Reporte técnico 2023-B107
Operación óptima del protocolo OLSR para mejorar la calidad del servicio mediante algoritmos evolutivos multi-objetivo

Alumnos: José Manuel Álvarez Guzmán, Jair Eduardo Contreras Contreras
Directores: M. en C. Israel Buitrón Damaso, Dr. Jesús Guillermo Falcón Cardona

27 de mayo de 2023

Índice

1. Introducción

El constante crecimiento de las ciudades y la necesidad de movilidad de los habitantes ha hecho que los medios de transporte tomen un papel importante en la vida diaria de las personas. La conexión de estos medios ha hecho que las redes ad-hoc móviles (MANETs) obtengan una gran relevancia en el medio de las comunicaciones inalámbricas. [1]. Al tener una gran movilidad en los nodos que conforman a la red las VANETs se presentan como una oportunidad para mejorar la eficiencia y seguridad de las comunicaciones que necesiten servicios adicionales como la comunicación de vehículo a vehículo (V2V)[2] Este tipo de redes nos permite la conexión entre vehículos sin la necesidad de contar con una infraestructura previa, lo que ayuda a tener una red auto-configurable y descentralizada. Sin embargo, presentan un desafío al momento de establecer un encaminamiento que sea confiable y que garantice el funcionamiento óptimo del servicio de comunicación.

Los protocolos de encaminamiento han demostrado ser una buena solución al momento de obtener una conexión entre dispositivos usando el menor numero de recursos disponibles dentro de la red. [3] Particularmente el protocolo de encaminamiento por estado de enlace optimizado (OLSR) se nos presenta como una buena alternativa para el encaminamiento de las redes adhoc. [4]. Sin embargo, la alta movilidad de los nodos y el constante cambio de la topología dentro de la red puede hacer que el rendimiento de este protocolo se vea afectado.

Con el fin de generar una operación óptima de este protocolo y mejorar la calidad de servicio. En este trabajo terminal se estudiara la configuración del protocolo OLSR mostrada en el RFC3626[5], así como el uso y análisis de los algoritmos evolutivos multi-objetivo (AEMOs).

Se describirán los conceptos básicos para el entendimiento de las VANETs, los protocolos de encaminamiento utilizados en este tipo de redes, y mas a detalle el protocolo OLSR, así como el uso de los AEMOs en este protocolo para la mejora de su rendimiento. Además de presentar los resultados del estudio de estas redes en distintos escenarios y demostrar su eficiencia. Además de su posible mejora con las herramientas mencionadas.

2. Objetivos

2.1. Objetivo general

Optimizar la operación del protocolo de encaminamiento OLSR para VANET para obtener una mejor calidad de servicio, considerando la optimización simultánea de múltiples funciones objetivo mediante el uso de algoritmos evolutivos multiobjetivo.

2.2. Objetivo específicos

- Analizar el diseño y comportamiento de distintos AEMOs del estado del arte para determinar su viabilidad de utilización en la solución del problema de optimización multi-objetivo (POM) relacionado a la búsqueda de parámetros ótimos de protocolo OLSR.
- Analizar el funcionamiento del simulador NS-2 para familiarizarse con el entorno de trabajo, para que en su uso aplicado a la evaluación de las comunicaciones VANET se obtenga una operación óptima por parte de este.
- Modelar el POM adecuado considerando las áreas que no fueron cubiertas por la propuesta de Toutouth y Alba [4], con el fin de que cada solución tentativa pueda obtener resultados más precisos.
- Implementar la decisión seleccionada en el contexto real utilizando un simulador NS-2, considerando los escenarios y la generación de datos que ofrece dicho simulador para estudiar el rendimiento obtenido tras aplicar el AEMO seleccionado.

3. Conceptos básicos

3.1. Optimización multi-objetivo

De acuerdo con Coello Coello *et al.* [6], un POM con restricciones (asumiendo, sin pérdida de generalidad, la minimización[1] de todos los objetivos) se define de la siguiente forma:

$$\min_{\vec{x}\in\Omega} f(\vec{x}) := [f_1(\vec{x}), f_2(\vec{x}), \ldots, f_m(\vec{x})]^T, \tag{1}$$

sujeto a p restricciones de desigualdad

$$g_i(\vec{x}) \leq 0 \qquad i = 1, 2, \ldots, p, \tag{2}$$

y q restricciones de igualdad

$$h_j(\vec{x}) = 0 \qquad j = 1, 2, \ldots, q, \tag{3}$$

donde $\vec{x} = (x_1, x_2, \ldots, x_n)^T$ es un vector de decisión, $\Omega \subseteq \mathbb{R}^n$ es el espacio de decisión, $g_i : \Omega \to \mathbb{R}, i = 1, 2, \ldots, p$ y $h_j : \Omega \to \mathbb{R}, j = 1, 2, \ldots, p$ representan las restricciones de desigualdad e igualdad que definen el espacio objetivo factible Λ. Por último, $f : \Omega \to \Lambda$ es el vector objetivo donde $f_k : \Omega \to \mathbb{R}, k = 1, 2, \ldots, m$.

Para la correcta definición de un POM es necesario que los objetivos estén en conflicto mutuamente. Es decir, la mejora de un objetivo implica el deterioro de, al menos, otro objetivo. Debido al conflicto entre los objetivos, la solución a un POM es un conjunto de soluciones óptimas que representan los mejores compromisos entre los objetivos. El conjunto de soluciones óptimas se denomina *conjunto de Pareto* y su imagen es denominado el *frente de Pareto*. Para distinguir cuáles vectores de decisión representan una solución óptima a un POM es necesario establecer una relación binaria de orden que involucre el espacio de decisión y el espacio objetivo. En optimización multi-objetivo, la relación binaria de orden usualmente empleada es la *dominancia de Pareto*. que se define a continuación.

Definición 3.1 (Dominancia de Pareto)**.** Dados cualesquiera dos vectores de decisión $\vec{x}$, $\vec{y} \in \Omega$, se dice que $\vec{x}$ domina a $\vec{y}$ (denotado como $\vec{x} \prec \vec{y}$) si y solo si $f_i(\vec{x}) \leq f_i(\vec{y})$ para todo $i = 1, 2, \ldots, m$ y existe al menos un índice $j \in \{1, 2, \ldots, m\}$ tal que $f_j(\vec{x}) < f_j(\vec{x})$.

Existen otras variantes de la dominancia de Pareto que son regularmente usadas en el contexto de optimización multi-objetivo. Estas son las dominancia débil y fuerte de Pareto que son definidas a continuación.

Definición 3.2 (Dominancia de Pareto débil)**.** Dados cualesquiera dos vectores de decisión $\vec{x}$, $\vec{y} \in \Omega$, se dice que $\vec{x}$ domina débilmente a $\vec{y}$ (denotado como $\vec{x} \preceq \vec{y}$) si y solo si $f_i(\vec{x}) \leq f_i(\vec{y})$ para todo $i = 1, 2, \ldots, m$.

[1] Un problema de minimización puede ser convertido a uno de maximización a través de la siguiente identidad: $\min f = -\max -f$.

Definición 3.3 (Dominancia de Pareto estricta)**.** Dados cualesquiera dos vectores de decisión $\vec{x}$, $\vec{y} \in \Omega$, se dice que $\vec{x}$ domina fuertemente a $\vec{y}$ (denotado como $\vec{x} \prec\prec \vec{y}$) si y solo si $f_i(\vec{x}) < f_i(\vec{y})$ para todo $i = 1, 2, \ldots, m$.

Con base en la dominancia de Pareto es posible caracterizar si un vector de decisión representa una solución óptima para el POM. Este tipo de soluciones son denominadas Pareto-óptimas.

Definición 3.4 (Optimalidad de Pareto)**.** Un vector de decisión $\vec{x}^* \in \Omega$ es Pareto-óptima si no existe algún $\vec{x} \in \Omega$ tal que $\vec{x} \prec \vec{x}^*$.

Debido al conflicto mutuo entre los objetivos de un POM, la solución a este es representada por un conjunto de soluciones Pareto-óptimas, denominado conjunto de Pareto. La imagen en el espacio objetivo del conjunto de Pareto es el denominado frente de Pareto.

Definición 3.5 (Conjunto de Pareto)**.** El conjunto de Pareto P^* está definido como:

$$P^* = \{\vec{x}^* \in \Omega \,|\vec{x}^* \text{ es Pareto-óptima}\} \tag{4}$$

Definición 3.6 (Frente de Pareto)**.** El frente de Pareto PF^* está definido como:

$$PF^* = \{f(\vec{x}^*) \in \Lambda | \vec{x}^* \in P^*\} \tag{5}$$

El frente de Pareto suele estar acotado por dos puntos especiales: el punto ideal ($\vec{z}^*$) y el punto de nadir ($\vec{z}^{nad}$). El punto ideal está compuesto por lo valores óptimos de cada función objetivo, considerándolas de forma independiente. Por otra parte, el punto de nadir está compuesto por lo peores valores de cada objetivo en el frente de Pareto. Estos puntos suelen ser empleados como vectores de referencia.

Definición 3.7 (Punto Ideal ($\vec{z}^*$))**.** Las m componentes del vector ideal se definen de la siguiente forma:

$$z_i^* = \min_{\vec{x} \in \Omega} f_i(\vec{x}), i = 1, 2, \ldots, m. \tag{6}$$

Es decir, $\vec{z}^*$ contiene los valores objetivo óptimos para cada f_i, optimizándolas de forma independiente. En consecuencia, $\vec{z} \notin \Lambda$.

Definición 3.8 (Punto de nadir ($\vec{z}^{nad}$))**.** Las m componentes del vector de nadir se definen de la siguiente manera:

$$z_i^{nad} = \max_{\vec{x} \in P^*} f_i(\vec{x}), i = 1, 2, \ldots, m. \tag{7}$$

Es decir, $\vec{z}^{nad}$ contiene los máximos valores objetivo por cada f_i dentro del frente de Pareto. Dependiendo de la geometría de Λ, $\vec{z}^{nad}$ puede ser factible o infactible.

En ocasiones, PF^* contiene una infinidad de vectores objetivo por lo que es impráctica su representación a través de una computadora. Es por ello que algunos algoritmos de optimización multi-objetivo, e.g., como los AEMOs, necesitan mantener una representación finita de P^* y, por consiguiente, de PF^*. A esta representación finita se le denominada conjunto de aproximación.

Definición 3.9 (Conjunto de aproximación)**.** Sea $A = \{\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_N\}$, $\vec{x}_i \in \Omega, i = 1, 2, \ldots, N$. A es un conjunto de aproximación donde $\vec{x}_i \nprec \vec{x}_j$ y $\vec{x}_j \nprec \vec{x}_i$ con $i \neq j$. Es decir, A contiene únicamente vectores de decisión mutuamente no dominados. La imagen de A, denotada como $f(A)$, representa una aproximación finita al frente de Pareto.

3.2. Algoritmos evolutivos multi-objetivo

Los AEMOs son técnicas de optimización estocástica, basadas en poblaciones y libres del uso de la derivada que se han utilizado en una gran variedad de aplicaciones que involucran resolver un POM [6]. Además, los AEMOs se basan en los principios de la selección natural y la transmisión de información entre individuos a través de mecanismos genéticos. En otra palabras, un AEMO se basa en los principios del neodarwinismo. La idea subyacente de un AEMO es evolucionar los individuos de la población de tal forma que después de un número establecido de generaciones, se construye un conjunto de aproximación (ver Definición 3.9). Este conjunto de aproximación representa la salida de un AEMO.

La gran mayoría de los AEMOs son técnicas con articulación de preferencias *a posteriori* [6]. Esto quiere decir que un AEMO busca un conjunto de aproximación que discretice lo mejor posible todo el frente de Pareto sin necesidad de que el tomador de decisiones establezca sus preferencias de antemano. En consecuencia, de acuerdo con Zitzler *et al.* [7], un AEMO debe generar un conjunto de aproximación que cumpla idealmente con las siguientes características:

- **Convergencia:** las soluciones generadas deben estar tan próximas como sea posible al frente de Pareto.
- **Cobertura:** las soluciones generadas deben cubrir la mayor parte del frente de Pareto.
- **Distribución:** las soluciones generadas deben estar uniformemente distribuidas a lo largo del frente de Pareto.

Dado que estas características son ideales, no siempre es posible cumplir con ellas de forma simultánea. Además, puesto que los AEMOs son metaheurísticas, el teorema *No-Free Lunch* [8] indica que no existe una única metaheurística que pueda tener buen desempeño en todo tipo de problemas. En consecuencia, esto ha llevado a la comunidad especializada a generar una gran cantidad de AEMOs con características de diseño diferentes [9, 10, 11, 12, 13, 14].

Independientemente de los diferentes AEMOs que existen actualmente, estos son definidos como algoritmos iterativos que procesan una población $P_t = \{\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_\mu\}$ (donde t es el numero de iteración) a través de los siguientes operadores:

- **Selección** (π): este operador selecciona de P_t un número de soluciones que conformarán el multi-conjunto de padres. Estas soluciones padre serán empleadas para generar nuevas soluciones mediante el operador de variación.
- **Variación** (v): la variación genera λ nuevas soluciones a partir del multi-conjunto $\pi(P_t)$ de padres. En la variación pueden entrar en juego operadores de cruza y mutación. El objetivo principal de los operadores de variación es explorar y explotar el espacio de decisión.
- **Selección ambiental** (σ): determina las soluciones que conformarán la siguiente generación a partir de la población actual P_t y la población de soluciones hijas $v[\pi(P_t)])$. Existen dos variantes de la selección ambiental: (1) $\sigma_{\mu,\lambda}$ y (2) $\sigma_{\mu+\lambda}$. En $\sigma_{\mu,\lambda}$ (también denominada como selección por reemplazo generacional), las mejores μ soluciones hijas de las λ existentes, reemplazan directamente a los μ padres. Por otra parte, en $\sigma_{\mu+\lambda}$ (también denominada como selección extintiva), los mejores μ individuos de la unión de P_t con $v[\pi(P_t)])$ son seleccionados para sobrevivir y dar paso a la población P_{t+1}.

En consecuencia, un AEMO puede ser descrito por la regla iterativa $P_{t+1} = \sigma_{\mu,\lambda}\{P_t, v[\pi(P_t)]\}$ o $P_{t+1} = \sigma_{\mu+\lambda}\{P_t \cup v[\pi(P_t)]\}$, en función del uso de una selección ambiental $\sigma_{\mu,\lambda}$ o $\sigma_{\mu+\lambda}$, respectivamente. Es necesario mencionar que algunos AEMOs pueden hacer uso de una población secundaria conocida como archivo $\mathcal{A}$. La función de un archivo es preservar las soluciones no dominadas que han sido encontradas a lo largo del proceso evolutivo. El archivo puede ser acotado o no acotado. Tomando lo anterior en cuenta, el Algoritmo 1 describe la estructura general de un AEMO. En la línea 1, se inicializa la población P_0 y de ser necesario el archivo $\mathcal{A}$. El ciclo principal se muestra en las líneas 4 a 10. En la línea 5, se selecciona el multi-conjunto de padres a través del operador de selección π. Luego, se genera el conjunto de soluciones hijas (O) a través del operador v. En caso de que se utilice un archivo, este se actualiza en la línea 7. La selección ambiental se lleva a cabo en la línea 8 a través de selección por reemplazo generacional o por selección extintiva para así formar la siguiente población P_{t+1}. Este ciclo continúa hasta que se cumpla el criterio de paro. El AEMO retorna a P_t o a $\mathcal{A}$ como conjunto de aproximación.

Algorithm 1 Estructura general de un AEMO

Input: Tamaño de población μ; Número λ de soluciones hijas
Output: Conjunto de aproximación

1: Generar la población inicial P_0 de tamaño μ
2: Inicializar $\mathcal{A}$ con las soluciones no dominadas de P_0
3: $t \leftarrow 0$
4: **while** el criterio de parada no se cumpla **do**
5: $M \leftarrow$ Seleccionar multi-conjunto de padres a través de $\pi(P_t)$ o $\pi(\mathcal{A})$
6: $O \leftarrow$ Generar un conjunto de λ soluciones hijas a través de $v[M]$
7: Actualizar $\mathcal{A}$ usando O
8: $P_{t+1} \leftarrow$ Seleccionar las soluciones sobrevivientes a través de $\sigma_{\mu,\lambda}\{P_t, O\}$ o $\sigma_{\mu+\lambda}\{P_t \cup O\}$
9: $t \leftarrow t + 1$
10: **end while**
11: **return** P_t o A

3.3. Redes ad-hoc

Una red ad-hoc (Figura 1) es una red de dispositivos independientes que pueden establecer enlaces para comunicarse entre sí sin necesidad de un punto de acceso centralizado. En otras palabras, estos dispositivos pueden generar una red sin infraestructura [1]. Dentro de esta red, los dispositivos pueden cooperar y unirse o abandonar la red en cualquier momento sin afectar el funcionamiento general de esta [15]. Los dispositivos en una red ad-hoc están equipados con tecnologías inalámbricas, como Wi-Fi o Bluetooth, que les permiten conectarse y comunicarse directamente entre sí. Las redes ad-hoc pueden ser formadas por distintos tipos de dispositivos. Por ejemplo, computadoras portátiles, teléfonos inteligentes, tabletas, sensores y dispositivos asociados al Internet de las cosas, entre otros. Las redes ad-hoc son especialmente útiles en situaciones en las que la infraestructura de red no está disponible, como en áreas remotas o en caso de desastres naturales [16]. Algunas de las principales características de las redes ad-hoc son:

- No hay necesidad de una infraestructura de red pre-existente debido a que los nodos pueden auto-configurarse.
- Los nodos se conectan y desconectan de forma dinámica, haciendo que la red formada sea temporal.
- Los nodos en una red ad-hoc deben tener la capacidad de encaminar los paquetes de datos a través de la red de manera eficiente. Debido a que no siempre se puede contar con una verificación de entrega de los datos, las conexiones deben ser lo fiables posible.
- Las redes ad-hoc pueden ser utilizadas en situaciones en las que no se dispone de una red pre-existente, como en áreas remotas o en situacio-

nes de emergencia. Esto gracias a la capacidad de los nodos de formar una red en cualquier momento dado.

- Las redes ad-hoc son útiles para dispositivos móviles como teléfonos inteligentes, tabletas y computadoras portátiles, ya que pueden conectarse y desconectarse fácilmente de la red.
- Las redes ad-hoc pueden ser utilizadas para la comunicación directa entre dispositivos, lo que puede ser útil en aplicaciones como la transmisión de datos entre vehículos o la comunicación de emergencia en situaciones de desastre natural.

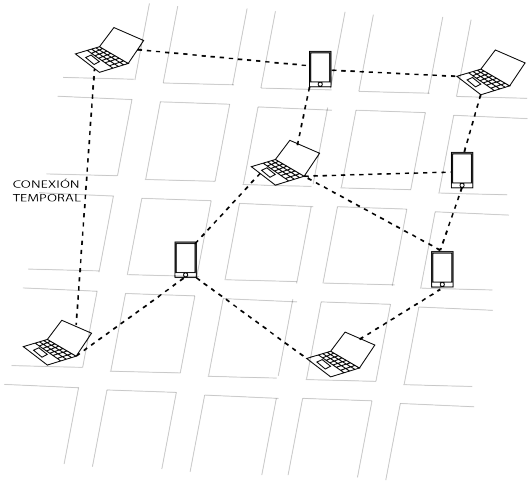

Figura 1: Ejemplificación de una red ad-hoc a partir de dispositivos heterogéneos.

De acuerdo con García et al. [17], las redes ad-hoc pueden clasificarse en dos tipos: redes ad-hoc estáticas y redes ad-hoc móviles. En una red ad-hoc estática, los dispositivos permanecen en una ubicación fija y la topología de la red es más estable (aunque puede variar debido a la unión o retirada de algunos dispositivos). Por otra parte, en una red ad-hoc movil (MANET), los dispositivos se mueven libremente. En consecuencia, la topología de la red puede cambiar constantemente. En la siguiente sección se profundizará en la descripción de MANETs debido a su interés particular en este trabajo.

3.4. Redes vehiculares ad-hoc

De manera general, una MANET es una red donde sus nodos pueden moverse libremente debido a la no existencia de infrestructura previa [3]. Por otra parte, una VANET es una forma especializada de una MANET que se forma por

nodos estáticos y dinámicos [3]. Los nodos estáticos son elementos fijos distribuidos a lo largo del camino o carretera en donde opera la red, llamados unidades de carretera (RSU, por sus siglas en inglés). La función de las RSUs es enviar, recibir y retransmitir paquetes en una red de comunicacion [18]. Además, los RSUs ofrecen acceso a Internet. Por otra parte, los nodos móviles son vehículos equipados con un dispositivo llamado unidad a bordo (OBU, por sus siglas en inglés). Cada OBU permite la conexión de un vehículo con los demás vehículos y con las RSUs. Al igual que las RSUs, los nodos móviles tienen la capacidad de enviar, recibir y retransmitir mensajes entre ellos. Las VANETs se basan en tecnologías inalámbricas de corto alcance. Un ejemplo importante es el estándar IEEE 802.11p [19], también conocido como Wi-Fi de vehículo a vehículo o V2V y el sistema de posicionamiento global (GPS, por sus siglas en inglés). Estas tecnologías permiten la comunicación entre vehículos y la transmisión de información a otros vehículos y a la infraestructura de la carretera [20]. Los vehículos equipados con dispositivos de comunicación inalámbrica pueden formar una red ad-hoc (como se puede observar en la Figura 2) de forma autónoma y establecer conexiones directas entre ellos (par-a-par). Esto es posible incluso si no hay una infraestructura de red fija disponible. En general, las VANETs son empleadas para mejorar la seguridad vial y proporcionar servicios de información y entretenimiento a los conductores y pasajeros. Además, también pueden ser utilizadas para mejorar la eficiencia del tráfico y reducir el consumo de combustible [21].

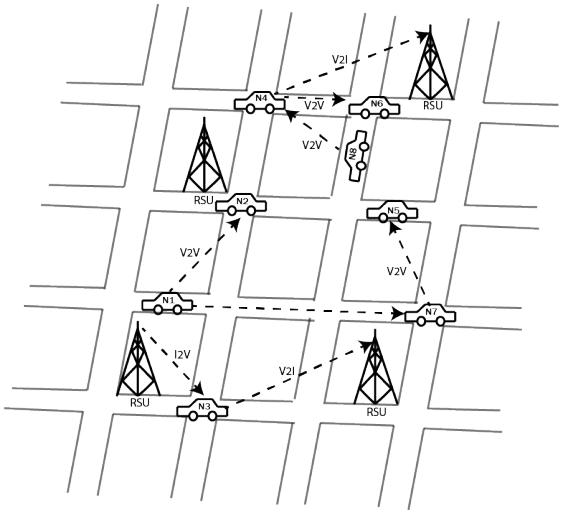

Figura 2: Ejemplificación de la arquitectura de comunicación en una VANET.

Existen tres modos de comunicación en una VANET los cuales son: (1) la comunicación de vehículo a vehículo (V2V), (2) la comunicación de vehículo a infraestructura (V2I), y (3) la comunicación de infraestructura a vehículo (I2V). El tipo de comunicación V2V permite a los vehículos intercambiar diferente tipo de información. Esta información puede variar desde la ubicación y la velocidad del vehículo hasta advertir de la presencia de obstáculos en la carretera. Con base en esta información, la comunicación V2V se usa para mejorar la seguridad en la carretera y proporcionar información en tiempo real sobre el tráfico y las condiciones del camino [3]. Por otra parte, los vehículos también pueden utilizar la comunicación V2V para proporcionar servicios de entretenimiento a los pasajeros y para mantenerse conectados a Internet [22, 23].

La comunicación I2V permite a las infraestructuras como semáforos y señales de tráfico intercambiar información con los vehículos [3]. La información enviada por la infraestructura puede incluir alertas de seguridad, como la presencia de un obstáculo en la carretera, así como información sobre el estado del tráfico y los tiempos de espera en los semáforos. La información enviada por los vehículos puede incluir la ubicación y la velocidad, así como las condiciones del tráfico. La comunicación I2V se utiliza para intercambiar para mejorar la eficiencia de del tráfico y garantizar la seguridad en las carreteras. [24, 25, 26].

La comunicación V2I permite a los vehículos intercambiar información con la infraestructura de la carretera, por ejemplo, con los sistemas de peaje y los centros de gestión del tráfico [3]. La información enviada por los vehículos puede incluir la ubicación y la velocidad, así como las condiciones del tráfico. La comunicación V2I se utiliza para mejorar la eficiencia del tráfico y proporcionar servicios de peaje [27].

Existen otras configuraciones en la comunicación de las redes vehiculares ad-hoc (VANETS) como: la comunicación vehículo a peatón (V2P), la comunicación directa en el vehículo (DIV), la comunicación vehículo a red eléctrica (VIG) y la comunicación vehículo a hogar (V2H). Sin embargo, las configuraciones antes descritas son muy poco recurridas debido a que aun no existe una red donde estas configuraciones sean totalmente aplicables [28].

Las VANETs se enfrentan a diversos desafíos, como la alta velocidad de los vehículos, la falta de una infraestructura de red fija, la gestión de la seguridad y la privacidad de los datos transmitidos, así como la necesidad de contar con un protocolo de encaminamiento eficiente y escalable, entre otros [29]. Estos desafíos requieren soluciones innovadoras y estrategias de optimización para garantizar un funcionamiento adecuado de las VANET y aprovechar al máximo su potencial en términos de comunicación vehicular y aplicaciones relacionadas. La investigación y el desarrollo en este campo continúan avanzando, y las VANETs tienen el potencial de mejorar significativamente la seguridad vial y la eficiencia del tráfico en el futuro [30].

3.5. Protocolos de encaminamiento para VANETs

Los protocolos de encaminamiento permiten establecer la conectividad en la capa de red, seleccionando siempre el mejor camino posible para la conexión de extremo a extremo [3]. El propósito principal de estos protocolos es minimizar el tiempo de comunicación, minimizando la cantidad de recursos utilizados de la red. En cuanto a las VANETs se refiere, el creciente uso y la gran utilidad de estas ha generado un interés particular en el estudio de sus protocolos de encaminamiento. Estos protocolos ayudan a las VANETs en la getión de red y el tráfico, la movilidad, la calidad del servicio y el intercambio r'apido de información [16]. Desafortunadamente, las VANETs no pueden hacer uso de los procolos de encaminamiento estándar como el IEEE 802.11 [19] debido al comportamiento dinámico de la red. Es por esto que para las VANETs existen tipos especializados de protocolos de encaminamiento [31]. La Figura 3 muestra una clasificación de los protocolos de encaminamiento en cinco categorías [3, 31]: (1) encaminamiento basado en posición, (2) encaminamiento basado en topología, (3) encaminamiento de geo-difusión, (4) encaminamiento de difusión, y (5) encaminamiento basado en clúster. En las siguientes secciones se describirá a mayor detalle cada una de estas categorías. Además, se describe a detalle el protocolo de encaminamiento OLSR como un ejemplo significativo de estos y sobre el cual trata este trabajo.

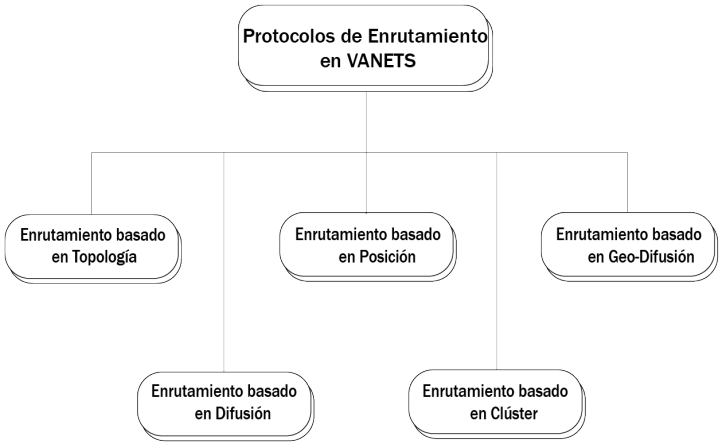

Figura 3: Clasificación de los protocolos de encaminamiento para VANETs.

3.5.1. Protocolo de encaminamiento basado en posición

En el protocolo de encaminamiento basado en posición se basa en el uso de coordenadas geográficas. Cada vez que el nodo origen trata de comunicarse con el nodo de destino, se emplean sus coordenadas geográficas y su dirección de red. La Figura 4 ejemplifica este tipo de encaminamiento. Por otra parte, este tipo de protocolo de encaminamiento no requiere de un establecimiento de la ruta. Cada que un nodo esté listo para enviar un paquete de información, se deben obtener la coordenadas geográficas de la ubicación del destino. Todas las actividades se realizan mediante el seguimiento de la ubicación y algún tipo de estrategia de reenvió de paquetes que se debe implementar en el nodo que envía la información. Alguna de las ventajas de usar el encaminamiento basado en posición son [3]:

- El descubrimiento y el seguimiento de las rutas de los nodos no son necesarios.
- Escalabilidad de la red.
- La gran movilidad de los nodos no afecta al funcionamiento de la red.

La desventaja de este tipo de protocolos es la dependencia a los servicios de ubucación como lo puede ser el GPS. Algunos ejemplos de este tipo de encaminamiento son: Encaminamiento Codicioso sin Estado de Perímetro (GPSR, por sus siglas en inglés) y Algoritmo de Efecto de Encaminamiento a Distancia para Movilidad (DREAM, por sus siglas en inglés) [3].

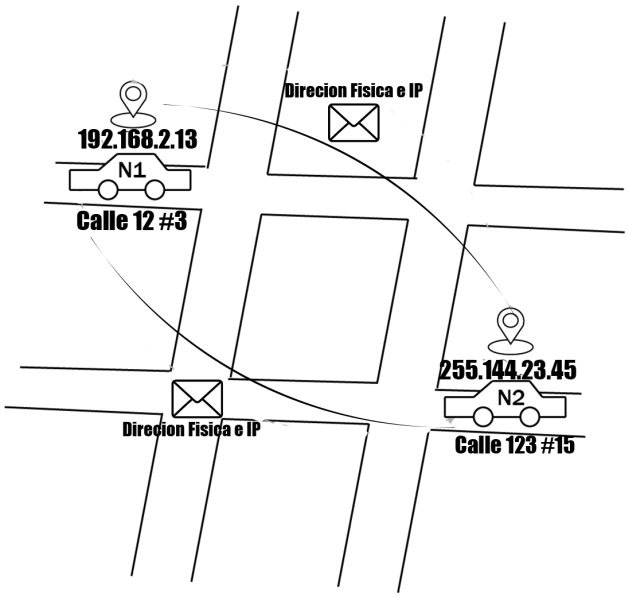

Figura 4: Ejemplificación del encaminamiento basado en posición donde se intercambia dirección IP y dirección fisica.

3.5.2. Protocolo de encaminamiento basado en topología

Este tipo de encaminamiento utiliza la información de los enlaces de la red para así enviar un paquete desde el nodo fuente al nodo destino. La información de los enlaces se explota a partir de las tablas de encaminamiento. Estas tablas almacenan los datos de los enlaces entre los nodos. La Figura 5 ejemplifica la idea básica de un protocolo de encaminamiento basado en topología. Con esta información se toman las decisiones de encaminamiento para el envío de la información entre un nodo origen y un nodo destino [16]. Debido a la alta movilidad de los nodos que provoca cambios dinámicos de la topología con el ingreso y egreso de nodos [16], es necesario calcular constantemente la topología de la red. Es decir, las tablas de encaminamiento son constantemente actualizadas. Esto hace que un protocolo de encaminamiento basado en topología sea más lento con respecto a los otros protocolos de encaminamiento para VANETs.

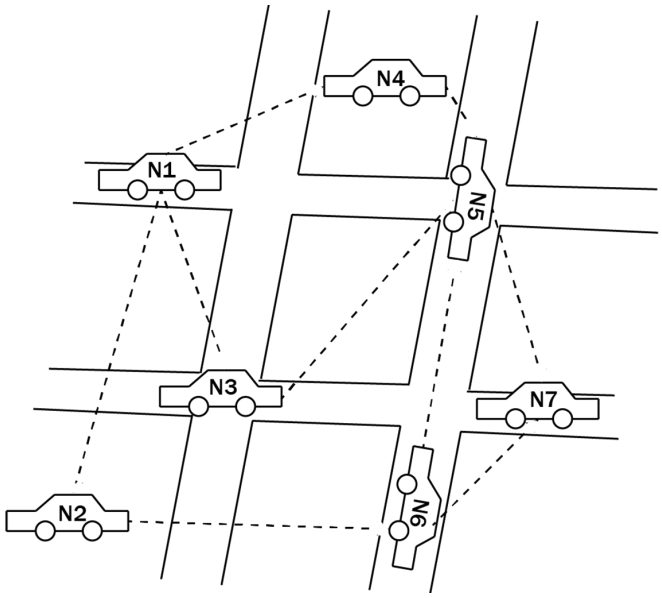

Figura 5: Ejemplificación del encaminamiento basado en topología.

Los protocolos en encaminamiento basados en topología se clasifican que dos tipos[3]: (1) proactivos y (2) reactivos. Estas categorías son descritas a continuación:

Proactivos En un protocolo de encaminamiento proactivo, la elección de la ruta depende de los algoritmos utilizados. Este tipo de protocolo mantiene actualizadas constantemente las tablas de encaminamiento y busca rutas óptimas de manera anticipada. Utiliza protocolos de encaminamiento vector-distancia estándar. Algunos ejemplos de este tipo de encaminamiento son: Vector de Distancia Secuenciado por Destino (DSDV, por sus siglas en inglés), Encaminamiento Optimizado Basado en Estado de Enlace (OLSR, Optimized Link State Routing por sus siglas en inglés) y Difusión de Topología Basada en Reenvío de Ruta en Sentido Inverso (TBRPF, por sus siglas en inglés).

Reactivos En un protocolo de encaminamiento reactivo, la ruta de encaminamiento se determina en función de la necesidad y se mantienen solo las rutas que se están utilizando en ese momento. Esto significa que no se mantienen actualizadas constantemente las tablas de encaminamiento, sino que se establecen dinámicamente cuando se necesita transmitir datos a un destino específico. De esta manera, se tiene un conjunto de rutas accesibles que están en uso en todo momento, lo que ayuda a reducir la sobrecarga en la red.

Algunos ejemplos de este tipo de encaminamiento reactivo son los protocolos Encaminamiento Dinámico de Fuente (DSR, Dynamic Source Routing por sus

siglas en inglés), Vector de Distancia Bajo Demanda para Redes Ad Hoc (AODV, por sus siglas en inglés), Algoritmo de Encaminamiento Temporalmente Ordenado (TORA, por sus siglas en inglés). Estos protocolos son ampliamente utilizados en redes ad hoc y ofrecen ventajas en términos de eficiencia y adaptabilidad a los cambios en la topología de la red.

3.5.3. Protocolo de encaminamiento basado en difusión

En este tipo de encaminamiento, el paquete es enviado por toda la red a todos los nodos disponibles como se muestra en la Figura 6. Este tipo de encaminamiento es particularmente útil cuando el nodo destino está fuera del alcance del nodo origen. No obstante, la difusión podría causar mayor uso de los recursos de la red si se emplea constantemente. Por lo regular, en las VANETs se utiliza un protocolo de encaminamiento de difusión para intercambiar información entre vehículos en situaciones de emergencia donde es crucial entregar mensajes de manera rápida y eficiente. En consecuencia, un nicho de aplicación importante de este tipo de protocolos de encaminamiento es en las aplicaciones de seguridad. Esto porque permiten la entrega oportuna de información en momentos críticos, como accidentes o eventos inesperados. Sin embargo cuentan con desventajas como la existencia de nodos que no puedan ser detectados directamente por los demás nodos de la red, o la alta probabilidad de que los mensajes colisionen entre sí. Algunos ejemplos de este tipo de protocolos de encaminamiento son: Encaminamiento Basado en Vector de Distancia CAST (DV-CAST), Algoritmo de Agrupamiento Descentralizado (DECA, por sus siglas en inglés) y Encaminamiento Basado en Agrupamientos Consciente de la Energía (POCA,g por sus siglas en inglés).

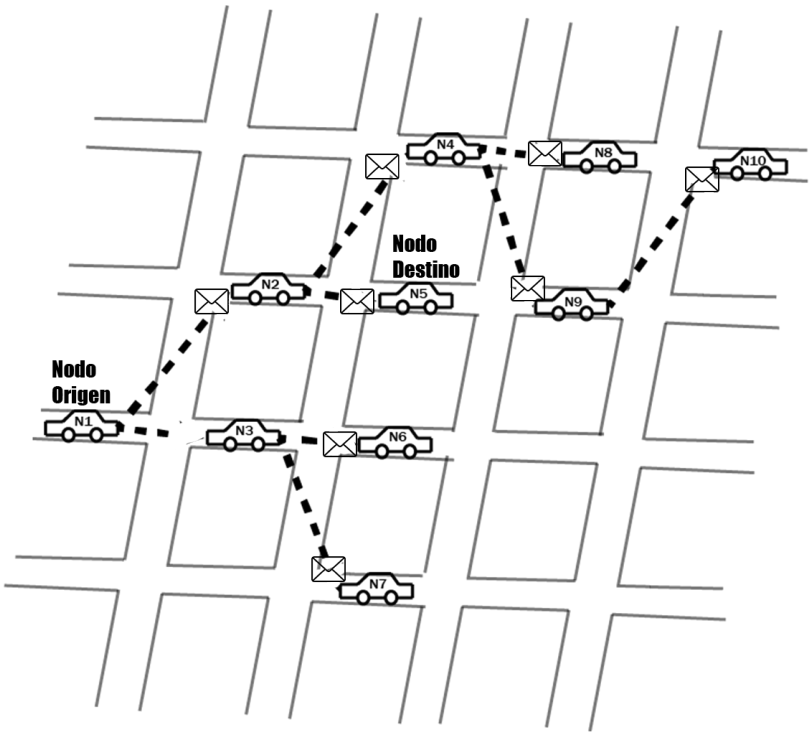

Figura 6: Encaminamiento basado en difusión donde el nodo origen(N_1) reenvía el paquete por toda la red hasta que encuentra al nodo destino(N_5).

3.5.4. Protocolo de encaminamiento basado en geo-difusión

Su objetivo principal es la comunicación de vehículos dentro de una región (Figura 7)prescrita en un momento determinado, conocida como zona de relevancia (ZORt). Cuando el nodo de destino pertenece a otras ZOR, se tiene que contar con una zona de reenvío (ZOF), de esta manera, el vehículo que entra en la ZOF debe reenviar el paquete a las otras ZOR. Sin embargo este tipo de protocolos presentan una desventajas en términos de latencia a la hora de realizar los envíos, así como los problemas de contención y colisión. Por otro lado, una de las ventajas es la reducción en la sobrecarga de la red. Algunos ejemplos de este tipo de encaminamiento son: Coordenadas Virtuales Geográficas Implícitas (IVG, por sus siglas en inglés) y CASTOR Geográfico Direccional (DG-CASTOR).

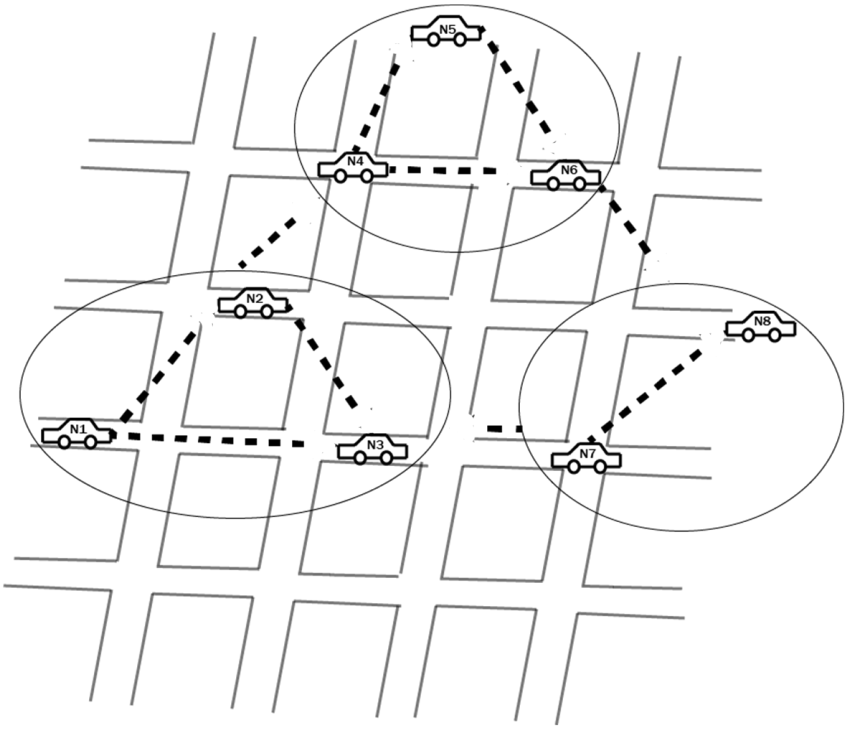

Figura 7: Encaminamiento basado en Geo-Difusión.

3.5.5. Protocolo de encaminamiento basado en cluster

Si varios vehículos con velocidad, dirección u otras características en común forman una red, se necesita un líder para gestionar la comunicación entre los nodos y proporcionar escalabilidad. Si el paquete debe enviarse dentro del mismo grupo, se utilizará el camino directo. Algunos ejemplos de este tipo de encaminamiento son: Red de Enrutamiento Oportunista Basada en Contención Centrada en la Información (COIN, por sus siglas en inglés) y Enrutamiento Basado en Contención para Lora (LORA_CBF, por sus siglas en inglés).

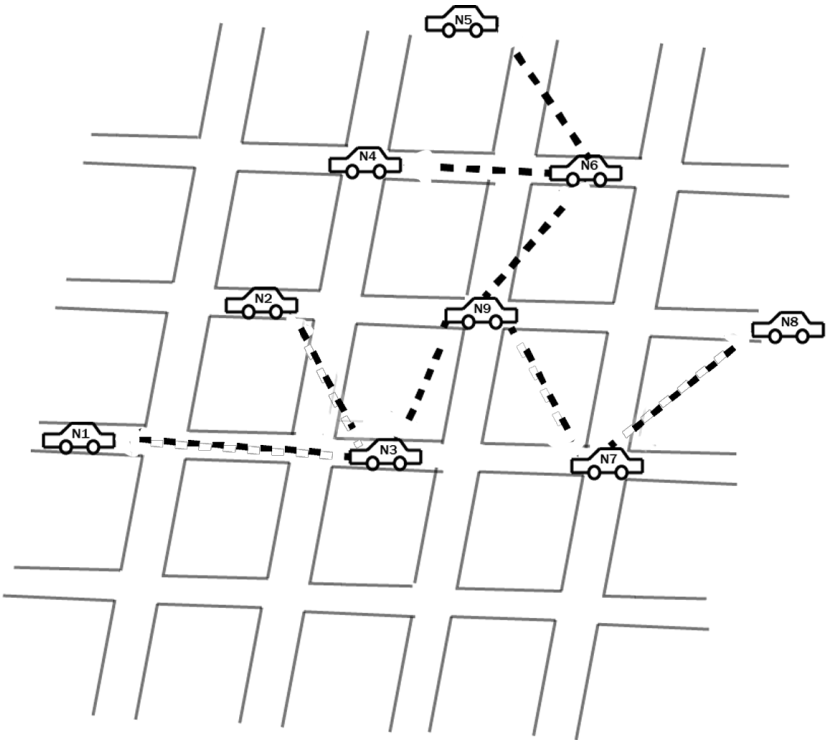

Figura 8: Encaminamiento basado en cluster.

3.5.6. Comparación de los distintos tipos de protocolos de encaminamiento.

Para una análisis mas concreto de las investigación que se está realizando, en este apartado se hace la comparación de los cinco tipos de protocolos de encaminamiento [31]. Para ello, se presenta la Tabla 1 en cual abordaremos las diferencias entre los tipos de protocolos de encaminamiento. La tabla destaca aspectos clave de los protocolos, como el método de reenvío utilizado, la estrategia de recuperación en caso de pérdida de paquetes, el entorno en el que se despliegan, la consideración de flujos de tráfico realistas, los requisitos de mapas digitales y equipos virtuales, y ejemplos representativos de cada protocolo.

Característica	Proactivo	Reactivo	Posición	Clúster	Difusión	Geo-Difusión
Método de reenvío	Inalámbrico Multi-Salto	Inalámbrico Multi-Salto	Metodo Heurístico	Inalámbrico Multi-Salto	Inalámbrico Multi-Salto	Inalámbrico Multi-Salto
Estrategia de recuperación	Reenvío Multisalto	Transporte y Reenvío	Transporte y Reenvío	Transporte y Reenvío	Transporte y Reenvío	Inundación
Ambiente	Urbano	Urbano	Urbano	Urbano	Carretera	Carretera
Flujo de tráfico realista	Sí	Sí	Sí	No	No	Sí
Requerimiento de mapa digital	No	No	No	Sí	No	No
Requisito de equipos virtuales	No	No	No	Sí	No	No
Ejemplos	DSDV OLSR TBRPF	DSR AODV TORA	GPSR DREAM	COIN LORA _CBF	DV-CAST DECA POCA	IVG DG-CASTOR

Cuadro 1: Comparación de los protocolos de encaminamiento usados en VANETs.

3.6. OLSR

OLSR es un protocolo de encaminamiento basado en topología del tipo proactivo [5]. OLSR representa una mejora al protocolo de enlace clásico para redes móviles ad hoc (IEEE 802.11) usado en la mayoría de las MANETs[19]. De acuerdo con el RFC 3626 [5], el diseño del protocolo OLSR se basa principalmente en: (1) estado de los enlaces, (2) la topología de la red, (3) la retransmisión multi-punto, (4) proactividad y reactividad con las que los nodos intercambiar mensajes, y (5) la difusión *multicast* y *unicast*. El diseño del protocolo OLSR permite su funcionamiento de manera distribuida OLSR. Adicionalmente, no requiere de un sistema centralizado de mensajes de control porque cada nodo envía mensajes de control periódicamente. Así, el protocolo puede soportar una perdida razonable de algunos mensajes de control que se producen frecuentemente por razones como colisiones u otros problemas de transmisión [32].

La entrega secuenciada de mensajes tampoco es requerida. Cada mensaje de control incluye un número de secuencia que se incrementa para cada mensaje, lo que permite al receptor del mensaje identificar de manera sencilla la información más reciente. Esto es posible incluso si los mensajes se han reordenado durante la transmisión. No es es necesario modificar el formato de los paquetes IP, lo que significa que cualquier pila IP existente puede ser utilizada sin modificaciones. El protocolo se centra exclusivamente en la gestión de las tablas de encaminamiento sin interferir en otros aspectos de las comunicaciones IP.

OLSR minimiza la sobrecarga de tráfico de control al usar los nodos relés multi-punto(MPR por su siglas en inglés). Estos son nodos de retransmisión multipunto que son responsables de reenviar a los demas nodos de su área de coberutra el tráfico de control[4]. Esto reduce significativamente el número de retransmisiones requeridas para enviar un mensaje a todos los nodos en la red [5]. Los dos procesos principales llevados a cabo en el protocolo OLSR son el descubrimiento de vecindario y la difusión de topología. En estos procesos se llevan a cabo el envío de tres tipos de mensajes [4]:

- **HELLO**: intercambio entre nodos vecinos para la detección de enlaces, detección del vecindario y selección de MPR.
- **TC**: generado por MPR para indicar que otros nodos lo han seleccionado como MPR.
- **MID**: se envían para informar sobre las interfaces de red utilizadas para participar en la red.

Los procesos del protocolo OLSR están regulados por un conjunto de parámetros predefinidos en el OLSR RFC 3626, que se muestran en la tabla 2 [5]:

Parámetro	Valor Estándar	Rango
HELLO_INTERVAL	2.0 [s]	$[2{,}0, 15{,}0]$
MID_INTERVAL	2.0 [s]	$[5{,}0, 15{,}0]$
TC_INTERVAL	5.0 [s]	$[4{,}0, 35{,}0]$
WILLINGNESS	3	$[0, 7]$
NEIGHB_HOLD_TIME	3 $\times$ HELLO_INTERVAL	$[5{,}5, 45{,}0]$
MID_HOLD_TIME	3 $\times$ TC_MID	$[10{,}5, 90{,}0]$
TOP_HOLD_TIME	3 $\times$ TC_INTERVAL	$[10{,}5, 90{,}0]$
DUP_HOLD_TIME	30.0 [s]	$[10{,}5, 90{,}0]$

Cuadro 2: Parámetros predefinidos para OLSR según el RFC 3626.

3.7. Calidad de servicio del protocolo OLSR en VANETs

Uno de los factores que puede afectar el rendimiento del protocolo OLSR es la elección de los ajustes para los parámetros.Por ejemplo la configuración del "$HELLO\_INTERVAL$" el cual se encarga de la detección de cambios en la topología. Por lo tanto el optimizar estas configuraciones es de suma importancia para mejorar la calidad de servicio (QoS) [4]. Debido a que a configuración estándar de OLSR ofrece una QoS moderada cuando se utiliza en VANETs [33].

La eficiencia de OLSR en términos de consumo de recursos y rendimiento ha sido ampliamente estudiada y documentada. Investigaciones como la realizada por Chen et al. [34] demostraron que OLSR tiene un bajo consumo de ancho de banda y energía en comparación con otros protocolos de encaminamiento utilizados en VANETs. Esta eficiencia en el consumo de recursos permite un mejor uso de la capacidad de la red y contribuye a una mayor calidad del servicio.

Otro factor para enfatizar la importancia de optimizar la operación de OLSR, es la capacidad de adaptación a entornos altamente dinámicos es otro factor que contribuye a su calidad de servicio [35].El protocolo OLSR es capaz de manejar eficientemente cambios rápidos en la topología de la red, lo que es crucial en entornos vehiculares donde los nodos pueden moverse rápidamente y las

conexiones pueden ser intermitentes. Esto garantiza una conectividad continua y confiable, lo que a su vez mejora la calidad del servicio ofrecida por OLSR en las VANETs.

4. Modelación del problema

Para mejorar el desempeño de OLSR, se plantea definir un problema que involucre optimizar de forma simultánea tres objetivos: (1) tasa de pérdida de paquetes (PLR, por sus siglas en inglés), (2) retardo de extremo a extremo (E2ED, por sus siglas en inglés), y (3) carga de encaminamiento normalizada (NRL, por sus siglas en inglés). Este problema fue considerado inicialmente en el trabajo de Toutouh y Alba [4].

La optimización automatica de los parámetros del protocolo OLSR puede ser definido en dos fase: la fase de evaluación y la fase de optimización[4].

- Fase de optimización: esta fase hace uso de los AEMOs para encontrar la solución óptima dentro de un espacio de busqueda.
- Fase de evaluación: durante la fase de evaluación se asignara un valor de calidad cuantitativo conocido como "$fitness$" el cual analizaremos para definir el rendimiento del protocolo.

Para poder evaluar y analizar el rendimiento de la configuración del protocolo se usan las siguientes métricas [36] [4]:

- Tasa de pérdida de paquetes (PLR por sus siglas en inglés). La tasa de pérdida de paquetes representa el número de paquetes que no se entregan durante la transmisión. Por lo que buscaremos que el PLR tenga el menor valor posible, ya que esto nos garantiza la entrega exitosa de los paquetes. Podemos definir el PLR de la siguiente manera:

$$PLR = \frac{Nmero\,de\,paquetes\,perdidos\,por\,el\,destino}{Nmero\,de\,paquetes\,enviados\,por\,la\,fuente} = 1 - PDR \quad (8)$$

  Siendo la Tasa de entrega de paquetes (PDR por sus siglas en inglés) los paquetes de datos que son entregados correctamente.

- Retardo de extremo a extremo (E2ED por sus siglas en inglés). Se refiere a la diferencia de tiempo entre la generación de datos por parte de una aplicación y el tiempo en que estos llegan al destino. Esta medida sirve para medir la demora entre la transmisión de los paquetes. Podemos definir el E2ED de la siguiente manera:

$$E2ED = \frac{\sum_{idpaquete=1}^{N}(NbRecibido_{idpaquete}) - NbEnviado_{idpaquete})}{N} \quad (9)$$

Donde N es el numero de paquetes que se enviaron correctamente, $Recibido_{idpaquete}$ es el tiempo que tarda el paquete en ser recibido y $Enviado_{idpaquete}$ es el tiempo que tarda el paquete en ser enviado.

- Carga de encaminamiento normalizada (NRL por sus siglas en inglés). Es la cantidad de paquetes de encaminamiento transmitidos dentro de la red por cada paquete de datos. Podemos definir el NRL de la siguiente manera:

$$NRL = \frac{Nmero\,de\,paquetes\,de\,encaminamiento\,transmitidos}{Nmero\,del\,total\,de\,paquetes\,recibidos} \tag{10}$$

  Un alto valor de NRL indicaría una mejor eficiencia en el protocolo OLSR ya que esto significaría que se estan gestionando de manera correcta un mayor número de paquetes. Por lo tanto nuestro objetivo sera minimizar el complemento de NRL($1 - NRL$)

La propuesta inicial contempla la minimización de cada una de las funciones objetivo y es descrita de la siguiente manera:

$$\text{mín}\, F_1 = PLR \tag{11}$$

$$\text{mín}\, F_2 = 1 - NRL \tag{12}$$

$$\text{mín}\, F_3 = E2ED \tag{13}$$

Una nueva propuesta en el modelado del POM, contempla incluir funciones objetivo enfocadas a evaluar 2 métricas que fueron omitidas en la revisión del estado del arte.

- Sobrecarga de encaminamiento (RO por sus siglas en inglés). Representa el número de bytes de encaminamiento requeridos por los protocolos de encaminamiento para construir y mantener sus rutas. Podemos definir el RO de la siguiente manera:

$$RO = \frac{Nmero\,de\,paquetes\,de\,encaminamiento}{Nmero\,de\,paquetes\,de\,datos} \tag{14}$$

  Cuando el valor de RO es alto, significa que se están generando y transmitiendo muchos mensajes de encaminamiento para mantener la conectividad y la comunicación dentro de la VANET. Esto puede tener varios efectos negativos [37], como: consumo de ancho de banda, retardo en la entrega y consumo de energía.

- Saltos (hops) Es el número de dispositivos por los que tiene que pasar el paquete de información desde el nodo emisor hasta el nodo destino, por lo que nuestro objetivo es minimizar el número de estos dispositivos. Proporciona un indicador para evaluar la eficiencia y latencia de la transmisión de los datos.

Debido a que el núnmero de saltos refleja la longitud de la ruta[37] podemos definir los saltos de la siguiente manera:

$$hops = d(E, D) \tag{15}$$

Donde los saltos(s) son calculados con la distancia(d) entre el nodo emisor(E) y el nodo destino(d).

En OLSR cada nodo crea y mantiene de manera temporal un conjunto de vecinos que se encuentran entre una distancia de 1 salto y dos saltos[37].

Así, la minimización de 2 funciones objetivo más propone la mejora en la eficiencia de entrega de paquetes [37]. La nueva propuesta del modelo del POM relacionados a la búsqueda de parámetros óptimos de OLSR es descrito a continuación:

$$\text{mín } F_1 = PLR \tag{16}$$

$$\text{mín } F_2 = 1 - NRL \tag{17}$$

$$\text{mín } F_3 = E2ED \tag{18}$$

$$\text{mín } F_4 = 1 - RO \tag{19}$$

$$\text{mín } F_5 = hops \tag{20}$$

5. Experimentación y Resultados

La validación de los resultados se centra en la implementación de un marco de trabajo experimental para la optimización multi-objetivo de OLSR mediante el uso del algoritmo NSGA-II que fue propuesto por Toutouh y Alba [4]. El proceso experimental incluye el uso del simulador de redes Network simulator 2 (ns-2) para obtener un análisis de las comunicaciones en escenarios VANET reales obtenidos con el simulador de tráfico vehicular SUMO. La importancia de validar los resultados de este marco de trabajo radica en poder obtener un punto de comparación adecuado para la siguiente fase del trabajo terminal. Además, validando los resultados del proceso experimental inicial sera posible asegurar la validez, confiabilidad y aplicabilidad de los resultados. Habiendo validado la propuesta inicial que motivo la realización de este trabajo terminal, es posible continuar con la siguiente fase de experimentación. Esta siguiente fase contempla aplicar la nueva propuesta del POM para la operación óptima de OLSR y el uso de diferentes AEMOs que no han sido contemplados en el estado del arte.

5.1. Configuración del ambiente computacional

5.1.1. Sistema operativo

El marco de trabajo experimental se estableció en una maquina virtual del sistema operativo Linux, usando la distribución Ubuntu en su versión 22.04 (Jammmy), a la maquina virtual se le asigno una memoria base de 4 Gb, 2 CPUs y un limite de ejecución del 100 %. El uso de la virtualización de un sistema operativo se debe a la incompatibilidad del simulador ns-2 con otras distribuciones y versiones de Linux, siendo la versión antes mencionada la mas estable y con la que se pudieron solucionar todos los problemas de compatibilidad. Una vez que la distribución esta operando de manera normal en la maquina virtual, se requiere de la instalación del paquete build-essentials, el cual contiene las herramientas esenciales para la construcción de la mayoría de los otros paquetes fuente (compilador C / C ++, libc, y make) [38]. Otro requisito que debe cumplir el sistema operativo es contar con los repositorios multiverse y universe que estuvieron disponibles hasta antes de la versión 18.04 de Ubuntu [39].

Es posible que al agregar los repositorios multiverse y universe en el sistema se genere un error GnuPrivacy Guard (GPG), para solucionarlo es necesario agregar una clave de autenticación GPG en Ubuntu, el identificador de la clave de autenticación será proporcionado por el sistema operativo al detectar el error GPG, por lo que solo se deberá solicitar el acceso desde el servidor de claves hkp://keyserver.ubuntu.com:80 [40]. Todos los ajustes en el sistema operativo se requieren para atender los requisitos de ns-2 (5.1.3).

5.1.2. OLSR

Debido a que ns-2 no cuenta de forma nativa con una implementación del protocolo OLSR, es necesario modificar algunos archivos fuente del simulador para poder configurar una versión de este protocolo previo a la instalación de ns-2, la versión de OLSR que se implemento fue la versión 1.0 de UM-OLSR, desarrollada en 2004 por Francisco J. Ros [41]. La versión 1.0 de UM-OLSR incluye dentro del paquete un archivo para la modificación de ficheros .patch que es el que permite modificar todas las entradas correspondientes en los archivos de configuración del simulador ns-2. El manejo de ficheros con el archivo .patch previene errores de escritura que pudieran alterar el funcionamiento correcto del protocolo. El uso de la versión 1.0 se sustenta en algunos problemas presentados con el archivo .patch de la versión 0.0.8, ya que no se completaba de manera satisfactoria la modificación de cada uno de los archivos de ns-2.

5.1.3. Simulador ns-2

Ns es un simulador de eventos discretos orientado a la investigación en redes. Ns permite simular protocolos TCP, de encaminamiento y multi-difusión en redes alámbricas e inalámbricas (locales y por satélite). Este simulador siempre ha contado con importantes contribuciones de otros investigadores, incluido el

código inalámbrico de los proyectos Daedelus de la UCB y Monarch de la CMU y Sun Microsystems [42]. La versión de Ns para realizar las simulaciones contempladas en el marco experimental es la 2.35, esta versión es la mas reciente que es compatible con UM-OLSR 1.0.

El paquete de instalación de ns-2, es ns-allinone-2.35 [43], el instalador contenido en este paquete incluye todas las herramientas que pueden ser utilizadas adicionalmente a ns-2. Este instalador genera una validación y una instalación limpia del simulador y de las herramientas adicionales . Las consideraciones que hay que tener en cuenta al instalar ns-2 en la versión 22.04 de Ubuntu son las siguientes:

Versiones del compilador gcc y g++: El simulador requiere de las versiones 4.8 de gcc y g++, en el directorio ns-allinone-2.35 se listan diferentes directorios en donde es necesario editar algunos archivos para cambiar la versión de los compiladores, las declaraciones en donde se encuentren las declaraciones @CC@ y @CPP@ se sustituyen por: gcc-4.8 y g++-4.8, respectivamente. Las rutas de los archivos que se deben modificar son las siguientes: ns-allinone-2.35/Makefile.in, ns-allinone 2.35/otcl-1.14/Makefile.in, nam-1.15/Makefile.in y xgraph-12.2/Makefile.in. Estos cambios junto a las modificaciones hechas en el sistema operativo evitaran conflictos entre versiones de los compiladores con los que trabaja el simulador.

Manejo de errores: Existe un error en el archivo ls.h [44] ubicado en la ruta: ns-2.35/linkstate/, este error esta presente en todas las versiones de ns-2 y se trata de un error de declaración que se soluciona cambiando la palabra **erase** por la instrucción **this->erase** en la linea 137.

Atender las consideraciones antes mencionadas es muy importante para garantizar el funcionamiento correcto del simulador, una vez realizados los cambios en los ficheros correspondientes y el patch para el protocolo OLSR (5.1.2), la instalación se realiza ejecutando el programa install, este único instalador se encarga de gestionar todos los instaladores adicionales en el paquete ns-allinone-2.35. La última modificación que se debe hacer en el sistema operativo es agregar en el archivo /.bashrc las rutas indicadas por el instalador al completar este proceso, cada ruta agrega la ubicación de cada herramienta para poder usar su respectiva orden, para verificar la correcta instalacion del simulador se escribe la orden ns, al ejecutar la orden la entrada debe ser el simbolo " %".

El simulador puede generar escenarios con protocolos de red como TCP y UDP, comportamientos de fuentes de tráfico como FTP, Telnet, Web, CBR y VBR, mecanismos de gestión de colas de enrutadores como Drop Tail, RED y CBQ, algoritmos de enrutamiento y muchos más [42]. En ns-2, se utiliza C++ para la implementación detallada del protocolo y Otcl para la configuración. Los objetos C++ compilados se ponen a disposición del intérprete Otcl y, de este modo, los objetos C++ ya creados pueden controlarse desde el nivel OTcl.

5.2. Réplica de resultados previos

5.2.1. Escenario urbano y ns-2

La propuesta inicial de Toutouh y Alba [4], propone el uso de una porción del área metropolitana de Málaga (Figura 9) para generar un escenario VANET a partir del flujo de trafico en esta área. El área propuesta consta de un total de 240,000 m^2 en Málaga-Costa del Sol, Málaga, España.

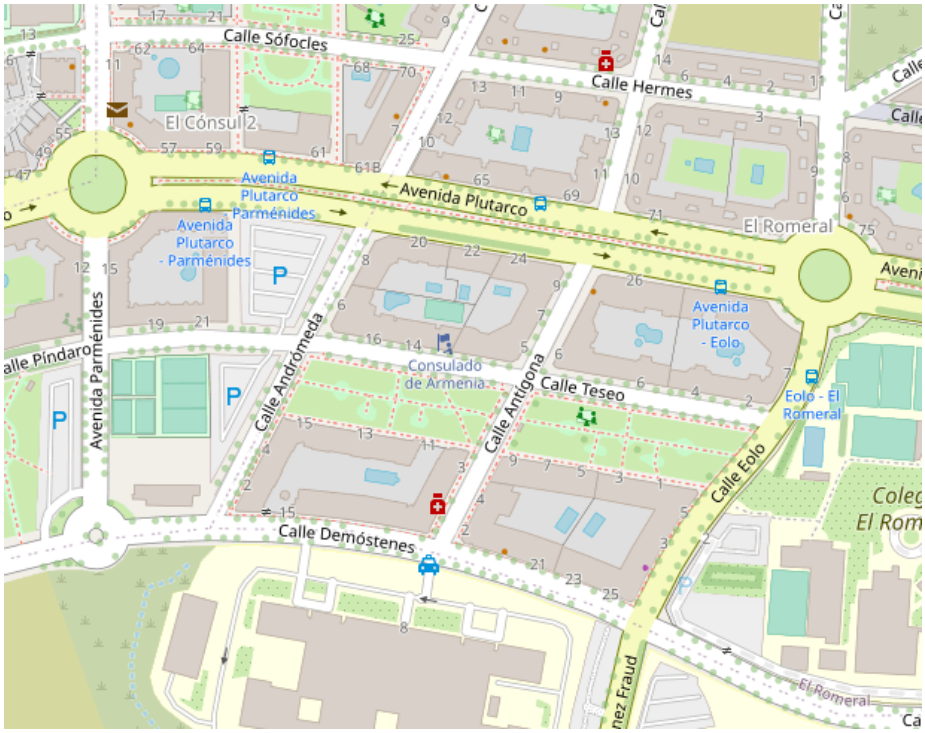

Figura 9: Área urbana de Málaga para la generación del escenario VANET.

Las densidades de tráfico propuestas por Toutouh y Alba [4] contemplan escenarios con 20, 30 y 40 vehículos en el área propuesta. En el presente trabajo se contempla el primer escenario para replicar la infraestructura propuesta y validar el funcionamiento y los resultados del marco de trabajo que se desarrolló (5.2.3) y (5.2.4). Para obtener el entorno vehicular y trasladarlo al simulador ns-2, se hizo uso del simulador de tráfico vehicular SUMO [45], este simulador contiene en su paquete de instalación otras herramientas adicionales para manejar la información en diferentes formatos para diferentes propósitos. Con SUMO se generarán reglas de trafico reales, ya que se toman elementos de la plataforma OpenStreetMap [46] como la ubicación de semáforos, sentidos de flujo vehicular, cruces peatonales, etc. La obtención de todo el escenario junto

con la información y reglas viales, se gestiona usando el programa osmWebWizard.py (Figura 10) contenido el el fichero tools del paquete de SUMO. La información generada por osmWebWizard permite extraer los diferentes elementos para establecer el diseño de la VANET en un script Tcl para simular este entorno en ns-2. El tiempo de simulación propuesto es de 3 minutos, en estos 3 minutos se simulará el movimiento de los 20 vehículos del escenario propuesto en el área geográfica de Málaga.

Figura 10: Generación del escenario VANET con osmWebWizard.

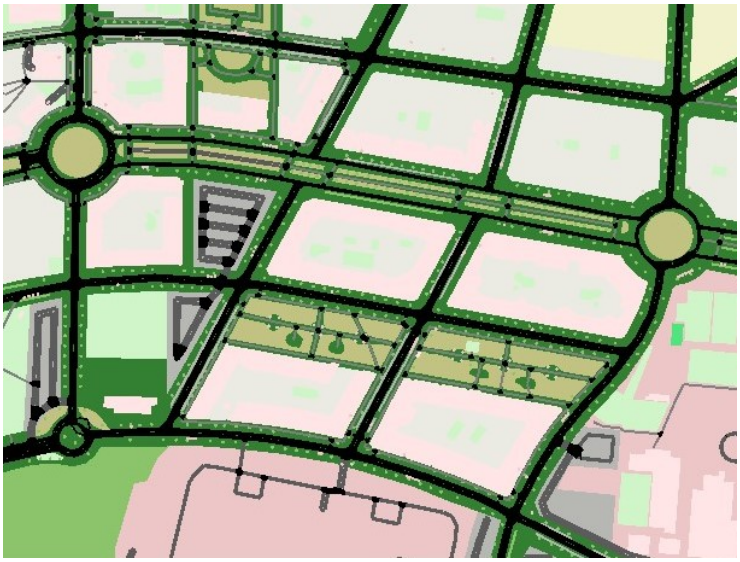

Figura 11: Composición del escenario para simular el trafico vehicular en SUMO.

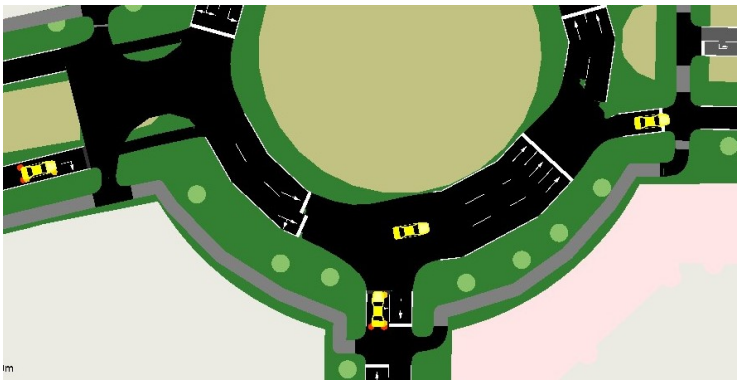

Figura 12: Simulación del comportamiento de vehículos en el escenario urbano.

La información que genera osmWebWizard esta contenida en un archivo con extensión .sumocfg, con esta configuración se puede verificar la simulación del entorno vehicular usando la interfaz gráfica de SUMO. Al cargar este archivo con extensión .sumocfg en el entorno de trabajo de SUMO se podrá observar el escenario urbano extraído de OpenStreetMap y se podrá gestionar una simulación donde se podrá observar el comportamiento del flujo vehicular (Figura 11 y 12).

La configuración se SUMO permite obtener las configuraciones finales para la simulación, la obtención de cada archivo esta descrita por un flujo de procesamiento, donde un archivo generará otro en secuencia con información que otros procesos necesitan para llegar a los archivos finales (Figura 13). Con la herramienta del paquete de herramientas de SUMO llamada polyconvert se extraen archivos con extensión .xml, donde una ultima herramienta llamada traceExporter genera archivos de movilidad Tcl para describir el comportamiento de los vehículos en ns-2. Este ultimo proceso siempre generara 3 archivos, el primero es un archivo que es una plantilla para configurar la red vehicular, posteriormente dos archivos mas, activity.tcl y mobility.tcl.

El primer archivo se usara para describir toda la configuración de la red que se va a simular en ns-2, activity.tcl contiene la declaración de los vehículos con un identificador de SUMO, esta información es un respaldo para recuperar la declaración de objetos que son descritos como vehículos, en el simulador cada vehículo toma el papel de un nodo de red en el escenario VANET. El archivo mobility.tcl va a contener información que describe el movimiento de cada nodo en el escenario urbano, cada nodo tiene 2 porciones de instrucciones:

1. Declaración de un nodo en el espacio de simulación en 2 dimensiones: cada nodo se genera en el espacio de simulación mediante coordenadas en los ejes X, Y y Z.

2. Descripción del movimiento de cada nodo: por cada momento en el tiempo de simulación cada nodo tiene asociado un bloque de movimientos

para los otros nodos que se generan, estos movimientos se describen con coordenadas de origen y destino en el espacio de de 2 dimensiones.

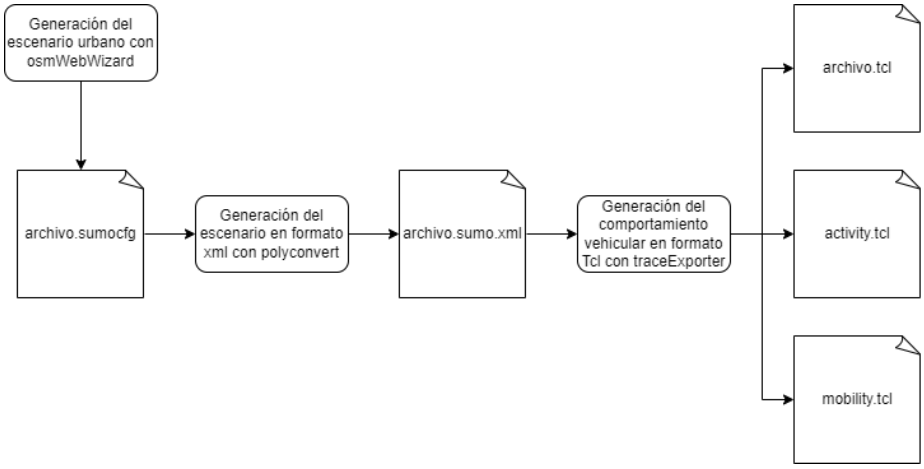

Figura 13: Flujo de procesamiento de los archivos de configuración.

Con la información obtenida, el siguiente paso para concretar la simulación del escenario VANET es terminar de construir el archivo Tcl que describirá el comportamiento de la red vehicular. En la plantilla generada por el ultimo procesos de conversión de archivos se describirán las características de la red, así como la configuración de hardware con la que trabajara cada nodo, la información de los 20 nodos (vehículos) se extrae directamente del archivo de configuración mobility.tcl. El cuadro 4 describe las especificaciones de de la red para el proceso experimental, estas especificaciones fueron propuestas por Toutouh y Alba [4] y el proceso para definir cada una se puede consultar a mayor detalle en [47] y [48].

Parámetro	**Valor/Protocolo**
Modelo de propagación	Nakagami
Frecuencia portadora	5.89 GHz
Capacidad media 1	6 Mbps
Capa PHY/MAC 1	IEEE 802.11p
Protocolo de encaminamiento	OLSR
Capa de transporte	UDP
Tamaño de paquete CBR	512 bytes
Tasa de datos CBR	500 kbps
Tiempo CBR	60 s

Cuadro 3: Especificaciones para la comunicación en la VANET.

Una vez cubiertas las especificaciones se puede concretar una simulación

completa del escenario con el protocolo OLSR implementado y 20 nodos de red con 10 conexiones entre nodos para establecer actividades que involucren CBR. Este proceso entregara un archivo con extensión .tr, este archivo es una traza que describe por columnas los resultados de la simulación. La traza que genera el simulador contiene los siguientes campos:

1. Tipo de evento
2. Tiempo de ejecución
3. Nodo origen
4. Nodo destino
5. Tipo de paquete
6. Tamaño de paquete
7. Banderas de estado
8. ID de Otcl
9. Dirección de origen
10. Dirección de destino
11. Número de secuencia
12. Identificador de paquete

La variación que presenta el archivo de traza cuando OLSR es implementado un identificador de paquete [HELLO 0 0 0], el identificador de paquete será siempre este debido al intercambio entre nodos vecinos. La variación en el archivo de traza propicio el desarrollo de un analizador de trazas descrito en la sección 5.2.3. El analizador de trazas es fundamental para obtener las métricas de QoS involucradas en la modelación del POM.

Para tener una referencia visual del comportamiento del escenario se usa la herramienta Network animator (nam), al finalizar la simulacion, desde el script se puede gestionar la creación de un archivo con extensión .nam, que contiene la información para que el comportamiento sea interpretado de manera visual (Figura 14) y sea posible percibir el movimiento de los nodos tal y como se estableció en SUMO.

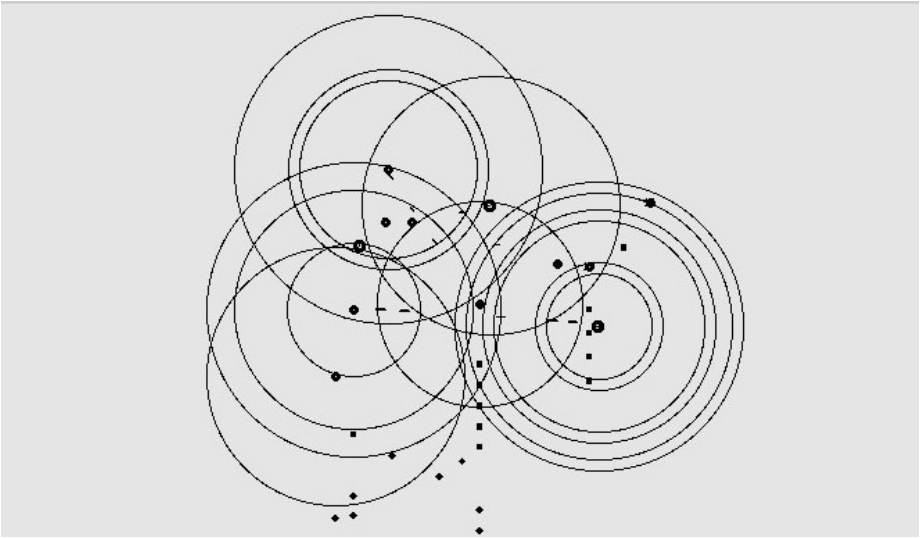

Figura 14: Comportamiento de la VANET en Network animator.

Considerando lo anterior es posible integrar el proceso de simulación al AEMO propuesto para validar los resultados del marco de trabajo.

5.2.2. NSGA-II

El algoritmo NSGA-II es una algoritmo evolutivo de ordenamiento no-dominado el cual se aplica a POMs. En el cual como se muestra en la Figura 15 se construye una población P_0 aleatoria la cual se ordenara a partir de su no-dominación. A cada solución se le asignará un rango de acuerdo a su nivel de no-dominación (donde el rango 1 es el mejor, el rango 2 es el segundo mejor, y así sucesivamente) [**?**].

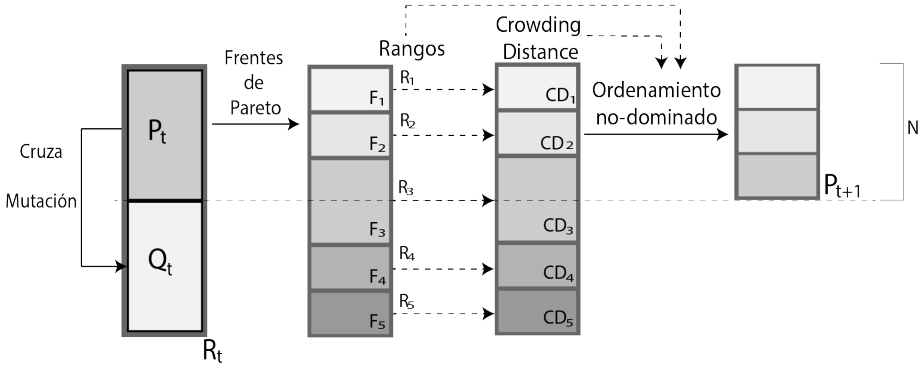

Figura 15: Ciclo principal del algoritmo NSGA-II.

Como se ve en el algoritmo 2, una vez tengamos la población inicial, se aplicaran técnicas de cruza(5.2.2.1) y mutación(5.2.2.2) (lineas 8 - 11) para crear la población Q_0 la cual tendrá un tamaño N (donde N es el tamaño de la población) Combinaremos las poblaciones para crear la población R_0 de tal manera que $R_0 = P_0 \cup Q_0$ y tendrá un tamaño $2N$ (linea 13) Se hará la selección de los mejores individuos de la población R a partir de: obtener sus frentes de Pareto (5.2.2.4), obtener su rango(5.2.2.5) de acuerdo al frente al que pertenece, con la poblacion R y los frentes de Pareto obtendremos su distancia de aglomeración (5.2.2.6), de acuerdo a la distancia de aglomeración y el rango obtendremos el ordenamiento no dominado(5.2.2.7)de la población. Y finalmente obtendremos los mejores individuos para formar la siguiente población P (lineas 13 - 19)

Algorithm 2 Pseudo-codigo del algoritmo NSGA-II

1: $P \leftarrow$ Población Inicial [N] // Población aleatoria de tamaño N
2: $Q \leftarrow \emptyset$ [N] // Población auxiliar de tamaño N
3: $Ng \leftarrow \mathbb{Z}$ //Número de generaciones
4: $Tm \leftarrow R \in [0{,}0, 1{,}0]$ //Taza de mutación
5: $Tc \leftarrow R \in [0{,}0, 1{,}0]$ //Taza de cruza
6: **while** $g < Ng$ **do**
7: **for** $i \leftarrow 1$ **to** (N/2) **do**
8: $padres \leftarrow seleccion(P)$
9: $nuevo\_individuo \leftarrow cruza(Tc, padres)$
10: $nuevo\_individuo \leftarrow mutacion(Tm, padres)$
11: $Q \cdot aadir(nuevo\_individuo)$
12: **end for**
13: $R \leftarrow P \cup Q$
14: $fronts \leftarrow frentes\_pareto(R)$
15: $rango \leftarrow rank(fronts)$
16: $crowding\_distance \leftarrow crowding(R, fronts)$
17: $ordenamiento\_no\_dominado \leftarrow ordenamiento(rango, crowding\_distance, R)$
18: $P \leftarrow seleccion\_mejores\_individuos(ordenamiento\_no\_dominado)$
19: $g \leftarrow g + 1$
20: **end while**

5.2.2.1 Cruza Para llevar a cabo la cruza, se deben elegir dos padres(soluciones) y llevar a cabo una versión modificada del operador de recombinación de N puntos para codificadores de problemas de valores reales. En este operador se define una combinación lineal entre el padre A y el padre B, lo que genera dos nuevos individuos el descendiente A y descendiente B. Como se muestra en la figura 16, para los parámetros del protocolo OLSR (Tabla 2) se define un intercambio de pares de genes los cuales son relacionados de la siguiente manera:

- Intercambio 1: HELLO_INTERVAL(1) - NEIGHB_HOLD_TIM(5)
- Intercambio 1: MID_INTERVAL(2) - MID_HOLD_TIM(6)
- Intercambio 3: TC_INTERVAL(3) - TOP_HOLD_TIM(7)

Y un intercambio de genes idénticos de los siguientes genes:

- Intercambio 4: WILLINGNES(4)
- Intercambio 5: DUP_HOLD_TIM(8)

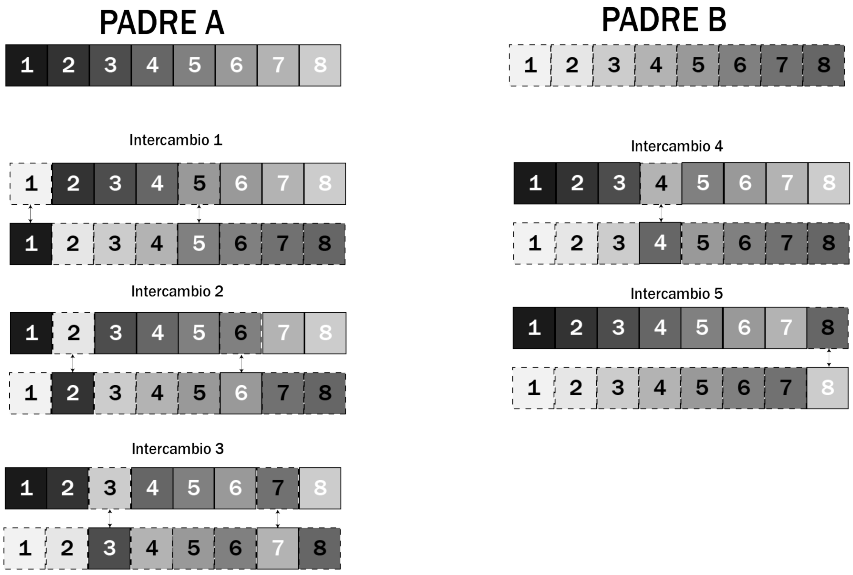

Figura 16: Tipos de intercambio durante la cruza de dos individuos(soluciones).

Cada que se realice una operación de cruza se efectuaran 3 de los intercambios mencionados anteriormente de manera aleatoria.

5.2.2.2 Mutación Con el fin de incluir diversidad en la población, se inluirá el operador de mutación. El cual genera nueva información genética de manera aleatoria, siempre y cuando el invididuo(solución) sea viable, o dicho de otra forma que se encuentre dentro de los parámetros establecidos(Tabla 2).
Por lo que se mutara información siguiendo la ecuación 21[4].

$$x_{p,i}^{(g+1)} = rand(z_{(i,MIN)}, z_{(i,MAX)}) \tag{21}$$

5.2.2.3 Matriz de dominancia La matriz de dominancia es utilizada para determinar la relación de dominancia entre dos individuos(soluciones) en los frentes de Pareto de acuerdo a la Definición 3.3. Es una representacion tabular que compara todas las soluciones en un frentes de Pareto, donde las filas y columnas corresponden a una solución dentro del frente. Dadas dos soluciones f_1 y f_2, podemos seguir el diagrama en la Figura 17para determinar la dominancia de una solución sobre la otra.

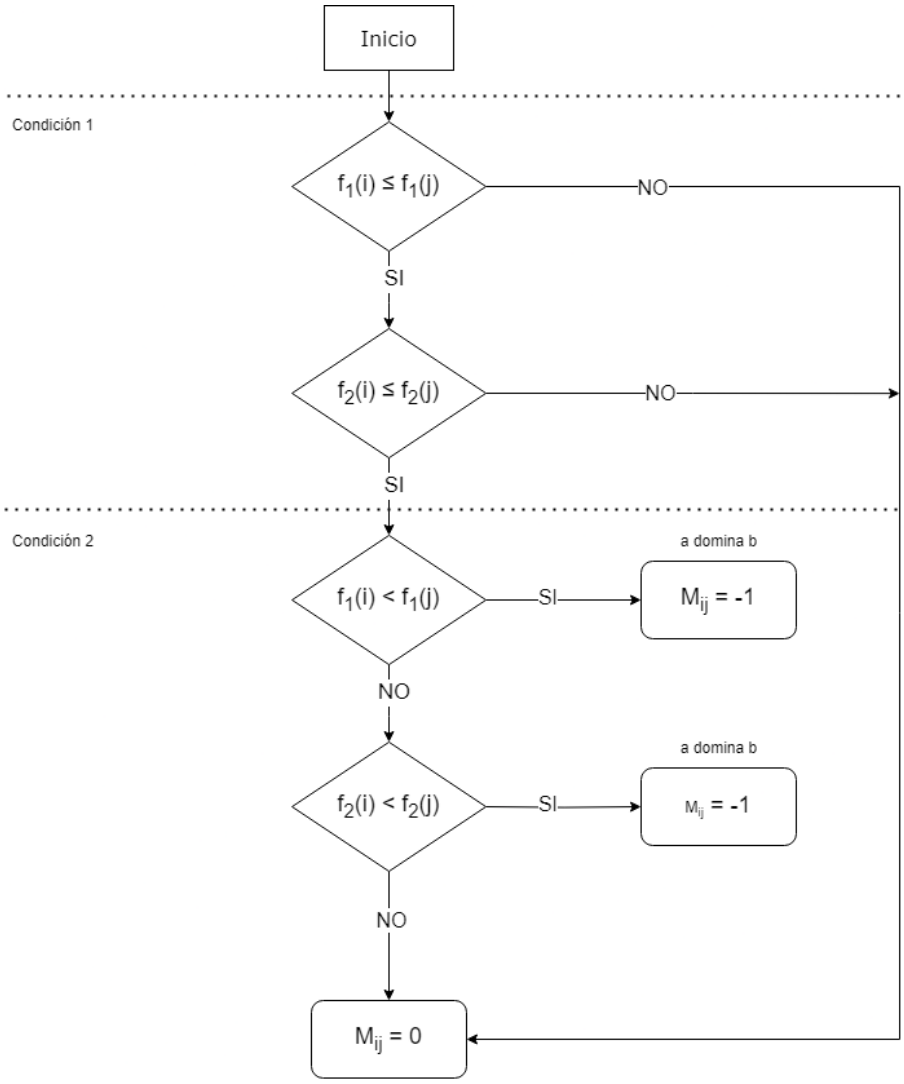

Figura 17: Diagrama para determinar la dominancia dentro de la matriz.

De esta manera la entrada de la matrix tomara el valor de $M_{ij} = 0$ si $i \not\prec j$, o $M_{ij} = -1$ si $i \prec j$

5.2.2.4 Frentes de Pareto Los frentes de pareto son una idea fundamental dentro de los **AEMOs!** (**AEMOs!**). Retomando la Definición 3.6 se puede decir que los frentes de Pareto son un conjunto de soluciones que cumplen con el mismo orden de no dominancia(Figura 18) de acuerdo con la Definición 3.3.

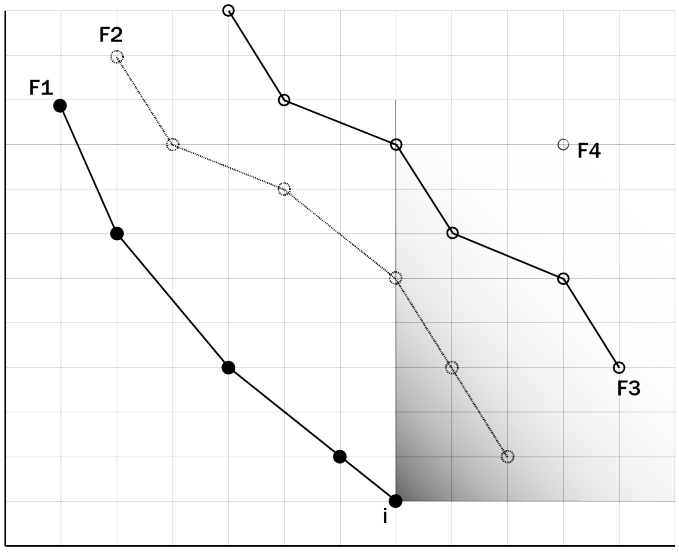

Figura 18: Ejemplo de frentes de Pareto.

Nos permiten explorar las soluciones óptimas dentro de un espacio de búsqueda de un POM. Una vez implementada la matriz de dominancia (5.2.2.3) podemos definir la implementación de los frentes de pareto con el siguiente pseudo-código:

Algorithm 3 GenerarFrentesPareto

```
1:  frentes ← []
2:  dominancia ← MatrizDeDominancia(poblacin)
3:  while Haya soluciones no asignadas en la población do
4:      frente_actual ← []
5:      for i ← 1 to cada solución en la población do
6:          for ci ← 1 to cada solución en la población do
7:              if dominancia[i, j] then
8:                  Agregar la solución a frente_actual
9:              end if
10:         end for
11:     end for
12:     agregar frente_actual a la lista de frentes
13: end while
14: return frentes
```

5.2.2.5 Rango Los rangos dentro de un AEMO nos permiten saber el nivel de no dominancia en el que la solución se encuentra. De acuardo al Algoritmo 4 A cada solución dentro del espacio de búsqueda se le asigna un rango. En este caso conforme a los frentes de Pareto(linea 4). Mientras mas bajo sea el rango, nos indica que tiene una mayor dominancia.

Como se muestra en la Figura 19, según el frente de Pareto en el que la solución se encuentra sera el rango que se le asignara.

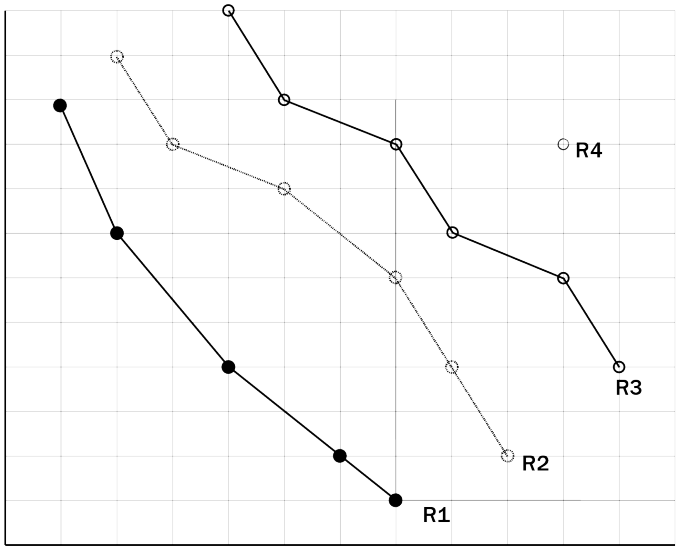

Figura 19: Ejemplo de rangos conforme a los frentes de Pareto(Figura 19).

Algorithm 4 Asignación de rango a las soluciones

1: $rangos \leftarrow []$
2: **for** $i \leftarrow 1$ **to** $Nmero\,de\,frentes$ **do**
3: **for** $solucion\,en\,el\,frente$ **do**
4: $rangos[solucion] = i$
5: **end for**
6: **end for**
7: **return** $rangos$

5.2.2.6 Distancia de aglomeración La distancia de aglomeración sirve como un indicador de la densidad de las soluciones que rodean a una solución especifica en la población. Para calcular la distancia de aglomeriacion haremos uso del perimetro del cuboide(Figura 20) formado por los vecinos más cercanos.

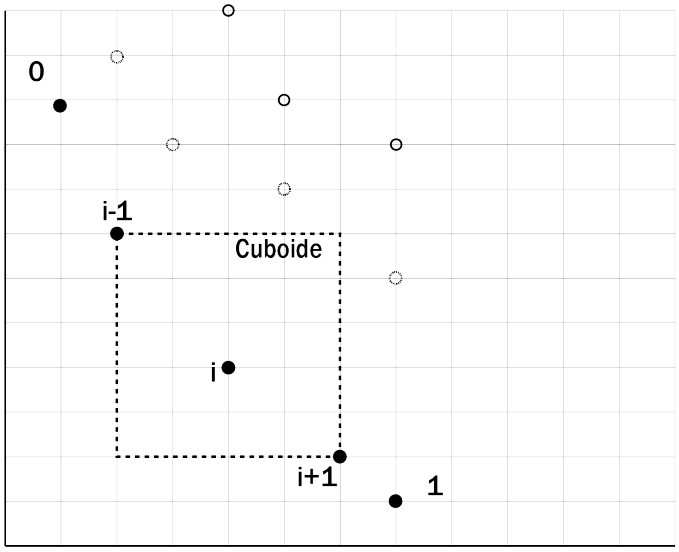

Figura 20: Cálulo de la distancia de aglomeriación.

Una vez tenemos identificado el cuboide de una solución, podemos calcular la distancia de aglomeración de la siguiente manera(Definición 5.1):

Definición 5.1 (Distancia de aglomeración)**.** Para cada F_i(frente de Pareto) de n individuos.
Inicializamos la distancia en o

$$F_i(CD_j) = 0 \tag{22}$$

Si CD es igual a:

$$CD_j = \sum_{m=1}^{M} \frac{f_m(X_{j+1}) - f_m(X_{j-1})}{f_m(X_{max}) - f_m(X_{min})} \tag{23}$$

Siendo $i = 1, 2, ..., k$ el número de frentes de Pareto, $j = 1, 2, ..., n$ el número de individuos y m el número de funciones objetivo.

El Algortimo 5 propone la implemnetación de la Definición 5.1.

Algorithm 5 Cálculo de la distancia de aglomeración)

1: $P \leftarrow$ Población
2: $N \leftarrow$ Tamaño de la población P
3: $M \leftarrow$ Número de objetivos
4: $CD \leftarrow 0$
5: **for** $m \leftarrow 1$ hasta M **do**
6: Ordenar la población P según el valor de función objetivo m
7: $CD[1] \leftarrow CD[N] \leftarrow \infty$
8: **for** $i \leftarrow 2$ hasta $N - 1$ **do**
9: $CD[j] \leftarrow CD[j] + (F_m[j+1] - F_m[j-1])/(F_m[MAX] - F_m[MIN])$
10: **end for**
11: **end for**

5.2.2.7 Ordenamiento no-dominado El ordenamiento de no-dominancia se trata de ordenar una población de tamaño N comparando cada solución con las demás soluciones de la población. En la primer etapa, se encuentran todos los individuos de la primera frontera, para poder encontrar las soluciones de la siguiente frontera, se eliminan temporalmente las soluciones de la frontera anterior y se vuelven a comparar las soluciones con todas las demás soluciones de la población. Este procedimiento se repite hasta encontrar todas las fronteras posibles en la población[49].

Para este ordenamiento tomaremos en cuenta el rango en el que se encuentre la solución, asi como su distancia de aglomeración. De tal manera que una solucion dominara a otra si:

Definición 5.2. Dadas dos soluciones A y B (A $\neq$ B), preferimos A si:

$$\text{Rango\_A} < \text{Rango\_B } OR \text{ (Rango\_A} = \text{Rango\_B } AND \text{ CD\_A > CD\_B)}$$

Siendo CD la distancia de aglomeración de una solución

La complejidad de este algoritmo en el caso donde solo haya un elemento por frontera es de $O(MN^2)$

A continuación se presenta el pseudo-código de la función para el ordenamiento no dominado (Algoritmo 6):

Algorithm 6 Ordenamiento no-dominado

```
1: P ← Poblacion
2: N ← Tamao de la poblacin P
3: for i ← 1 to N do
4:    A ← Solucion[i]
5:    for j ← 1 to N do
6:       B ← Solucion[j]
7:       if Rango(A) < Rango(B) then
8:          A domina a B
9:       else if Rango(B) < Rango(A) then
10:         B domina a A
11:      else
12:         if CD_A > CD_B then
13:            A domina a B
14:         else if CD_B > CD_A then
15:            A domina a B
16:         end if
17:      end if
18:   end for
19: end for
```

5.2.2.8 Problemas de Prueba En este apartado se pondrá a prueba el algoritmo NSGA-II descrito anteriormente. Con el objetivo de analizar el correcto funcionamiento del algoritmo. Para ello se describirán distintos problemas numericos multi-objetivo a continuación:

Fonseca El problema Fonseca se define con los siguientes parámetros:
Número de variables de la función objetivo = 3
Dominio = $[-4, 4]$
Funciones objetivo:
$f_1(x) = 1 - e^{-\sum_{i=1}^{n}(x_i - \frac{1}{\sqrt{n}})^2}$
$f_2(x) = 1 - e^{-\sum_{i=1}^{n}(x_i + \frac{1}{\sqrt{n}})^2}$
Soluciones óptimas: $x_1 = x_2 = x_3 \in [-1/\sqrt{3}, 1/\sqrt{3}]$
Los resultados los podemos observar en la Figura 21

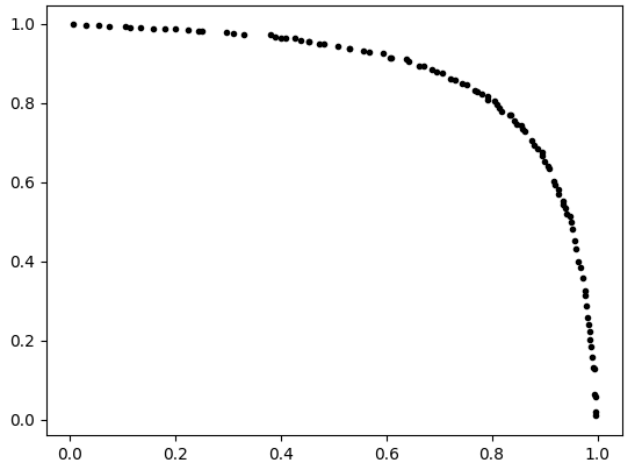

Figura 21: Ejecución del problema Fonseca en NSGA-II.

Kursawe El problema Kursawe se define con los siguientes parámetros:
Número de variables de la función objetivo = 3
Dominio = $[-5, 5]$
Funciones objetivo:
$f_1(x) = \sum_{i=1}^{n-1} \left[-10e^{-0,2\sqrt{x_i^2+x_{i+1}^2}}\right]$
$f_2(x) = \sum_{i=1}^{n} \left[|x_i|^0,8 + 5\sin(x_i^3)\right]$
Los resultados los podemos observar en la Figura 22

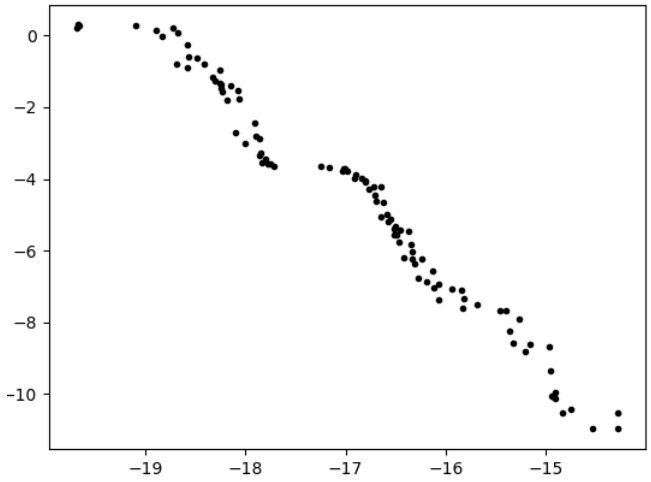

Figura 22: Ejecución del problema Kursawe en NSGA-II

ZDT1 El problema ZDT1 se define con los siguientes parámetros:
Número de variables de la función objetivo = 30
Dominio = $[0, 1]$
Funciones objetivo:

$$f_1(x) = x_1$$

$f_2(x) = g(x) \cdot [1 - \sqrt{x_1/g(x)}]$
donde:
$g(x) = 1 + 9 \cdot \frac{\sum_{i=2}^{n} x_i}{n-1}$
Soluciones óptimas: Las soluciones óptimas están en el frente de Pareto y corresponden a $x_1 \in [0, 1], x_i = 0, i = 2, ..., n$.
Los resultados los podemos observar en la Figura 23

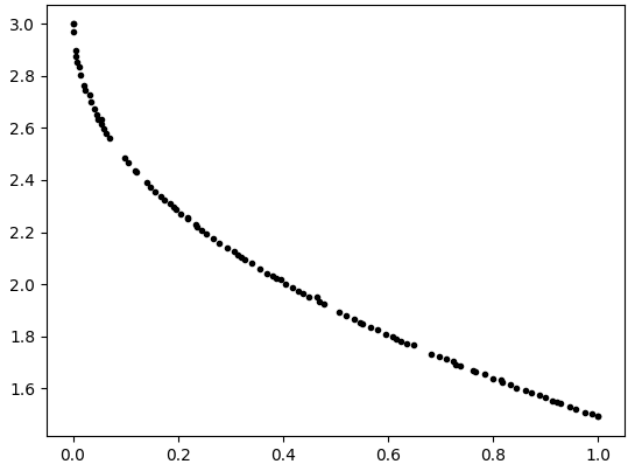

Figura 23: Ejecución del problema ZDT1 en NSGA-II.

DTLZ1 El problema DTLZ1 se define con los siguientes parámetros:
Número de variables de la función objetivo = M + K - 1
Dominio = $[0, 1]^n$
Funciones objetivo:

$$f_1(x) = \frac{1}{2} \cdot (1 + g(x_M)) \cdot x_1 \cdot x_2 \cdot ... \cdot x_{M-1}$$

$f_2(x) = \frac{1}{2} \cdot (1 + g(x_M)) \cdot x_1 \cdot x_2 \cdot ... \cdot (1 - x_{M-1})$

$$f_{M-1}(x) = \frac{1}{2} \cdot (1 + g(x_M)) \cdot x_1 \cdot (1 - x_2) \cdot ... \cdot (1 - x_{M-1})$$

...

$$f_M(x) = \frac{1}{2} \cdot (1 - x_1) \cdot (1 + g(x_M))$$

donde:
$g(x_M) = \sum_{i=M}^{M+K-1} (x_i - 0{,}5)^2 - \cos(20\pi(x_i - 0{,}5))$
Soluciones óptimas: Las soluciones óptimas están en el frente de Pareto y corresponden a $x_1 = x_2 = ... = x_n = 0{,}5$. Los resultados los podemos observar en la Figura 24

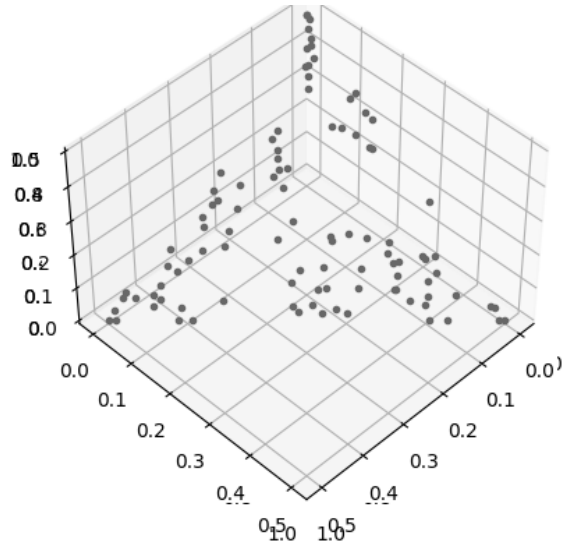

Figura 24: Ejecución del problema DLTZ1 en NSGA-II.

5.2.3. Conexión de módulos

Establecer la conexión entre una aplicación consintió en modificar la función fitness en donde se evalúan los individuos para obtener el desempeño de cada uno en la simulación mediante la obtención de las tres métricas utilizadas para medir la calidad del servicio. Estos valores se guardan en una matriz temporal para posteriormente ser evaluados por la función de optimización multi-objetivo, lo que regresa la función es un stack con los valores que serán interpretados para determinar si el rendimiento fue bueno o malo comparado con la media establecida en el RFC [5] de OLSR.

A continuación se plantea el modelado UML describiendo la interacción entre los elementos principales del marco de trabajo, en la figura 25 se muestra el diagrama de secuencia desde el punto en que se inicia el experimento, mostrando los llamados a funciones por un ciclo del algoritmo NSGA-II. En la figura 26 se plantea la entrada y salida de datos que involucra la modificación de la cabecera de OLSR, que es OLSR.h, ya que en este archivo es donde se modifican los parámetros de configuración del protocolo. La importancia de verificar el cierre del archivo radica en su uso durante la ejecución de ns-2.

El simulador se ejecuta como una instrucción mas dentro de la función fines definiendo un patrón de ejecución que ejecuta un sub-proceso, por lo que no se crean conflictos con otras instrucciones para la ejecución del algoritmo. De esta manera se evita la construcción de una clase externa y la ejecución del simulador se limita a una sola instrucción.

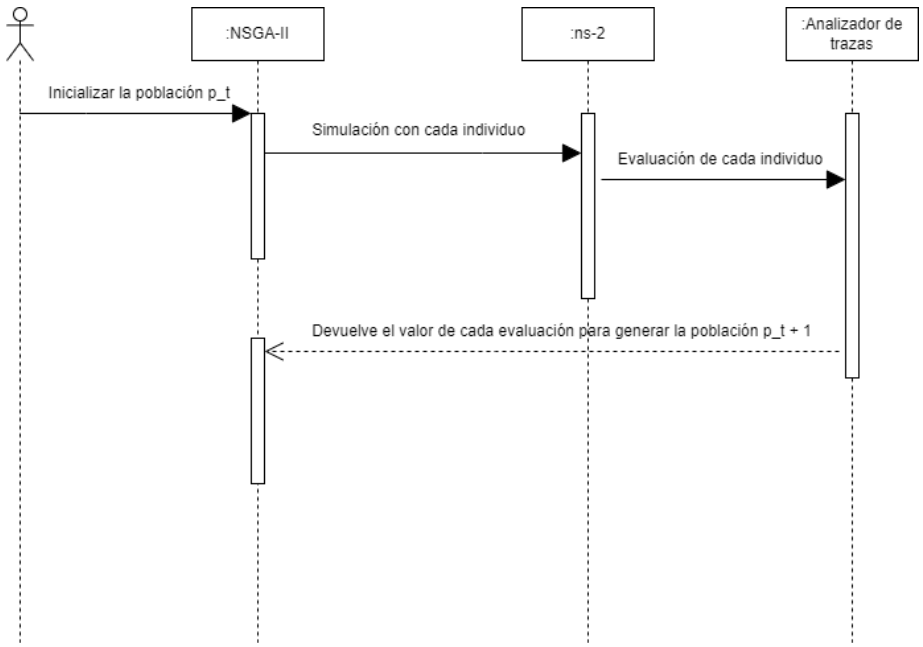

Figura 25: Diagrama de secuencia de los elementos principales del marco de trabajo experimental.

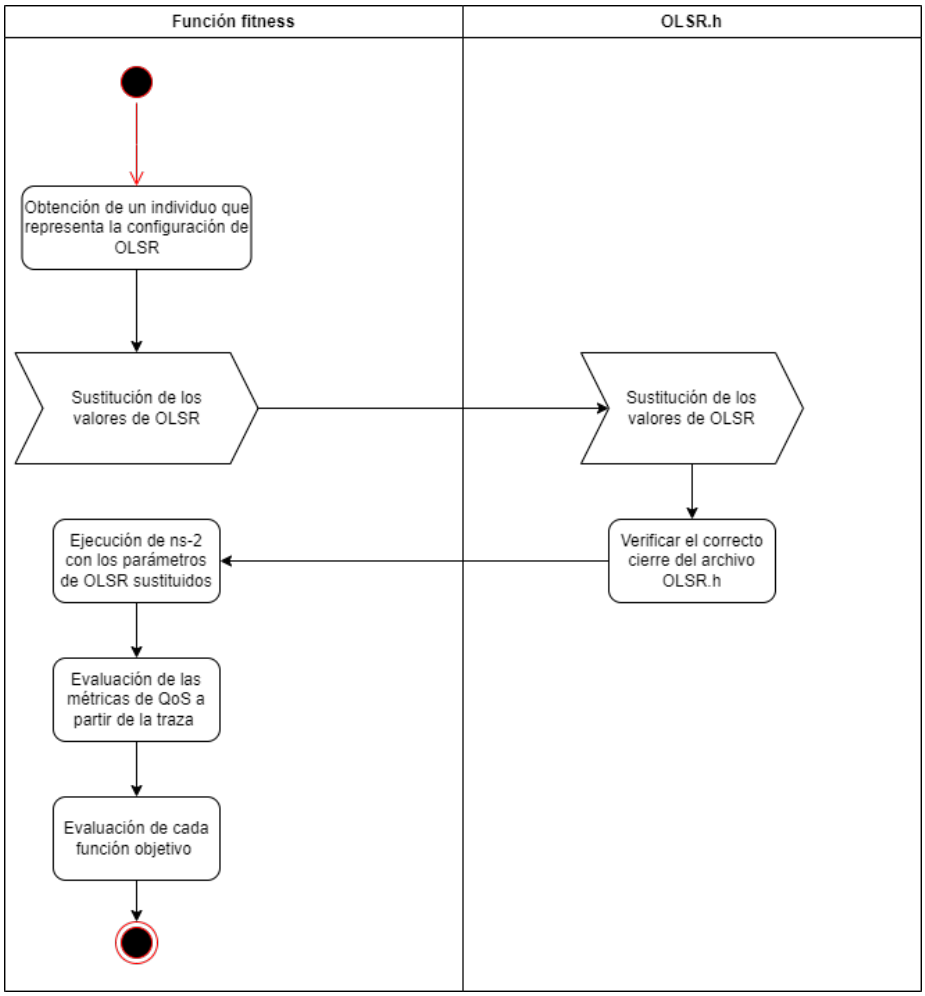

Figura 26: Diagrama de actividades de la operación de la función fitness en el algoritmo NSGA-II.

5.2.4. Resultados numéricos

Una vez configurada la conexión entre elementos, el marco de trabajo experimental esta listo para ejecutar el experimento con el escenario urbano propuesto en la sección (5.2.1). Las modificaciones propuestas a los mecanismos de cruza y mutación descritos en la sección (5.2.2), el propósito de estos cambios es mantener la uniformidad y la consistencia en cada gen de los individuos, ya que cada gen tiene un rango de valores especifico que debe mantenerse.

Se realizo una ejecución independiente de NSGA-II en el sistema operativo virtualizado (5.1.1), se contemplaron 64 generaciones con una población inicial

p_t de 8 individuos. El promedio de tiempo que tomo ejecutar cada generación tuvo variaciones y no fue uniforme, siendo el tiempo total de ejecución de las 64 generaciones un aproximado de 15 horas.

Se genero una evaluación en la función de optimización por cada individuo. La ultima generación es el conjunto de soluciones que se presenta como resultado final para un análisis de resultados. La figura 27 muestra los resultados en la ejecución de la ultima generación de NSGA-II, en la parte superior se muestran los individuos de la generación, en la parte inferior se muestran los valores de la evaluación en la función objetivo de cada individuo respectivamente. Sin embargo el valor real de PDR y NRL se obtienen restando a 1 el valor de los dos primeros elementos de cada lista.

```
[[ 5.0190877 11.0827    13.0827     0.        49.6065    16.1366
  62.4955    14.2771   ]
 [ 5.0190877 11.0827    13.0827     0.        49.6065    16.1366
  62.4955    14.2771   ]
 [ 5.0190877 11.0827    13.0827     0.        49.6065    16.1366
  62.4955    14.2771   ]
 [ 4.3775    11.2271    15.0827     1.        49.6065    15.1366
  62.4955    11.2771   ]
 [ 4.3775    11.2271    15.0827     1.        49.6065    15.1366
  62.4955    11.2771   ]
 [ 4.3775    11.2271    15.0827     1.        49.6065    15.1366
  62.4955    11.2771   ]
 [ 4.0190877 11.0827    15.0827     0.        49.6065    15.1366
  62.4955    12.2771   ]
 [ 4.0190877 11.0827    15.0827     0.        49.6065    15.1366
  62.4955    12.2771   ]
 [ 4.0190877 11.0827    15.0827     0.        49.6065    15.1366
  62.4955    12.2771   ]
 [ 4.0190877 11.0827    15.0827     7.        59.6065    15.1366
  62.4955    11.2771   ]
 [ 4.0190877 11.0827    15.0827     7.        59.6065    15.1366
  62.4955    11.2771   ]
 [ 4.0190877 11.0827    15.0827     7.        59.6065    15.1366
  62.4955    11.2771   ]
 [ 4.0190877 11.0827    15.0827     4.        49.6065    17.1366
  62.4955    11.2771   ]
 [ 4.0190877 11.0827    15.0827     5.        49.6065    15.1366
  62.4955    11.2771   ]
 [ 4.0190877 11.0827    15.0827     1.        49.6065    15.1366
  63.4955    11.2771   ]
 [ 4.0190877 11.0827    15.0827     1.        49.6065    15.1366
  62.4955    11.2771   ]]
[[ 0.3299809   0.23604026 33.22812   ]
 [ 0.3299809   0.23604026 33.22812   ]
 [ 0.3299809   0.23604026 33.22812   ]
 [ 0.3899809   0.4360213  36.90696   ]
 [ 0.3899809   0.4360213  36.90696   ]
 [ 0.3899809   0.4360213  36.90696   ]
 [ 0.3790827   0.46025    35.135611  ]
 [ 0.3790827   0.46025    35.135611  ]
 [ 0.3790827   0.46025    35.135611  ]
 [ 0.47237     0.434609   36.822681  ]
 [ 0.47237     0.434609   36.822681  ]
 [ 0.47237     0.434609   36.822681  ]
 [ 0.4320001   0.414239   36.822681  ]
 [ 0.47237     0.434609   36.822681  ]
 [ 0.32411087  0.233771   33.389443  ]
 [ 0.32411087  0.233771   33.389443  ]]
doom@doom:~/scenarios$
```

Figura 27: Resultados de la ultima generación de NSGA-II para el problema de optimización.

A continuación se muestra la codificación de cada individuo candidato a ser una configuración de OLSR, en donde cada elemento de $\vec{X}$ es el valor de un parámetro de configuración de OLSR mostrado en la tabla (2) y su respectivo valor de cada métrica de QoS obtenida de los resultados mostrados en la figura

X_0	X_1	X_2	X_3	X_4	X_5	X_6	X_7	**PDR**	**NRL**	**E2ED**
5.0190877	11.0827	13.0827	0.0	49.6065	16.1366	62.4955	14.2771	0.6700191	0.76395974	33.22812
5.0190877	11.0827	13.0827	0.0	49.6065	16.1366	62.4955	14.2771	0.6700191	0.76395974	33.22812
5.0190877	11.0827	13.0827	0.0	49.6065	16.1366	62.4955	14.2771	0.6700191	0.76395974	33.22812
4.3775	11.2271	15.0827	1.0	49.6065	15.1366	62.4955	11.2771	0.6700191	0.76395974	33.22812
4.3775	11.2271	15.0827	1.0	49.6065	15.1366	62.4955	11.2771	0.6100191	0.5639787	36.90696
4.3775	11.2271	15.0827	1.0	49.6065	15.1366	62.4955	11.2771	0.6100191	0.5639787	36.90696
4.0190877	11.0827	15.0827	0.0	49.6065	15.1366	62.4955	12.2771	0.6209173	0.53975	35.135611
4.0190877	11.0827	15.0827	0.0	49.6065	15.1366	62.4955	12.2771	0.6209173	0.53975	35.135611
4.0190877	11.0827	15.0827	0.0	49.6065	15.1366	62.4955	12.2771	0.6209173	0.53975	35.135611
4.0190877	11.0827	15.0827	7.0	59.6065	15.1366	62.4955	11.2771	0.6209173	0.53975	35.135611
4.0190877	11.0827	15.0827	7.0	59.6065	15.1366	62.4955	11.2771	0.52763	0.565391	36.822681
4.0190877	11.0827	15.0827	7.0	59.6065	15.1366	62.4955	11.2771	0.52763	0.565391	36.822681
4.0190877	11.0827	15.0827	4.0	49.6065	17.1366	62.4955	11.2771	0.5679999	0.585761	36.822681
4.0190877	11.0827	15.0827	5.0	49.6065	15.1366	62.4955	11.2771	0.52763	0.565391	36.822681
4.0190877	11.0827	15.0827	1.0	49.6065	15.1366	63.4955	11.2771	0.67588913	0.766229	33.389443
4.0190877	11.0827	15.0827	1.0	49.6065	15.1366	62.4955	11.2771	0.67588913	0.766229	33.389443

Cuadro 4: Valores de las métricas de QoS para cada individua de la ultima generación que se produjo con el algoritmo NSGA-II.

6. Conclusiones parciales

Tras investigar y analizar los diferentes conceptos como: los protocolos de encaminamiento, los POM y los AEMO se ha logrado tener un mejor entendimiento acerca de la optimizacioń del protocolo OLSR mediante la mejora del rendimiento de sus parámetros como consecuencia del uso NSGA-II. Hasta este punto del proyecto, hemos logrado obtener datos que se aproximan a los que se reportan en la publicación de Toutouh y Alba [4], misma que se tiene como referencia base y que se ha buscado reproducir en esta etapa del proyecto. Con base en esto, afirmamos que se han obtenido resultados favorables (ver Cuadro 4). Es importante mencionar que los resultados pueden variar dependiendo de los detalles de las implementaciónes, así como las configuraciones en el simulador y el analizador de trazas que se use. En nuestro caso particular, el analizador de trazas es un desarrollo propio. Este se mantiene en desarrollo y es posible que al ser la primera versión se estén omitiendo algunas consideraciones al leer la traza y que afecten en el resultado final de las métricas.

Sin embargo, según las mejoras en los parámetros de configuración del protocolo OLSR, se ha podido demostrar que el algoritmo NSGA-II es un buen punto de comparación para comenzar la siguiente fase del proyecto en donde se tratará el POM propuesto en la Sección 4. Aun se requiere completar la fase de experimentación propuesta por Toutouh y Alba [4], sin embargo el tener ya un primer modelo de la infraestructura de trabajo permitirá que el desarrollo del trabajo terminal continué de la manera en que se planteó.

Referencias

[1] G. M. Carles and P. A. Josep, "Redes ad-hoc: El prÓximo reto," 07 2004.

[2] S. Malik and P. Kumar Sahu, "A comparative study on routing protocols for vanets," 08 2019.

[3] D. Patel, M. F. Khan, P. Batavia, S. Makhjia, and M. Roja, "Overview of routing protocols in vanet," *International Journal of Computer Applications*, vol. 136, February 2016.

[4] J. Toutouh and E. Alba, "Multi-objective olsr optimization for vanets," 2012.

[5] T. Clausen and P. Jacquet, "Optimized link state routing protocol (olsr)," RFC 3626, Internet Engineering Task Force, October 2003.

[6] C. A. Coello Coello, G. Lamont A., and D. A. Van Veldhuizen, *Evolutionary Algorithms for Solving Multi-Objective Problems*. New York: Springer, 2007. ISBN: 978-0-387-33254-3.

[7] E. Zitzler, K. Deb, and L. Thiele, "Comparison of multiobjective evolutionary algorithms: Empirical results," *Evolutionary Computation*, vol. 8, no. 2, pp. 173–195, 2000.

[8] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Transactions on Evolutionary Computation*, vol. 1, pp. 67–82, April 1997.

[9] B. Li, J. Li, K. Tang, and X. Yao, "Many-Objective Evolutionary Algorithms: A Survey," *ACM Computing Surveys*, vol. 48, September 2015.

[10] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A Fast and Elitist Multiobjective Genetic Algorithm: NSGA–II," *IEEE Transactions on Evolutionary Computation*, vol. 6, pp. 182–197, April 2002.

[11] Q. Zhang and H. Li, "MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition," *IEEE Transactions on Evolutionary Computation*, vol. 11, pp. 712–731, December 2007.

[12] R. Hernández Gómez and C. A. Coello Coello, "A Hyper-Heuristic of Scalarizing Functions," in *2017 Genetic and Evolutionary Computation Conference (GECCO'2017)*, (Berlin, Germany), pp. 577 –584, ACM Press, July 15-19 2017. ISBN 978-1-4503-4920-8.

[13] N. Beume, B. Naujoks, and M. Emmerich, "SMS-EMOA: Multiobjective selection based on dominated hypervolume," *European Journal of Operational Research*, vol. 181, pp. 1653–1669, 16 September 2007.

[14] J. G. Falcón-Cardona and C. A. Coello Coello, "A Multi-Objective Evolutionary Hyper-Heuristic Based on Multiple Indicator-Based Density Estimators," in *2018 Genetic and Evolutionary Computation Conference (GECCO'2018)*, (Kyoto, Japan), pp. 633–640, ACM Press, July 15–19 2018. ISBN: 978-1-4503-5618-3.

[15] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks," *IEEE Communications Magazine*, vol. 40, no. 8, pp. 102–114, 2002.

[16] I. Akyildiz and X. Wang, "A survey on wireless mesh networks," *IEEE Communications Magazine*, vol. 43, no. 9, pp. S23–S30, 2005.

[17] J. García, M. Rodríguez, P. López, and F. Martínez, "A comparative study of static and dynamic vanets for intelligent transportation systems," *IEEE Transactions on Intelligent Transportation Systems*, vol. 27, no. 3, pp. 1432–1443, 2021.

[18] A. Laouiti, A. Qayyum, and M. Naufal, *Vehicular AdHoc Networks for Smart Cities*. Singapore, Pte.: Springer, 2017. ISBN: 978-981-10-3502-9.

[19] C. Crespo Cintas, "Integración de protocolos de acceso avanzados en redes wlan ieee 802.11," 2008.

[20] J.-P. Vasseur and A. Dunkels, *Interconnecting Smart Objects with IP The Next Internet*. San Francisco, CA.: Morgan Kaufmann, 2010. ISBN: 978-0-12-375165-2.

[21] G. Karagiannis, O. Altintas, E. Ekici, G. Heijenk, B. Jarupan, K. Lin, and T. Weil, "Vehicular networking: A survey and tutorial on requirements, architectures, challenges, standards and solutions," *IEEE Communications Surveys Tutorials*, vol. 13, no. 4, pp. 584–616, 2011.

[22] N. H. Hussein, C. T. Yaw, S. P. Koh, S. K. Tiong, and K. H. Chong, "A comprehensive survey on vehicular networking: Communications, applications, challenges, and upcoming research directions," *IEEE Access*, vol. 10, pp. 86127–86180, 2022.

[23] Ø. Risan and E. Peytchev, "A vehicle-to-vehicle communication protocol for collaborative identification of urban traffic conditions," in *Ad Hoc Networks* (J. Zheng, D. Simplot-Ryl, and V. C. M. Leung, eds.), (Berlin, Heidelberg), pp. 482–494, Springer Berlin Heidelberg, 2010.

[24] M. Jerbi, P. Marlier, and S. M. Senouci, "Experimental assessment of v2v and i2v communications," in *2007 IEEE International Conference on Mobile Adhoc and Sensor Systems*, pp. 1–6, 2007.

[25] H. P. Dai Nguyen and R. Zoltán, "The current security challenges of vehicle communication in the future transportation system," in *2018 IEEE 16th International Symposium on Intelligent Systems and Informatics (SISY)*, pp. 000161–000166, 2018.

[26] H. Zhang and X. Lu, "Vehicle communication network in intelligent transportation system based on internet of things," *Computer Communications*, vol. 160, pp. 799–806, 2020.

[27] K. Ullah, *On the use of opportunistic vehicular communication for roadside services advertisement and discovery*. PhD thesis, 09 2016.

[28] J. A. Arizaga Silva, M. Alonso Pérez, and R. Gonzalez, "Redes vanet vehicular ad-hoc networks, la conectividad de los autos. primera parte.," vol. 12, 09 2018.

[29] M. Maad Hamdi, L. Audah, S. Abduljabbar Rashid, A. Hamid Mohammed, S. Alani, and A. Shamil Mustafa, "A review of applications, characteristics and challenges in vehicular ad hoc networks (vanets)," in *2020 International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA)*, pp. 1–7, 2020.

[30] S. Hayat, E. Yanmaz, and R. Muzaffar, "Survey on unmanned aerial vehicle networks for civil applications: A communications viewpoint," *IEEE Communications Surveys Tutorials*, vol. 18, no. 4, pp. 2624–2661, 2016.

[31] S. Singh and S. Agrawal, "Vanet routing protocols: Issues and challenges," pp. 1–6, 2014.

[32] Cisco, "Dynamic routing protocols: Ospf, eigrp, ripv2, is-is, bgp." `https://community.cisco.com/t5/networking-knowledge-base/dynamic-routing-protocols-ospf-eigrp-ripv2-is-is-bgp/ta-p/4511577#toc-hId--1136496333`, 2021. [Acceso: 28 de abril de 2023].

[33] J. Santa, M. Tsukada, T. Ernst, O. Mehani, and A. Skarmeta, "Assessment of vanet multi-hop routing over an experimental platform," *IJIPT*, vol. 4, pp. 158–172, 09 2009.

[34] S. Zeadally, R. Hunt, Y.-S. Chen, A. Irwin, and A. Hassan, "Vehicular ad hoc networks (vanets): Status, results, and challenges," *Telecommunication Systems*, vol. 50, pp. 1–25, 08 2010.

[35] A. V. Leonov and G. A. Litvinov, "About applying aodv and olsr routing protocols to relaying network scenario in fanet with mini-uavs," in *2018 XIV International Scientific-Technical Conference on Actual Problems of Electronics Instrument Engineering (APEIE)*, pp. 220–228, 2018.

[36] N. Harrag and H. Adbelghani, "Bio-inspired olsr routing protocol," 04 2019.

[37] J. Härri, F. Filali, and C. Bonnet, "Performance comparison of aodv and olsr in vanets urban environments under realistic mobility patterns," 01 2006.

[38] "Using build-essentials." Accedido el 23 de mayo de 2023.

[39] "Dépôts bionic." `https://doc.ubuntu-fr.org/depots_bionic`. Accedido el 23 de mayo de 2023.

[40] "keyserver for gpg." Accedido el 23 de mayo de 2023.

[41] "Um-olsr for ns2." https://sourceforge.net/projects/um-olsr/reviews/. Accedido el 23 de mayo de 2023.

[42] "User information for ns-2." https://nsnam.sourceforge.net/wiki/index.php/User_Information. Accedido el 23 de mayo de 2023.

[43] "nsnam files." https://sourceforge.net/projects/nsnam/files/allinone/ns-allinone-2.35/. Accedido el 23 de mayo de 2023.

[44] "ns error." https://www.math.unipd.it/~cpalazzi/resources/README.txt. Accedido el 23 de mayo de 2023.

[45] "Sumo." https://sumo.dlr.de/docs/index.html. Accedido el 23 de mayo de 2023.

[46] "Opne street map." https://sumo.dlr.de/docs/Networks/Import/OpenStreetMap.html. Accedido el 23 de mayo de 2023.

[47] "Cbr traffic in ns2." https://ns2simulator.com/cbr-traffic-in-ns2/. Accedido el 23 de mayo de 2023.

[48] "How to do data transmission between the nodes using udp in ns2?." https://slogix.in/source-code/ns2-tutorials/how-to-do-data-transmission-between-the-nodes-using-udp-in-ns2/. Accedido el 23 de mayo de 2023.

[49] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan, "A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: Nsga-ii," in *Parallel Problem Solving from Nature PPSN VI* (M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J. J. Merelo, and H.-P. Schwefel, eds.), (Berlin, Heidelberg), pp. 849–858, Springer Berlin Heidelberg, 2000.

Lista de tablas

Lista de figuras