Búsqueda de vecindad variable para el problema del conjunto *k*-dominante mínimo

Raúl Fauste-Jiménez[1][0009−0009−9828−3558],
Isaac Lozano-Osorio[1][0000−0002−2608−8464],
Jesús Sánchez-Oro[1][0000−0003−1702−4941] y
Óscar Cordón[2][0000−0001−5112−5629]

[1] GRAFO Research, Departamento de Informática y Estadística, Universidad Rey Juan Carlos, C/Tulipán S/N, Móstoles, 28933, Madrid, España
{raul.fauste,isaac.lozano,jesus.sanchezoro}@urjc.es
[2] Dept. de Informática e Inteligencia Artificial e Instituto Andaluz de Investigación en Ciencia de Datos e Inteligencia Computacional, Universidad de Granada, 18071, España
ocordon@decsai.ugr.es

Resumen El problema del conjunto k-dominante mínimo es un problema de optimización combinatoria $\mathcal{NP}$-Difícil. Las aplicaciones en el mundo real van desde las redes de comunicación hasta problemas de localización de instalaciones. Dado un grafo, el objetivo es encontrar un subconjunto de vértices dominantes con mínima cardinalidad, de forma que todo vértice que no pertenezca a este subconjunto sea adyacente al menos a k vértices dominantes. Dada la complejidad del problema, se propone una metaheurística para su resolución, una Búsqueda de Vecindad Variable (VNS). Tanto las decisiones tomadas a lo largo de este trabajo, como las componentes del algoritmo, han sido validadas mediante experimentos computacionales. Además, se incluye una comparativa con la propuesta más reciente de la literatura: *Restart Local Search with Relaxed Configuration Checking* (RLS_RCC). El VNS propuesto obtiene una desviación promedio de 0.34 % demostrando ser una aproximación competitiva para resolver el problema del conjunto k-dominante mínimo.

Keywords: Minimum k-Dominating Set · VNS · Metaheurísticas · Optimización combinatoria · Investigación Operativa.

1. Introducción

El problema del conjunto k-dominante mínimo (MKDS, del inglés *Minimum k-Dominating Set*) es un problema de optimización combinatoria $\mathcal{NP}$-Difícil. Se define un grafo $G = (V, E)$ donde V es el conjunto de vértices y E el de las aristas, junto con un entero $k > 0$. El objetivo del problema es encontrar un subconjunto $D_k \subseteq V$ con mínima cardinalidad tal que todo vértice $v \in V$ verifique al menos una de las siguientes condiciones: todo vértice debe pertenecer al conjunto dominante D_k, es decir, $v \in D_k$; o estar conectado con al menos k

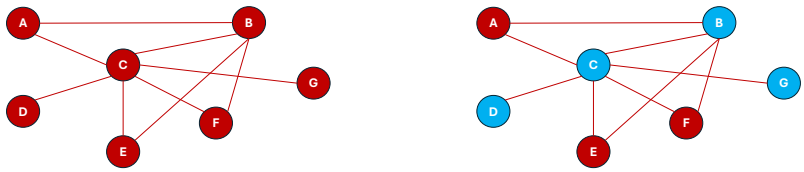
vértices del conjunto dominante, es decir, $|N(v) \cap D_k| \geq k$. Si un vértice cumple la segunda condición se dice que está k-dominado. Formalmente,

$$\begin{aligned} \text{mín} \quad & \sum_{i \in V} x_i \\ \text{s.a.} \quad & \sum_{j \in N(i)} x_j \geq k(1 - x_i), \forall i \in V \\ & x_i \in \{0, 1\}, \forall i \in V \end{aligned}$$

donde x_i es una variable binaria que toma el valor 1 si el vértice i pertenece al conjunto dominante, $i \in D_k$, y el valor 0 en otro caso. Además, $N(i)$ denota el conjunto de vértices adyacentes al vértice $i \in V$, es decir,

$$N(i) = \{u \in V : (u, i) \in E\}$$

Una solución al problema MKDS viene dada por la selección de un conjunto dominante. Es decir, basta con indicar los vértices que pertenecen a D_k. En la Figura 1 se detalla una solución factible, $D_k = \{\texttt{B}, \texttt{C}, \texttt{D}, \texttt{G}\}$ para el grafo descrito, considerando un valor de $k = 2$.



(a) $D_k = \{\}$ (b) $D_k = \{\texttt{B}, \texttt{C}, \texttt{D}, \texttt{G}\}$

Figura 1: Instancia del problema MKDS (Figura 1a) y propuesta de solución factible (Figura 1b).

En este caso, los vértices que forman parte de la solución: $\{\texttt{B}, \texttt{C}, \texttt{D}, \texttt{G}\}$, ya satisfacen la primera restricción del problema por pertenecer al conjunto dominante. Para que el resto de vértices satisfagan la segunda restricción, deberían estar conectados a, al menos $k = 2$ vértices del conjunto dominante. En este caso, los vértices `A`, `E` y `F` están conectados a los vértices `B` y `C` que pertenecen a D_k.

El problema MKDS es una variación del problema clásico del conjunto dominante mínimo (MDS, del inglés *Minimum Dominating Set*). La versión clásica ha sido ampliamente estudiada [3,5] y tiene múltiples aplicaciones en varios campos, como redes sociales [2] o redes inalámbricas [1]. Nótese que siguiendo la definición del problema MKDS, si $k = 1$, el problema a resolver se convierte en el MDS. El objetivo de este trabajo es cubrir el caso donde $k > 1$, por lo que el problema clásico no se considera en el estudio.

Sin embargo, el problema MKDS es de gran interés ya que sus soluciones proporcionan redes robustas y resilientes a fallos. Al igual que la versión clásica, tiene numerosas aplicaciones en el mundo real, desde usos en redes de comunicación hasta problemas de localización de instalaciones [6,7].

En la literatura se encuentran múltiples propuestas para la resolución del problema MKDS, desde búsquedas heurísticas [6] hasta las propuestas más recientes donde se proponen estrategias de resolución más avanzadas. En [9] proponen una estrategia novedosa de selección de vértices combinada con una búsqueda local. La estrategia de selección se basa en una puntuación que se asigna a cada vértice en función del tiempo que el vértice y sus vecinos pasan sin estar k-dominados. La búsqueda local, continúa utilizando la puntuación de cada vértice para determinar qué movimiento de los propuestos es el elegido en cada momento. Los movimientos considerados consisten en eliminar o añadir vértices de forma aleatoria, o en función de la puntuación de cada vértice. Por otro lado, el trabajo más reciente de la literatura, desarrollado por los mismos autores, propone un algoritmo multiarranque denominado *Restart Local Search with Relaxed Configuration Checking* (RLS_RCC) [10]. La construcción de la solución inicial consiste en, partiendo de una solución vacía, seleccionar el vértice con mayor puntuación y, en caso de empate, el vértice con el mínimo número de vecinos. En caso de un nuevo empate, se resuelve aleatoriamente, permitiendo así un método multiarranque. Además, ambos trabajos proponen un mecanismo para evitar el problema de los ciclos en la búsqueda local, es decir, volver a visitar soluciones previamente vistas. Para ello, almacenan el estado de cada vértice y su vecindario de primer nivel [9], o hasta su vecindario de segundo nivel [10]. Se considera que una solución ya se ha visitado si para todo vértice, su estado y el de su vecindario (de primer o segundo nivel) se ve sin alterar. Al considerar vecindarios de segundo nivel, se detectan más alteraciones en la solución, ya que se tiene en consideración un mayor número de vecinos, permitiendo así una mayor diversidad en la exploración de soluciones. Esta información se utiliza en la búsqueda local para determinar con qué vértices operar los movimientos.

En este trabajo se propone una aproximación metaheurística para el MKDS, una Búsqueda de Vecindad Variable (VNS) [13]. El documento se organiza como sigue. Primero, la propuesta algorítmica se describe en la Sección 2. A continuación, el análisis experimental se puede encontrar en la Sección 3. Por último, las conclusiones y trabajos futuros se discuten en la Sección 4.

2. Propuesta algorítmica

En este trabajo se propone una Búsqueda de Vecindad Variable (VNS, del inglés *Variable Neighborhood Search*), una metaheurística trayectorial que originalmente se introdujo en [13] y ha demostrado un rendimiento competitivo en diversos problemas de optimización combinatoria [4,11].

El algoritmo propuesto comienza construyendo una solución inicial (D_k), con el procedimiento descrito en la Sección 2.1. A continuación, de forma iterativa hasta cumplir un criterio de parada, se perturba la mejor solución incumbente,

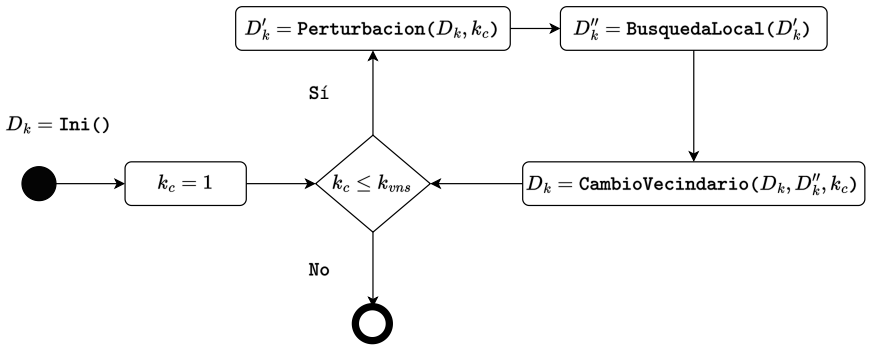


Figura 2: Diagrama de flujo del VNS.

D_k, para escapar de la región actual del espacio de búsqueda, con el mecanismo descrito en la Sección 2.4. La solución resultante D'_k se somete a una fase de mejora, hasta alcanzar un mínimo local D''_k (Sección 2.3). El método `CambioVecindario` evalúa cada mínimo local encontrado frente a la mejor solución incumbente, actualizando la mejor solución incumbente en caso de encontrar una solución diferente de igual o menor tamaño, por razones de diversidad. Adicionalmente, el método vuelve a empezar la búsqueda desde el primer vecindario ($k_c = 1$) si la solución actual mejora la mejor solución incumbente, pasando al siguiente vecindario en caso contrario, $k_c = k_c + k_{step}$, donde k_{step} es el tamaño del incremento de vecindad, que se estudiará en la experimentación. Por último, se devuelve la mejor solución encontrada una vez alcanzado el criterio de parada.

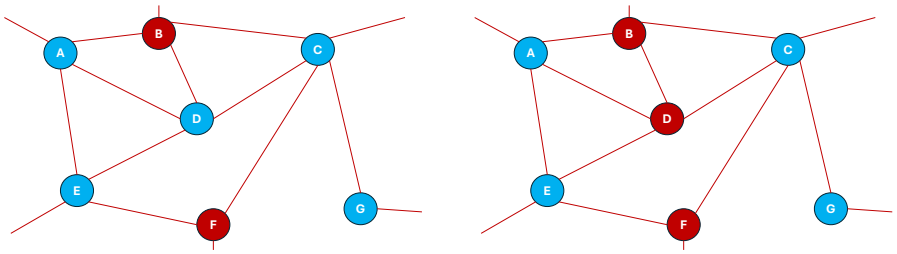
2.1. Construcción de la solución inicial

La metodología VNS asegura que la elección del punto de partida no afecta a la calidad de la solución final encontrada. Sin embargo, proporcionar un punto de partida prometedor hace que el algoritmo converja antes, por lo que en este trabajo se opta por una solución inicial de alta calidad. La estrategia elegida consiste en un método voraz basado en un criterio de cobertura. Se comienza con el conjunto dominante vacío, $D_k = \{\}$. Iterativamente, se añade el vértice con mayor cobertura (rompiendo empates de forma aleatoria). La cobertura de un vértice se define como el número de vértices que se volverán k-dominados si v es añadido al conjunto dominante D_k. Nótese que los vértices con menor grado que k se incluyen directamente en la solución ya que es imposible k-dominarlos en otro caso. El procedimiento termina cuando se obtiene una solución factible, es decir, todos los vértices o bien pertenecen a D_k o bien están k-dominados.

2.2. Post-proceso

La solución resultante del procedimiento de construcción de soluciones puede contener vértices redundantes. Un vértice es redundante si $v \in D_k$ y es adyacente

al menos a k vértices de D_k (es decir, está k-dominado), verificando además que $\forall w \in N(v) \backslash D_k$, w es adyacente al menos a $k+1$ vértices de D_k. Esto se debe a que un procedimiento voraz toma decisiones inmediatas, sin tener en cuenta el efecto producido a largo plazo sobre la solución. Para corregir este comportamiento, se define un método a modo de post-proceso para reducir el tamaño de la solución, eliminando estos vértices redundantes.



(a) Vértice $\texttt{D} \in D_k$ es redundante. (b) Vértice $\texttt{D}$ se elimina de D_k.

Figura 3: Ejemplo de post-proceso, con $k = 2$, sobre la solución $D_k = \{\texttt{A}, \texttt{C}, \texttt{D}, \texttt{E}, \texttt{G}\}$, donde el vértice $\texttt{D}$ es redundante.

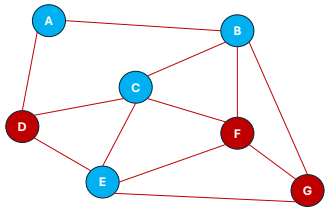
La Figura 3 muestra cómo funciona el post-proceso descrito. La Figura 3a muestra una solución factible: $D_k = \{\texttt{A}, \texttt{C}, \texttt{D}, \texttt{E}, \texttt{G}\}$. El vértice $\texttt{D}$ es adyacente a: $\texttt{A}, \texttt{E}, \texttt{C} \in D_k$, luego está k-dominado. Por otro lado, el vecino de $\texttt{D}$ que no pertenece a D_k, es el vértice $\texttt{B}$, que es adyacente a: $\texttt{A}, \texttt{C}, \texttt{D} \in D_k$. Por lo tanto, el vértice $\texttt{D}$ es redundante, por lo que se elimina de D_k, resultando en la solución de la Figura 3b. El método se repite hasta que no se encuentran vértices redundantes (en este caso, ya no existen vértices redundantes adicionales).

2.3. Fase de mejora

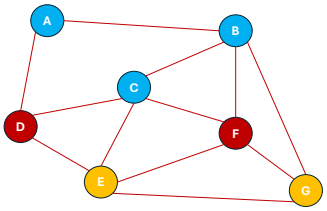
La solución resultante del post-proceso no tiene por qué ser un óptimo local respecto a ninguna vecindad, por lo que puede ser mejorada. Para ello, se propone una fase de mejora que partiendo de una solución trata de alcanzar un óptimo local con respecto a una vecindad concreta. Por norma general, se compone de tres elementos: un movimiento, el vecindario definido por el movimiento y la estrategia para explorar dicho vecindario. Se proponen dos búsquedas locales diferentes para resolver el problema MKDS, ambas basadas en la estrategia de búsqueda *First Improvement*, la cual ejecuta un movimiento cada vez que este lleva a una mejora, en lugar de explorar la vecindad completa para ejecutar el mejor movimiento. Esto favorece el rendimiento del algoritmo en términos de esfuerzo computacional sin afectar en gran medida a la calidad de las soluciones encontrada [8].

La primera búsqueda local propuesta, $EX_{1\times1}$, realiza un movimiento que consiste en intercambiar un vértice $v \in D_k$ con otro vértice $w \notin D_k$. Al seleccionar

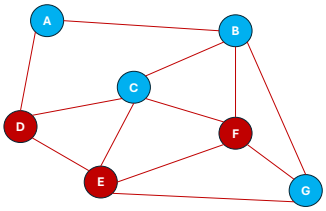
el vértice $v \in D_k$ no todo vértice $w \notin D_k$ es válido para el intercambio si se desea mantener una solución factible. Al eliminar v de D_k es posible que algún vértice perteneciente a $N(v) \cup \{v\}$ se vuelva no k-dominado. Por lo tanto, el vértice $w \notin D_k$ deberá k-dominar dichos vértices de nuevo. Por lo tanto, para que la solución siga siendo factible, w debe pertenecer a la intersección del conjunto de vértices no k-dominados después de eliminar v y su vecindario cerrado. El vecindario cerrado de un vértice $v \in V$, se denota como $N[v]$ y es aquel compuesto por los vecinos de v incluyéndose a sí mismo, es decir, $N[v] = N(v) \cup \{v\}$.



(a) Intercambio del vértice E.



(b) E, G se vuelven no k-dominados.



(c) Solución obtenida tras el intercambio.

Figura 4: Ejemplo de intercambio entre los vértices E y G.

La Figura 4 ilustra un ejemplo de intercambio. Al tratar de intercambiar el vértice E, es necesario analizar qué vértices se vuelven no k-dominados tras eliminarlo de D_k. Como se detalla en la Figura 4b, en este ejemplo, los vértices $\mathtt{E}, \mathtt{G}$ se vuelven no k-dominados. Por tanto, los vértices candidatos para el intercambio deben volver a k-dominar $\mathtt{E}, \mathtt{G}$. Los candidatos se deducen de la intersección de $\{\mathtt{E}, \mathtt{G}\}$, $N[\mathtt{E}] = \{\mathtt{C}, \mathtt{D}, \mathtt{E}, \mathtt{F}, \mathtt{G}\}$ y $N[\mathtt{G}] = \{\mathtt{B}, \mathtt{E}, \mathtt{F}, \mathtt{G}\}$, es decir $\{\mathtt{E}, \mathtt{G}\} \cap N[\mathtt{E}] \cap N[\mathtt{G}] = \{\mathtt{E}, \mathtt{G}\}$.

Como el intercambio debe ser con vértices distintos el vértice E no puede ser considerado, siendo el único candidato viable el vértice G. Por último, la solución resultante se muestra en la Figura 4c.

Nótese que ninguna solución resultante de un movimiento $EX_{1\times1}$ podrá mejorar el valor de la función objetivo. Dado que el movimiento es un intercambio, el tamaño del conjunto dominante será siempre constante. Por este motivo, es

necesario combinarla con el uso de un método que elimine vértices redundantes de D_k. Para ello, se utiliza el post-proceso descrito en la Sección 2.2.

La segunda búsqueda local propuesta trata de solventar el problema de la dependencia de un post-proceso reduciendo de forma directa la cardinalidad del conjunto dominante. Esta búsqueda, denotada por $EX_{2\times1}$, consiste en aplicar un movimiento de intercambio de dos vértices $u, v \in D_k$ por otro vértice $w \notin D_k$.

$$EX_{2\times1}(D_k, u, v, w) \equiv (D_k \backslash \{u, v\}) \cup \{w\}$$

Como ocurría en $EX_{1\times1}$, no todos los vértices $w \notin D_k$ son válidos para el intercambio. La estrategia para encontrar dichos vértices candidatos es análoga a la descrita para el movimiento $EX_{1\times1}$ con el objetivo de asegurar que la solución resultante siempre sea factible.

2.4. Perturbación

La fase de perturbación trata de escapar de óptimos locales alterando la mejor solución encontrada hasta el momento. Esta perturbación consiste en añadir vértices aleatoriamente a D_k. La cantidad de vértices a añadir viene determinada por un parámetro, k_{vns}, que se ajustará en la Sección 3. Esta fase es la responsable de la diversificación dentro de VNS. Aunque hay estudios que han analizado el efecto de incluir intensificación dentro de esta fase de perturbación con lo que se ha denominado *Intensified Shake* [14], en este trabajo se considera un mecanismo de perturbación centrado puramente en la diversificación ya que es una parte clave en el problema tratado.

3. Resultados computacionales

En la literatura [9,10] se consideran tres conjuntos de instancias: DIMACS, General Graphs (GG) y Unit Disk Graphs (UDG). Este trabajo, al ser preliminar, solo considera el conjunto GG por ser el más genérico. El conjunto GG consiste en 54 instancias, cada una con tres valores diferentes de k: k_{min}, k_{med} y k_{max}, calculados como se describe en [9] (i.e. 162 instancias en total). Todos los experimentos han sido realizados en un servidor con procesador AMD EPYC 7282, 16 cores, 32GB de RAM y Java 25.

En trabajos previos [9,10] se evalúa el comportamiento de los modelos de programación lineal entera sobre las instancias del problema utilizando para ello el solver CPLEX. En dichos estudios, CPLEX solo pudo resolver 22 de las 54 instancias garantizando optimalidad (con cada valor de k) con una restricción de tiempo de 3600 segundos. Con estas prestaciones el solver se ve superado por la mejor propuesta de la literatura [10], dejando clara la necesidad de métodos heurísticos para el MKDS, por lo que se descarta CPLEX de la comparativa.

La experimentación se divide en dos fases, una de ajuste de parámetros y otra final. La primera tiene como objetivo determinar la mejor configuración del algoritmo para obtener el mejor rendimiento posible. Por otro lado, en la

segunda fase se trata de medir el rendimiento del algoritmo comparándolo con la propuesta más reciente del estado del arte en el conjunto GG completo.

El análisis de rendimiento se realiza estudiando las siguientes métricas: F.O., promedio de la función objetivo; Desv. (%) porcentaje promedio de desviación con respecto a la mejor solución conocida; Tiempo (s), tiempo de cómputo promedio en segundos; y #Mejor, número de instancias donde el algoritmo alcanza la mejor solución conocida.

3.1. Ajuste de parámetros

El algoritmo VNS requiere la configuración de un conjunto de parámetros para obtener su mejor rendimiento. El ajuste se realizó con ayuda de la herramienta `irace` [12], un software diseñado para obtener la configuración automática de algoritmos mediante test estadísticos y aprendizaje automático. Se llevó a cabo sobre un subconjunto de 15 instancias originales (5 para cada uno de los 3 valores diferentes de k) seleccionadas por su dificultad, principalmente las instancias de mayor tamaño. Los parámetros considerados y sus dominios de búsqueda se describen a continuación:

- $\mathbf{k_{vns}}$: Parámetro utilizado para determinar el número máximo de vecindarios a explorar en la fase de perturbación. Para que el valor de este parámetro sea escalable con el tamaño de la solución, el valor será un porcentaje del número de vértices:

$$k_{vns} \in \{0.05, 0.1, 0.2, 0.3, 0.4, 0.5\}$$

- $\mathbf{k_{step}}$: Parámetro utilizado para determinar el número de vértices a añadir aleatoriamente en la fase de perturbación. Al igual que el parámetro anterior, se trata de un porcentaje del número de vértices para garantizar la escalabilidad de la propuesta (se incluye el valor 0 que representa la adición de un solo nodo):

$$k_{step} \in \{0, 0.005, 0.01, 0.015, 0.02, 0.025, 0.03\}$$

- ***EX***: Parámetro para determinar cuál de las dos búsquedas locales propuestas obtiene mejores resultados.

La mejor configuración obtenida por `irace` ha sido:

$$\{k_{vns} = 0.4, k_{step} = 0, EX = EX_{1\times1}\}$$

Una vez obtenida la mejor configuración del algoritmo propuesto, en la Sección 3.2 se comparan los resultados obtenidos contra la mejor propuesta de la literatura.

3.2. Experimentación final

En esta sección se presenta el experimento final diseñado para evaluar el rendimiento del algoritmo propuesto cuando se compara con la mejor propuesta del estado del arte: RLS_RCC [10]. La configuración del algoritmo para el experimento final es la descrita en Sección 3.1 con un criterio de parada de 6000 iteraciones. El criterio de parada se define para obtener tiempos de cómputo similares a la propuesta de la literatura. Además, tal y como se realiza en la propuesta de la literatura, se han realizado 10 ejecuciones de VNS independientes, reportando la mejor solución encontrada.

Tabla 1: VNS vs RLS_RCC

Algoritmo	Instancias	F.O.	Tiempo (s)	Desv. (%)	#Mejor
VNS	k_{min}	88.06	83.16	0.75	42
	k_{med}	164.26	**76.45**	0.23	44
	k_{max}	234.04	**15.68**	**0.02**	49
	Promedio	162.12	58.43	0.34	135
RLS_RCC	k_{min}	**87.35**	**34.66**	**0.00**	**53**
	k_{med}	**163.69**	81.32	**0.11**	**49**
	k_{max}	**234.00**	58.66	**0.02**	**51**
	Promedio	**161.68**	**58.21**	**0.05**	**153**

Se pueden extraer varias conclusiones de la Tabla 1. Primero, RLS_RCC obtiene en, prácticamente todas las instancias, el mejor valor conocido (153 de 162), reflejándose en la desviación media que consigue dicha propuesta, 0.05 %. Por otro lado, el VNS propuesto destaca por ser una alternativa competitiva (con una desviación media de 0.34 %). Conforme aumenta el tamaño de k, se observa un mejor comportamiento del VNS, obteniendo un mayor número de mejores soluciones. Por este motivo, se puede concluir que la propuesta escala mejor conforme se incrementa el valor de k (pasando de una desviación media de 0.75 % a 0.02 %). Pese a no vencer al algoritmo RLS_RCC, el VNS descrito obtiene resultados competitivos, mostrando otra aproximación competente para resolver el problema MKDS.

4. Conclusiones

Este trabajo propone una metaheurística basada en VNS para resolver el problema MKDS. El algoritmo propuesto comienza desde una solución inicial guiada por un criterio de cobertura. A continuación, se lleva a cabo una fase de perturbación para diversificar, seguida de una fase de mejora para intensificar, finalizando con un post-proceso para eliminar vértices redundantes de la solución.

Los experimentos computacionales tratan de analizar el rendimiento de la propuesta frente al mejor algoritmo de la literatura: RLS_RCC. Los resultados muestran competitividad, pese a que la propuesta previa obtiene mejores resultados.

Como futuros trabajos, conviene considerar la incorporación de componentes que potencien la diversidad del algoritmo propuesto. Además, diseñar nuevas aproximaciones para intensificar las soluciones obtenidas, mejorando así la calidad de estas. También resulta de especial interés considerar diferentes criterios de aceptación de soluciones, o quizá otras metaheurísticas. Por último, analizar la propuesta con el resto de conjuntos de instancias, estudiando la robustez de la propuesta frente a distintas topologías.

Agradecimientos Este trabajo ha sido posible gracias al apoyo del Ministerio de Economía y Competitividad (subvención con referencia PID2024-156045NB-I00), Ministerio para la Transformación Digital y de la Función Pública (Cátedra ENIA AI4DDS, ref. TSI-100930-2023-3) y la Comunidad Autónoma de Madrid (subvención con referencia TEC-2024/COM-404).

Declaración de intereses Los autores no tienen ningún conflicto de intereses que declarar que sea relevante para el contenido de este artículo.

Referencias

1. Boria, N., Murat, C., Paschos, V.T.: The probabilistic minimum dominating set problem. Discrete Applied Mathematics **234**, 93–113 (2018)
2. Bouamama, S., Blum, C.: An improved greedy heuristic for the minimum positive influence dominating set problem in social networks. Algorithms **14**(3), 79 (2021)
3. Casado, A., Bermudo, S., López-Sánchez, A.D., Sánchez-Oro, J.: An iterated greedy algorithm for finding the minimum dominating set in graphs. Mathematics and Computers in Simulation **207**, 41–58 (2023)
4. Casado, A., Mladenović, N., Sánchez-Oro, J., Duarte, A.: Variable neighborhood search approach with intensified shake for monitor placement. Networks **81**(3), 319–333 (2023)
5. Chinnasamy, A., Sivakumar, B., Selvakumari, P., Suresh, A.: Minimum connected dominating set based rsu allocation for smartcloud vehicles in vanet. Cluster Computing **22**(Suppl 5), 12795–12804 (2019)
6. Corcoran, P., Gagarin, A.: Heuristics for k-domination models of facility location problems in street networks. Computers & Operations Research **133**, 105368 (2021)
7. Gagarin, A., Corcoran, P.: Multiple domination models for placement of electric vehicle charging stations in road networks. Computers & Operations Research **96**, 69–79 (2018)
8. Hansen, P., Mladenović, N.: First vs. best improvement: An empirical study. Discrete Applied Mathematics **154**(5), 802 – 817 (Apr 2006), iV ALIO/EURO Workshop on Applied Combinatorial Optimization
9. Li, R., Liu, H., Wu, X., Wu, J., Yin, M.: An efficient local search algorithm for the minimum k-dominating set problem. IEEE Access **6**, 62062–62075 (2018)
10. Li, R., Liu, S., Wang, F., Gao, J., Liu, H., Hu, S., Yin, M.: A restart local search algorithm with relaxed configuration checking strategy for the minimum k-dominating set problem. Knowledge-Based Systems **254**, 109619 (2022)

11. Lozano-Osorio, I., Sánchez-Oro, J., López-Sánchez, A., Duarte, A.: A variable neighborhood search for the median location problem with interconnected facilities. International Transactions in Operational Research **32**(1), 69–89 (2025)
12. López-Ibáñez, M., Dubois-Lacoste, J., Pérez Cáceres, L., Birattari, M., Stützle, T.: The irace package: Iterated racing for automatic algorithm configuration. Operations Research Perspectives **3**, 43–58 (2016)
13. Mladenović, N., Hansen, P.: Variable neighborhood search. Computers & Operations Research **24**(11), 1097–1100 (1997)
14. Sánchez-Oro, J., Pantrigo, J.J., Duarte, A.: Combining intensification and diversification strategies in vns. an application to the vertex separation problem. Computers & Operations Research **52**, 209–219 (2014)