wireless Sensor Networks», *Int. J. Eng. Sci.*, vol. 4, 2012.
[30] D. K. Chaudhary y R. L. Dua, «Application of Multiobjective Particle Swarm Optimization to maximize Coverage and Lifetime of wireless Sensor Network», 2012.
[31] W. Guo, B. Zhang, G. Chen, X. Wang, y N. Xiong, «A PSO-Optimized Minimum Spanning Tree-Based Topology Control Scheme for Wireless Sensor Networks», 2013.
[32] A. Tahiri, E. Egea-López, J. Vales-Alonso, J. García-Haro, y M. Essaaidi, «A NOVEL APPROACH FOR OPTIMAL WIRELESS SENSOR NETWORK DEPLOYMENT».
[33] «Real-Time Maude Case Study: The OGDC Algorithm for Wireless Sensor Networks». [En línea]. Disponible en: http://heim.ifi.uio.no/peterol/RealTimeMaude/OGDC/. [Accedido: 22-may-2013].
[34] M. P. Hansen, «Tabu search for multiobjective optimization: MOTS», en *Proceedings of the 13th international conference on multiple criteria decision making*, 1997, pp. 574–586.
[35] E. Chen y S. H. Son, «A fault tolerant topology control in wireless sensor networks», en *ACS/IEEE International Conference on Computer Systems and Applications*, 2005, pp. 57–69.

[36] G. Kortsarz, R. Krauthgamer, y J. Lee, «Hardness of approximation for vertex-connectivity network design problems», *SIAM J. Comput.*, vol. 33, n.º 3, pp. 704-720, 2004.
[37] C. A. Coello Coello,, G. B. Lamont, y Van Veldhuizen, *Evolutionary Algorithms for Solving Multi-Objective Problems*. .
[38] *Applications of Multi-objective Evolutionary Algorithms*. World Scientific, 2004.
[39] C. A. C. Coello, «A comprehensive survey of evolutionary-based multiobjective optimization techniques», *Knowl. Inf. Syst.*, vol. 1, n.º 3, pp. 269–308, 1999.
[40] P. Moscato y C. Cotta, «A Gentle Introduction to Memetic Algorithms», en *Handbook of Metaheuristics*, F. Glover y G. A. Kochenberger, Eds. Springer US, 2003, pp. 105-144.
[41] J. Brownlee, «Clever Algorithms», *Nat.-Inspired Program. Recipes*, p. 436, 2011.
[42] V. V. Vazirani, *Approximation algorithms*. springer, 2001.
[43] R. Poli, J. Kennedy, y T. Blackwell, «Particle swarm optimization», *Swarm Intell.*, vol. 1, n.º 1, pp. 33–57, 2007.
[44] M. Reyes-Sierra y C. C. Coello, «Multi-objective particle swarm optimizers: A survey of the state-of-the-art», *Int. J. Comput. Intell. Res.*, vol. 2, n.º 3, pp. 287–308, 2006.
[45] G. Toscano-Pulido, C. A. C. Coello, y L. V. Santana-Quintero, «EMOPSO: a multi-objective particle swarm optimizer with emphasis on efficiency», en *Evolutionary Multi-Criterion Optimization*, 2007, pp. 272–285.
[46] D. S. Johnson, C. R. Aragon, L. A. McGeoch, y C. Schevon, «Optimization by Simulated Annealing: An Experimental Evaluation; Part I, Graph Partitioning», *http://dx.doi.org/10.1287/opre.37.6.865*, 01-dic-1989. [En línea]. Disponible en: http://pubsonline.informs.org/doi/citedby/10.1287/opre.37.6.865. [Accedido: 22-may-2014].
[47] J. J. Durillo y A. J. Nebro, «jMetal: A Java framework for multi-objective optimization», *Adv. Eng. Softw.*, vol. 42, n.º 10, pp. 760-771, oct. 2011.
[48] «jMetal». [En línea]. Disponible en: http://jmetal.sourceforge.net/algorithms.html. [Accedido: 11-abr-2013].
[49] J. J. Durillo, A. J. Nebro, y E. Alba, «The jMetal framework for multi-objective optimization: Design and architecture», en *2010 IEEE Congress on Evolutionary Computation (CEC)*, 2010, pp. 1-8.
[50] S. Cahon, N. Melab, y E.-G. Talbi, «ParadisEO: A Framework for the Reusable Design of Parallel and Distributed Metaheuristics», *J. Heuristics*, vol. 10, n.º 3, pp. 357-380, may 2004.
[51] «ParadisEO - ParadisEO-MOEO». [En línea]. Disponible en: http://paradiseo.gforge.inria.fr/index.php?n=Main.MOEO. [Accedido: 27-may-2013].
[52] M. G. Martin Lukasiewycz, «Opt4J: a modular framework for meta-heuristic optimization.», pp. 1723-1730, 2011.
[53] «Lecture 20: Hash tables and amortized analysis». [En línea]. Disponible en: http://www.cs.cornell.edu/courses/cs312/2008sp/lectures/lec20.html. [Accedido: 04-jul-2014].
[54] S. Even y R. E. Tarjan, «Network Flow and Testing Graph Connectivity», *SIAM J. Comput.*, vol. 4, n.º 4, pp. 507-518, dic. 1975.
[55] J. Edmonds y R. M. Karp, «Theoretical Improvements in Algorithmic Efficiency for Network Flow Problems», *J ACM*, vol. 19, n.º 2, pp. 248–264, abr. 1972.
[56] K. Deb, A. Pratap, S. Agarwal, y T. Meyarivan, «A fast and elitist multiobjective genetic algorithm: NSGA-II», *IEEE Trans. Evol. Comput.*, vol. 6, n.º 2, pp. 182-197, 2002.

[57] E. Zitzler, M. Laumanns, y L. Thiele, «SPEA2: Improving the Strength Pareto Evolutionary Algorithm», 2001.
[58] J. Knowles y D. Corne, «The Pareto archived evolution strategy: a new baseline algorithm for Pareto multiobjective optimisation», en *Proceedings of the 1999 Congress on Evolutionary Computation, 1999. CEC 99*, 1999, vol. 1, p. -105 Vol. 1.
[59] D. W. Corne, J. D. Knowles, y M. J. Oates, «The Pareto Envelope-based Selection Algorithm for Multiobjective Optimization», en *Proceedings of the Parallel Problem Solving from Nature VI Conference*, 2000, pp. 839–848.