

UNIVERSIDAD AUTÓNOMA DEL ESTADO DE MORELOS

INSTITUTO DE INVESTIGACIÓN EN CIENCIAS BÁSICAS Y APLICADAS

CENTRO DE INVESTIGACIÓN EN CIENCIAS

**PROPIEDADES MATEMÁTICO-COMPUTACIONALES
ASOCIADAS A PROBLEMAS DE TEORÍA DE DOMINACIÓN**

TESIS PROFESIONAL PARA OBTENER EL GRADO DE:

DOCTOR EN CIENCIAS

PRESENTA:

M.C.M. Ernesto Parra Inza

DIRECTOR:

Dr. Nodari Vakhania Maisuradze

CO-DIRECTOR:

Dr. José María Sigarreta Almira

CUERNAVACA, MORELOS

FEBRERO, 2025



Lista del jurado

Presidente: Dra. Liudmila Lvovna Sabinina
Secretario: Dr. Antonio Daniel Rivera López
Vocal: Dr. José María Sigarreta Almira
Vocal: Dr. José Alberto Hernández Aguilar
Vocal: Dr. Nodari Vakhania Maisuradze
Suplente: Dr. Ulises Morales Fuentes
Suplente: Dr. Omar Rosario Cayetano

Lista de publicaciones relacionadas con esta tesis

1. Hernández Mira, F. A., Parra Inza, E., Sigarreta Almira, J. M., & Vakhania, N. (2021). Properties of the Global Total k-Domination Number. *Mathematics*, 9(5), 480. <https://doi.org/10.3390/math9050480>
2. Parra Inza, E., Hernández Mira, F.A., Sigarreta Almira, J.M., Vakhania, N. (2021). Enumerative Algorithm for finding minimum dominating set in a graph. In *Proceedings of the 1st Online Conference on Algorithms*, MDPI: Basel, Switzerland. Disponible en: <https://sciforum.net/paper/view/10907>
3. Hernández Mira, F. A., Parra Inza, E., Sigarreta Almira, J. M., & Vakhania, N. (2022). A polynomial-time approximation to a minimum dominating set in a graph. *Theoretical Computer Science*, 930, 142-156. <https://doi.org/10.1016/j.tcs.2022.07.020>
4. Parra Inza, E., & Sigarreta Almira, J. M. (2023). Dataset complementary prism networks. *Data in Brief*, 50, 109539. <https://doi.org/10.1016/j.dib.2023.109539>
5. Parra Inza, E., Vakhania, N., Sigarreta Almira, J. M., & Hernández Mira, F. A. (2024). Exact and heuristic algorithms for the domination problem. *European Journal of Operational Research*, 313(3), 926-936. <https://doi.org/10.1016/j.ejor.2023.08.033>
6. Parra Inza, E., Vakhania, N., Sigarreta Almira, J. M., & Hernández-Aguilar, J. A. (2024). Approximating a Minimum Dominating Set by Purification. *Algorithms*, 17(6), 258. <https://doi.org/10.3390/a17060258>
7. Parra Inza, E., Vakhania, N., Sigarreta Almira, J.M., Hernández Mira, F.A. (2025). Algorithms for the Global Domination Problem. *Computers and Operations Research*, 173, 106876. <https://doi.org/10.1016/j.cor.2024.106876>

Resumen

Esta tesis aborda el estudio de la Teoría de Dominación en Grafos, centrándose en las propiedades matemático-computacionales de diversos problemas de dominación y el desarrollo de algoritmos eficientes para su resolución. Los problemas tratados incluyen el problema del *conjunto dominante*, el *conjunto dominante global*, el *conjunto dominante global total* y el *conjunto k -dominante global total*. Un *conjunto dominante* de un grafo es un subconjunto de vértices tal que todo vértice no perteneciente a dicho conjunto es adyacente a al menos un vértice del conjunto. Un *conjunto dominante global* es un conjunto que domina tanto al grafo como a su complemento. Un *conjunto dominante global total* es un conjunto donde cada vértice del grafo es adyacente a al menos un vértice del conjunto, incluidos los vértices del conjunto mismo, tanto en el grafo como en su complemento. Finalmente, un *conjunto k -dominante global total* requiere que cada vértice del grafo sea adyacente a al menos k vértices del conjunto en el grafo y su complemento.

El problema científico que guía esta investigación se centra en cómo contribuir al desarrollo de la Teoría de Dominación en Grafos mediante el análisis de las propiedades de estos problemas y la creación de algoritmos que permitan resolverlos de forma óptima o subóptima.

La tesis se estructura en cinco capítulos. En el **Capítulo 1**, se sientan las bases teóricas de la investigación, abordando definiciones clave de la teoría de grafos, familias de grafos y la complejidad computacional asociada. Se presenta un marco conceptual que sustenta el análisis y desarrollo de los algoritmos que se proponen en los capítulos posteriores.

En el **Capítulo 2**, se analiza el *problema del conjunto dominante* clásico, un problema NP-hard que consiste en encontrar el conjunto mínimo de vértices que dominan a un grafo. Para

su resolución, se propuso un enfoque de dos etapas. En la primera etapa, se utiliza un algoritmo voraz que genera un conjunto dominante inicial, mientras que para la segunda etapa se implementaron cinco procedimientos de purificación que reducen el tamaño del conjunto obtenido y entregan un conjunto dominante minimal. Se implementaron también enfoques basados en programación lineal entera (ILP) y enumeración implícita, que lograron soluciones óptimas en instancias de tamaño mediano.

En el **Capítulo 3**, se estudia el *problema del conjunto dominante global*, donde el objetivo es encontrar un conjunto mínimo de vértices que domine tanto el grafo como a su complemento. Este problema es también NP-hard, y se demostró que mantiene su complejidad para ciertas familias de grafos, como los grafos planares y los grafos split. Para abordar este problema se desarrollaron tres heurísticas y un procedimiento de purificación. También se formuló y resolvió el problema mediante ILP y un algoritmo de enumeración implícita.

El **Capítulo 4** está dedicado al *problema del conjunto dominante global total*, una variante más compleja del problema anterior, en la que cada vértice del grafo debe estar dominado por al menos un vértice del conjunto dominante en el grafo y en su complemento. Se desarrollaron tanto métodos exactos como heurísticos. Además, se presentaron nuevas cotas inferiores y superiores para el número de dominación global total (cardinalidad del conjunto dominante global total mínimo) y se probó la efectividad de los algoritmos en una amplia variedad de instancias.

El **Capítulo 5** extiende el problema del capítulo anterior al *problema del conjunto k -dominante global total*, donde cada vértice del grafo debe estar dominado por al menos k vértices del conjunto dominante tanto en el grafo como en su complemento. Se establecieron relaciones entre los número de dominación global total para diferentes valores de k , lo que permitió derivar nuevas cotas y desarrollar un método para generar un conjunto k -dominante global total a partir de un conjunto $(k - 1)$ -dominante global total.

En general, esta tesis presenta avances significativos en la resolución de problemas de dominación en grafos mediante el desarrollo de algoritmos eficientes y técnicas de optimización.

Contribuye al desarrollo de la teoría y a su aplicación en instancias reales de gran tamaño, proporcionando soluciones tanto exactas como heurísticas, y abordando problemas NP-hard de manera eficiente y con alta precisión. Los resultados experimentales obtenidos son los mejores de su tipo existentes en la literatura para estos problemas. Los procedimientos de purificación mejoraron significativamente la calidad de las soluciones entregadas por las heurísticas, logrando una reducción importante en el tamaño de las soluciones. Las heurísticas propuestas, luego de la purificación, lograron encontrar el óptimo en aproximadamente la mitad de las instancias analizadas, y el error promedio fue de aproximadamente un vértice.

Abstract

This thesis addresses the study of Domination Theory in Graphs, focusing on the mathematical and computational properties of various domination problems and the development of efficient algorithms for their resolution. The problems discussed include the *dominating set* problem, the *global dominating set*, the *global total dominating set*, and the *global total k -dominating set*. A *dominating set* of a graph is a subset of vertices such that every vertex not belonging to this set is adjacent to at least one vertex in the set. A *global dominating set* is a set that dominates both the graph and its complement. A *global total dominating set* is a set in which each vertex in the graph is adjacent to at least one vertex in the set, including vertices within the set itself, both in the graph and its complement. Finally, a *global total k -dominating set* requires each vertex in the graph to be adjacent to at least k vertices in the set within both the graph and its complement.

The scientific problem guiding this research focuses on contributing to the development of Domination Theory in Graphs by analyzing the properties of these problems and creating algorithms that allow for their optimal or suboptimal solution.

The thesis is structured into five chapters. In **Chapter 1**, the theoretical foundations of the research are established, addressing key definitions in graph theory, families of graphs, and the associated computational complexity. A conceptual framework is presented to support the analysis and development of the algorithms proposed in the following chapters.

In **Chapter 2**, the classic *dominating set problem* is analyzed, an NP-hard problem that involves finding the minimum set of vertices that dominate a graph. A two-stage approach was proposed for its resolution. In the first stage, a greedy algorithm generates an initial domi-

nating set, while in the second stage, five purification procedures are implemented to reduce the size of the obtained set, yielding a minimal dominating set. Integer Linear Programming (ILP) and implicit enumeration approaches were also implemented, achieving optimal solutions in medium-sized instances.

In **Chapter 3**, the *global dominating set problem* is studied, where the objective is to find a minimum set of vertices that dominate both the graph and its complement. This problem is also NP-hard, and it has been shown to retain its complexity for certain graph families, such as planar graphs and split graphs. To address this problem, three heuristics and a purification procedure were developed. The problem was also formulated and solved using ILP and an implicit enumeration algorithm.

Chapter 4 is dedicated to the *global total dominating set problem*, a more complex variant of the previous problem, in which every vertex in the graph must be dominated by at least one vertex in the dominating set in both the graph and its complement. Both exact and heuristic methods were developed. In addition, new lower and upper bounds were presented for the global total domination number (cardinality of the minimum global total dominating set), and the effectiveness of the algorithms was tested on a wide variety of instances.

Chapter 5 extends the problem from the previous chapter to the *global total k -dominating set problem*, where each vertex in the graph must be dominated by at least k vertices from the dominating set in both the graph and its complement. Relationships between global total domination numbers for different values of k were established, allowing new bounds to be derived and a method to generate a global total k -dominating set from a global total $(k - 1)$ -dominating set to be developed.

In general, this thesis presents significant advances in solving domination problems in graphs through the development of efficient algorithms and optimization techniques. It contributes to the advancement of the theory and its application to large-scale real-world instances, providing both exact and heuristic solutions, and addressing NP-hard problems with efficiency and high accuracy. The experimental results obtained are the best of their kind in the litera-

ture for these problems. The purification procedures significantly improved the quality of the solutions provided by the heuristics, achieving a substantial reduction in solution size. The proposed heuristics, after purification, were able to find the optimal solution in approximately half of the analyzed instances, with an average error of approximately one vertex.

Dedicatoria

Dedico este trabajo a mi familia, cuyo apoyo incondicional ha sido fundamental tanto en mi trayectoria académica como en mi vida. También a todas las personas especiales que me acompañaron en esta etapa, contribuyendo a mi crecimiento profesional y personal.

A Dios, por ser mi guía en cada paso del camino, brindándome fortaleza y bendiciones para seguir adelante con determinación en la búsqueda de mis metas.

Agradecimientos

A mis padres, por todos los sacrificios que hicieron para darme la oportunidad de estudiar.

A Tania, por todo su apoyo incondicional.

A los Drs. Ricardo Abreu Blaya, José María Sigarreta Almira y Juan Carlos Hernández Gómez, porque sin ellos nada de esto sería posible.

A mis asesores, por creer en mi y permitirme trabajar a su lado.

A mis amigos y compañeros, por todos sus consejos y su apoyo.

Al CONAHCyT, por haberme brindado el apoyo económico necesario para realizar mis estudios de posgrado.

Índice general

Introducción	1
1. Preliminares	7
1.1. Definiciones en Teoría de Grafos	7
1.1.1. Familias de Grafos	10
1.2. Complejidad Computacional	13
2. Conjunto Dominante	16
2.1. Introducción	16
2.2. Propiedades básicas	20
2.3. Heurística de dos Etapa	21
2.3.1. Etapa 1: Algoritmo <i>Greedy</i>	21
2.3.2. Generalizaciones de clústeres y recorridos	27
2.3.3. Etapa 2: Procedimiento de purificación PP_0	30
2.3.4. Otros procedimientos de purificación PP_1 - PP_4	42
2.4. ILP formulation	48
2.5. Algoritmo de enumeración implícita	49
2.5.1. <i>Procedure_Next</i> (ν) y <i>Procedure_Priority_LIST</i> ()	50
2.5.2. Algoritmo BDS	52
2.6. Resultados experimentales	54
2.7. Conclusiones	63
3. Conjunto Dominante Global	65
3.1. Introducción	65
3.2. Propiedades básicas	67

3.3.	Resultados de NP-complejidad	68
3.3.1.	Grafo Split	69
3.3.2.	Grafo Planar	71
3.4.	Algoritmos exactos	73
3.4.1.	ILP formulation of the problem	73
3.4.2.	Algoritmo de enumeración implícita	73
3.5.	Algoritmos Heurísticos	75
3.5.1.	Heurística H1	75
3.5.2.	Heurística H2	79
3.5.3.	Heurística H3	83
3.5.4.	Procedimiento de purificación	85
3.6.	Resultados experimentales	86
3.6.1.	Rendimiento de los algoritmos heurísticos	87
3.6.2.	Rendimiento de IE vs. CPLEX e intlinprog	94
3.7.	Conclusiones	97
4.	Conjunto Dominante Global Total	99
4.1.	Introducción	99
4.2.	Cotas y resultados exactos para $\gamma_t^g(G)$	101
4.2.1.	Formulación ILP	110
4.3.	Algoritmos Heurísticos	112
4.3.1.	Heurística GTAD	112
4.3.2.	Heurística GTD	115
4.3.3.	Heurística GTD-AD	119
4.3.4.	Procedimiento de purificación	121
4.4.	Resultados experimentales	122
4.5.	Conclusiones	129
5.	Conjunto k-Dominante Global Total	130
5.1.	Introducción	130
5.2.	Relaciones entre $\gamma_{kt}^g(G)$ y $\gamma_{kt}(G)$	131
5.3.	Cotas superiores para $\gamma_{kt}^g(G)$	133

5.4. Cotas inferiores para $\gamma_{kt}^g(G)$	136
5.5. Derivación de cotas superiores para $\gamma_{(k+1)t}^g(G)$ a partir de $\gamma_{kt}^g(G)$	142
5.6. Conclusiones	146
Conclusiones Generales	148
Problemas abiertos	151
Referencias	152

Índice de figuras

1.1. Ejemplos de subgrafo y subgrafo inducido	9
1.2. Grafo ciclo C_5 y su complemento $\overline{C_5}$	10
1.3. Grafo camino P_8	10
1.4. Grafo ciclo C_{10}	11
1.5. Grafo completo K_{10}	11
1.6. Grafo bipartito completo $K_{4,6}$	12
1.7. Grafo estrella S_{10}	12
1.8. Grafo rueda W_{11}	12
2.1. Grafo S_{10} con $\gamma(S_{10}) = 1$	17
2.2. Grafo de la familia $\mathcal{F}_1$	24
2.3. Solución proporcionada por Greedy (vértices negros)	24
2.4. Grafo G , donde $S = \{1, 2, 3\}$ y el único $\gamma(G)$ -conjunto es $\{3, 5\}$	26
2.5. Bosques T^{h-1} y T^h (la arista roja en T^{h-1} es omitida en T^h)	29
2.6. Graph $C_5 \odot K_1$	36
2.7. Solución obtenida por el algoritmo en [89] y el Algoritmo 4	38
2.8. Solución del Alg. 4 para $\mathcal{T}(P_2, P_4)$ y $\mathcal{T}'(P_2, N_4)$	38
2.9. Grafo G con $\gamma(G) = 2$	40
2.10. Resultados de los procedimientos de purificación ($\min\{S_i^*\}$), $\gamma(G)$, y U	58
2.11. Tiempo de los procedimientos de purificación	59
2.12. Cota inferior y superior del intervalo de búsqueda para el algoritmo BDS en tres iteraciones consecutivas, y $\gamma(G)$	60
2.13. time	61
2.14. BDS vs. ILP para grafos densos	62

3.1. Grafo split con vértices de tipo x y u	71
3.2. Grafo planar H (grafo estrella) y G	71
3.3. Grafo $G \circ_r H$	72
3.4. Grafo G	78
3.5. Grafo con $\gamma_g(G) = 3$	79
3.6. (a) Grafo Petersen; (b) grafo G	81
3.7. Grafo $G' = G \circ_v H$, donde H es el grafo de Petersen.	82
3.8. Grafo $G' = P_3 \circ_v H$, donde H es el grafo de Petersen	82
3.9. Grafo $G' = G \circ_{v_i} H_i$	83
3.10. $\gamma_g(G)$ vs. mejor resultado de las heurísticas	87
3.11. Comportamiento asintótico de los radios de aproximación	88
3.12. Cota inferior y superior del intervalo de búsqueda para el algoritmo IE en cuatro iteraciones consecutivas, y $\gamma_g(G)$	94
3.13. IE vs. ILP para grafos con densidad $\approx 0,5$ (se aplicó escala logarítmica) . . .	96
3.14. IE vs. ILP para grafos densos (se aplicó escala logarítmica)	97
4.1. Grafo G con $diam(G) = diam(\overline{G}) = 2$	102
4.2. Grafo de Petersen y su complemento	103
4.3. Grafo G y G' tales que $diam(G) = 2$ y $g(G) = 4$	105
4.4. Grafo G con $\delta(G) = 2$, $diam(G) = 2$ y $\gamma_t^g(G) = 5$	106
4.5. Grafo G con $\delta(G) = 3$ y $\gamma_t^g(G) = 6$	107
4.6. Grafo G con $diam(G) = 4$ y $\gamma_t^g(G) = \gamma_t^g(G') + 1$	108
4.7. Grafo con $\gamma_t^g(G) = 4$ y $ S_h = 5$	118
4.8. Grafo $G' = G \circ_{q_i} Q_i$	118
4.9. Resultados de la aproximación vs. $\gamma_t^g(G)$	123
4.10. Radio de aproximación	125
4.11. Tiempo Formulación ILP vs. Heurística (se aplicó escala logarítmica)	126
4.12. Tiempos de ejecución de las heurísticas	129
5.1. Grafo G y $\overline{G}$, que satisfacen $\gamma_{2t}(G) = 5$, $\gamma_{2t}(\overline{G}) = 5$ y $\gamma_{2t}^g(G) = 6$	132
5.2. (a) G , (b) $\gamma_{2t}(G)$ -conjunto y (c) $\gamma_{2t}^g(G)$ -conjunto	136
5.3. Grafo G de orden $n = 10$ y $\delta = 5$	137

5.4.	Grafo G con $\gamma_{2t}^g(G) \geq \frac{\sqrt{8n+1}+1}{2}$.	140
5.5.	Grafo G con $\gamma'(A, B) = 2$ y $\gamma_{1t}^g(G) = 6$	143
5.6.	Grafo G con $\gamma_{1t}^g(G) = 4$, $\gamma_{2t}^g(G) = 6$ y $\gamma_{3t}^g(G) = 8$	146

Índice de tablas

2.1. Resultados de PP_0 - PP_4 para instancias de muestra	56
2.2. Promedio de purificación sobre todas las instancias	56
2.3. Comparación entre los resultados de PP_1 - PP_4 , $\gamma(G)$, y U	58
2.4. Resultados para grafos con densidad $\approx 0,2$	61
2.5. Resultados para grafos con densidad $\approx 0,5$	62
2.6. Resultados para grafos con densidad $\approx 0,8$	63
3.1. Resultados de las heurísticas H1 y H2	89
3.2. Análisis comparativo de las heurísticas H1 y H2 antes de la purificación . . .	90
3.3. Análisis comparativo de las heurísticas H1 y H2 después de la purificación .	90
3.4. Análisis comparativo de las dos versiones de la heurística H1 antes de la purificación	91
3.5. Análisis comparativo de las dos versiones de la heurística H1 después de la purificación	91
3.6. Resultados de las dos versiones de la heurística H3	92
3.7. Análisis comparativo de las dos versiones de la heurística H3 antes de la purificación	93
3.8. Análisis comparativo de las dos versiones de la heurística H3 después de la purificación	93
3.9. Análisis comparativo de las heurísticas H1, H2 y H3 tras la purificación . . .	93
3.10. Resultados para grafos con densidad inferior a 0,3	95
3.11. Resultados para grafos con densidad $\approx 0,5$	96
3.12. Resultados de grafos con densidad $\approx 0,8$	97

4.1.	Comparación de los valores de $\gamma_t(G)$ y $\gamma_t^g(G)$ con las soluciones heurísticas para 30 instancias analizadas con CPLEX	124
4.2.	Análisis comparativo de las heurísticas antes de la purificación	126
4.3.	Resultados de las heurísticas GTAD, GTD y GTD-AD	127
4.4.	Análisis comparativo de las heurísticas después de la purificación	128
4.5.	Instancias purificadas por heurística	128

Introducción

Las matemáticas representan tanto la clave como el acceso al conocimiento científico, desempeñando un papel esencial en el desarrollo cultural de cada sociedad. Su capacidad para modelar diversas problemáticas del mundo real contribuye al fortalecimiento del pensamiento abstracto, lógico, analítico y deductivo. No existe rama de las matemáticas, por compleja o teórica que parezca, que no encuentre aplicación en el estudio de los fenómenos del entorno. Su evolución ha sido un pilar fundamental en los avances científicos y tecnológicos actuales, dado que muchos de estos progresos dependen, de manera directa o indirecta, de esta disciplina.

Por su parte, los problemas de Optimización Discreta emergieron a mediados del siglo pasado por su amplia gama de aplicaciones en la industria. Algunos años después, han aparecido otras aplicaciones de relevancia, incluyendo los sistemas de cómputo, ruteo, calendarización y dominación. Los problemas de dominación en grafos constituyen una parte importante de los problemas de Optimización Discreta y la Teoría de Dominación en Grafos, con aplicaciones tanto teóricas como prácticas [4, 58, 59, 100].

El estudio de los grafos tiene sus inicios en 1736 con el trabajo de Leonhard Euler sobre el problema de los puentes de Königsberg, considerado el primer resultado formal en esta área. A partir de este análisis, surgió una de las ramas de las matemáticas con mayor aplicabilidad en la actualidad. Tanto en matemáticas como en ciencias de la computación, la teoría de grafos se encarga de analizar y comprender las propiedades estructurales de estas representaciones.

Un grafo, consiste en un conjunto de vértices, no vacío, y un conjunto de aristas definidas sobre el conjunto de vértices, este es un concepto esencial en Matemática Discreta [15]. Di-

cho concepto, es un instrumento poderoso para representar y modelar diferentes problemas teóricos-prácticos de optimización y estructuras de datos que guardan información de manera eficiente. Claros ejemplos de ello, son los problemas de ruteo, planificación o análisis de redes tróficas [8, 16, 19, 84, 93, 105, 109].

Los problemas teóricos en grafos poseen múltiples aplicaciones en la vida real; formando parte de ellos, los denominados problemas de dominación [13, 23, 24, 26, 80]. Por el interés de los investigadores en esta área, a mediados de 1900 empiezan a aparecer algunas publicaciones e investigaciones asociadas a la dominación en grafos, hasta crearse lo que hoy conocemos como Teoría de Dominación en Grafos [66]. Los problema de este tipo consisten en que dado un grafo $G = (V, E)$, encontrar un subconjunto de vértices $D \subseteq V$, de manera tal que cada vértice de $V \setminus D$ sea vecino a algún vértice en D . El número de dominación de G es la menor cardinalidad de cualquier conjunto dominante del grafo G y se denota por $\gamma(G)$. Para mayor profundización ver [3, 5, 13, 24, 28, 34, 48, 99].

Las soluciones constructivas para varios tipos de problemas de dominación en grafos tienen aplicaciones teórico-prácticas. Ha de destacarse que, la mayoría de los problemas de dominación en grafos son NP-hard [7, 18, 21, 30, 39, 60]; por lo tanto, encontrar algoritmos de aproximación para dichos problemas tendrá no solo un alcance teórico, si no también práctico.

La complejidad temporal de un problema de optimización discreto, puede ser polinomial o NP-hard. Al enfrentarnos a un problema polinomial se trata de mejorar la complejidad temporal del mejor algoritmo existente. En el caso de los problemas NP-hard es muy poco probable construir un algoritmo óptimo en tiempo polinomial. Sin embargo, se podría construir un algoritmo que corre en menos tiempo que otros algoritmos existentes para la mayoría de las instancias del problema y/o que garantiza una mejor aproximación al valor óptimo que los algoritmos existentes para el mismo problema [22, 31, 101, 104, 106–108, 110]. Cabe destacar que el estudio de conjuntos dominantes y la implementación de algoritmos eficientes para dar solución a este tipo de problemas es un tema novedoso y de actualidad [29, 77].

Problema científico

En función de lo anteriormente expuesto, se propone resolver el siguiente problema científico:
¿Cómo contribuir al desarrollo de la Teoría de Dominación en Grafos, mediante el estudio de sus propiedades matemático-computacionales y la elaboración de algoritmos eficientes para la solución de este tipo de problemas?

Objetivos

Objetivo General

Para resolver el problema científico antes enunciado, en esta investigación se propone como objetivo general:

Estudiar las propiedades matemático-computacionales asociadas a problemas de la teoría de dominación en grafos.

Objetivos Específicos

- Estudiar las propiedades matemático-computacionales de algunos problemas de dominación en grafos (dominante, dominante global, dominante global total, k -dominante global total).
- Desarrollar nuevos algoritmos eficientes para estos tipos de problemas considerando las versiones con conjuntos dominante, dominante global, dominante global total.
- Encontrar familias de grafos para los cuales los algoritmos construidos arrojen una solución óptima o sub-óptima e implementar y probar los algoritmos desarrollados para instancias de tamaño real.
- Encontrar nuevas cotas teóricas para el número de dominación global total y k -dominación global total.

Metodología

La metodología que seguirá esta investigación, estará sustentada en las siguientes etapas de trabajo:

- Aproximación problémica;
- Concreción;
- Representación-Valoración.

La etapa de Aproximación, consta de dos momentos básicos: identificación, formulación-reformulación y el hacer explícito los problemas de investigación a resolver, en tal sentido hemos llevado a cabo una primera aproximación (identificación) de la investigación. Así, nuevamente se realizará una revisión bibliográfica exhaustiva, para conocer los nuevos resultados y/o métodos utilizados en el tema, se formularán o reformularán los problemas a resolver y se analizarán las posibles técnicas empleadas por otros matemáticos, para enfrentar estas problemáticas.

En la etapa de Concreción, se pondrán en práctica los métodos y estrategias seleccionados, para la solución de los problemas delimitados en la primera etapa. Además, en esta etapa se elaborarán, si son necesarias, nuevas técnicas, procedimientos y estrategias para enfrentar los problemas de investigación. En esta etapa se desarrollarán algoritmos con base en las propiedades teóricas obtenidas en la etapa anterior.

En la etapa de Representación-Valoración, se implementarán y se realizarán pruebas, se codificarán los algoritmos desarrollados en la etapa anterior y se correrán para las instancias del problema correspondiente. Esto permitirá verificar la eficiencia práctica de los algoritmos construidos y establecer las clases de grafos para las cuales el algoritmo dado garantiza obtener la solución óptima. Además, se escribirán los resultados obtenidos ya sea en forma de teoremas, proposiciones, descripciones de los algoritmos construidos, las demostraciones de sus correctitudes y sus complejidades temporales, y las descripciones de las pruebas experimentales; para su posible publicación como artículos científicos de investigación o como elementos de divulgación de la ciencia. Así mismo, se valorarán los alcances teórico-prácticos

de los resultados de la investigación.

La tesis está estructurada en cinco capítulos. En el **Capítulo 1**, se sientan las bases teóricas de la investigación, abordando definiciones clave de la teoría de grafos, familias de grafos y la complejidad computacional asociada. Se presenta un marco conceptual que sustenta el análisis y desarrollo de los algoritmos que se proponen en los capítulos posteriores.

En el **Capítulo 2**, se analiza el *problema del conjunto dominante* clásico, un problema NP-hard que consiste en encontrar el conjunto mínimo de vértices que dominan a un grafo. Para su resolución, se propuso un enfoque de dos etapas. En la primera etapa, se utiliza un algoritmo voraz que genera un conjunto dominante inicial, mientras que en la segunda etapa se implementa un procedimiento de purificación que reduce el tamaño del conjunto obtenido y entrega un conjunto dominante minimal. Los resultados mostraron que la purificación mejoró significativamente la calidad de las soluciones del algoritmo inicial, logrando una reducción importante en el tamaño del conjunto. En pruebas realizadas con más de 1300 instancias, el algoritmo de dos etapas logró soluciones óptimas en el 46.33 % de las instancias, con un error promedio de solo 1.01 en las instancias restantes. Además, se desarrollaron cuatro nuevos procedimientos de purificación que mejoraron aún más las soluciones. Se implementaron también enfoques basados en programación lineal entera (ILP) y enumeración implícita, que lograron soluciones óptimas en instancias de tamaño mediano.

En el **Capítulo 3**, se estudia el *problema del conjunto dominante global*, donde el objetivo es encontrar un conjunto mínimo de vértices que domine tanto el grafo como a su complemento. Este problema es también NP-hard, y se demostró que mantiene su complejidad para ciertas familias de grafos, como los grafos planares y los grafos split. Para abordar este problema, se desarrollaron tres heurísticas que fueron capaces de generar soluciones óptimas en un 56.75 % de las instancias probadas, con un error absoluto promedio de aproximadamente un vértice en los casos donde no se obtuvo la solución óptima. También se formuló y resolvió el problema mediante programación lineal entera (ILP) y un algoritmo de enumeración implícita, logrando soluciones óptimas para instancias con hasta 8000 vértices.

El **Capítulo 4** está dedicado al *problema del conjunto dominante global total*, una variante más compleja del problema anterior, en la que cada vértice del grafo debe estar dominado por al menos un vértice del conjunto dominante en el grafo y en su complemento. Se desarrollaron tanto métodos exactos como heurísticos, logrando soluciones óptimas en el 41 % de las instancias probadas. En las instancias donde no se alcanzó la solución óptima, el error absoluto promedio fue de 1.58 vértices. Además, se presentaron nuevas cotas inferiores y superiores para el número de dominación global total (cardinalidad del conjunto dominante global total mínimo) y se probó la efectividad de los algoritmos en una amplia variedad de instancias.

El **Capítulo 5** extiende el problema del capítulo anterior al *problema del conjunto k -dominante global total*, donde cada vértice del grafo debe estar dominado por al menos k vértices del conjunto dominante tanto en el grafo como en su complemento. Se establecieron relaciones entre los número de dominación global total para diferentes valores de k , lo que permitió derivar nuevas cotas y desarrollar un método para generar un conjunto k -dominante global total a partir de un conjunto $(k - 1)$ -dominante global total.

Al finalizar, se exponen las conclusiones generales sobre la investigación realizada, y se enuncian algunas recomendaciones o problemas abiertos a tratar en trabajos futuros.

Preliminares

En el presente capítulo se tratan los aspectos teóricos que fundamentan esta investigación. El objetivo es sustentar teóricamente el estudio y antecedentes en general que se consideran válidos para el correcto desarrollo del trabajo. Para esto se analizan elementos tales como algunas definiciones de la teoría de grafos, familias de grafos y complejidad computacional.

1.1. Definiciones en Teoría de Grafos

A lo largo de los años, ante la necesidad de resolver determinados problemas de la teoría de grafos, se han definido términos y conceptos teóricos de los grafos. A continuación se exponen algunas definiciones y notaciones que se usarán en la presente investigación.

Definición 1.1. *Un grafo finito G es un par ordenado $G = (V, E)$ donde:*

- *$V(G)$ es un conjunto finito, no vacío, de elementos que se denominan vértices y*
- *$E(G) \subseteq \{\{u, v\} | u, v \in V(G)\}$ es el conjunto de aristas del grafo. Por simplicidad, se denotará a una arista $e \in E$ como $e = \{u, v\}$.*

Definición 1.2. *Dado un grafo finito $G = (V, E)$,*

- *Se llaman bucle o lazo a toda arista de la forma $\{v, v\}$.*
- *Dos vértices u, v son adyacentes o vecinos si $e = \{u, v\} \in E$.*

- Dada una arista $e = \{u, v\}$ se dice que e incide en los vértices u y v , si los extremos de e son u y v .
- Se dice que un grafo tiene aristas múltiples si existen dos o más aristas que conectan el mismo par de vértices.
- Un vértice es aislado si no es adyacente a ningún otro vértice del grafo.
- El grafo $G(V, E)$ es simple si no tiene bucles ni aristas múltiples.

A lo largo de esta investigación cuando se haga alusión a grafos, siempre se estará refiriendo a grafos simples. De esta manera, se omitirá en lo adelante este hecho.

Dado un grafo $G(V, E)$, se define el orden del grafo G como $n = |V(G)|$ y el tamaño como $m = |E(G)|$, considerando que $| \cdot |$ representa la cantidad de elementos o cardinalidad de un conjunto.

Definición 1.3. Sean $G(V, E)$ un grafo y $D \subseteq V(G)$ con $u \in V(G)$.

- $N_D(u)$ es el conjunto de vecinos de u en D .
- $\overline{N}_D(u)$ es el conjunto de no vecinos de u en D .
- $\delta_D(u)$ es el grado de u en D , $\delta_D(u) = |N_D(u)|$.
- $\delta(G)$ es el grado mínimo de G , $\delta(G) = \min\{\delta_V(u) \mid u \in V(G)\}$.
- $\Delta(G)$ es el grado máximo de G , $\Delta(G) = \max\{\delta_V(u) \mid u \in V(G)\}$.
- Un vértice hoja es un vértice de G con grado uno. Al conjunto de vértices hoja se denotará por $Leaf(G)$.
- Un vértice soporte es un vértice adyacente a una hoja. Al conjunto de vértices soporte se denotará por $Supp(G)$.

Definición 1.4. La vecindad privada de un vértice $v \in S \subseteq V(G)$ se define como $pn(v, S, G) = \{u \in V(G) : N(u) \cap S = \{v\}\}$; un vértice en la vecindad privada de v se denomina vecino privado con respecto al conjunto S .

Definición 1.5. Sea v un vértice del grafo $G(V, E)$. La excentricidad $\epsilon(v)$ de v es la mayor distancia entre v y los restantes vértices de G , $\epsilon(v) = \max_{u \in V(G) - v} \text{dis}(v, u)$. Donde $\text{dis}(u, v)$ denota el número de aristas en el camino más corto que conecta a v y u .

Definición 1.6. El diámetro de un grafo G es $d(G) = \max_{v \in V(G)} \epsilon(v)$, y el radio $r(G) = \min_{v \in V(G)} \epsilon(v)$.

Definición 1.7. El cuello de un grafo es la longitud del ciclo más corto contenido en el grafo, el cual se denota como $g(G)$.

Definición 1.8. Un grafo $H(V, E)$ es un subgrafo de $G(V, E)$ si $V(H) \subseteq V(G)$ y $E(H) \subseteq E(G)$. Si además cumple que $\forall u, v \in V(H)$, $\exists e = \{u, v\} \in E(G)$ y se tiene que $e \in E(H)$, se dice que H es un subgrafo inducido de G .

Veamos los siguientes ejemplos:

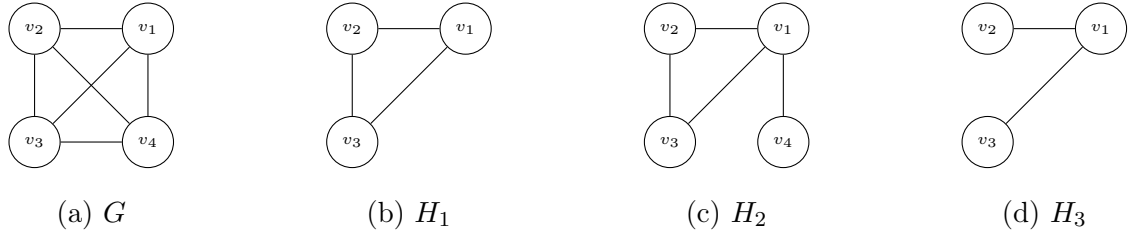
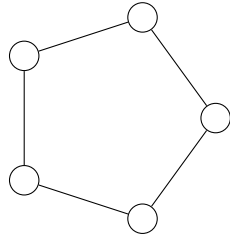


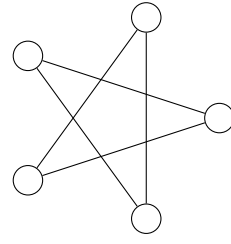
Figura 1.1: Ejemplos de subgrafo y subgrafo inducido

Como se muestra en la Figura 1.1, los grafos H_1 , H_2 , H_3 son subgrafos de G y H_1 es además el subgrafo de G inducido por los vértices $\{v_1, v_2, v_3\}$. En lo que sigue, se denotará el subgrafo de G inducido por el conjunto $V' \subseteq V(G)$, como $G[V']$.

Definición 1.9. Sea $G = (V, E)$ un grafo, su complemento $\overline{G} = (V, \overline{E})$ es el grafo con igual conjunto de vértices V , y su conjunto de aristas $\overline{E}$ está formado por todos los pares de vértices $\{u, v\}$ tales que $\{u, v\} \notin E$, es decir, $\overline{E} = \{\{u, v\} \mid u, v \in V \text{ y } \{u, v\} \notin E\}$.



(a) Ciclo C_5



(b) Complemento $\overline{C_5}$

Figura 1.2: Grafo ciclo C_5 y su complemento $\overline{C_5}$

Definición 1.10. Dos grafos $G = (V, E)$ y $G' = (V', E')$ son isomorfos si existe una biyección $f : V \rightarrow V'$ que preserva la relación de adyacencia, es decir $\{u, v\} \in E$ si y solo si $\{f(u), f(v)\} \in E'$.

Si dos grafos son isomorfos, deben contar con la misma cantidad de vértices. Además, cualquier propiedad derivada de la relación de adyacencia debe coincidir en ambos, incluyendo el número de aristas y los grados de los vértices.

1.1.1. Familias de Grafos

A continuación se definirán algunas familias de grafos que se utilizarán en los capítulos siguientes, estas definiciones fueron tomadas de [118].

Definición 1.11. Un **grafo camino** es una secuencia de vértices, tal que exista una arista entre cada vértice y el siguiente (ver Figura 1.3).

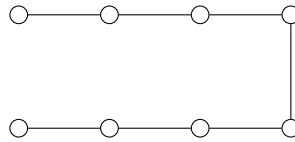


Figura 1.3: Grafo camino P_8 .

Los caminos se denotan por P_n , donde n representa el número de vértices del grafo. La Figura 1.3 muestra el grafo P_8 .

Definición 1.12. Un **grafo ciclo** es un camino cerrado en el que no se repite ningún vértice, y además el vértice inicial es igual al vértice final (ver Figura 1.4).

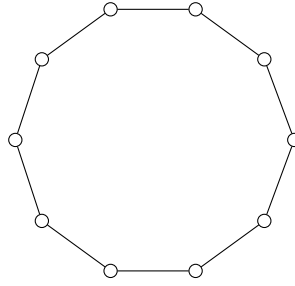


Figura 1.4: Grafo ciclo C_{10}

Un ciclo con n vértices se representa como C_n . En este tipo de grafo, la cantidad de vértices es igual a la cantidad de aristas, y cada vértice tiene grado 2, es decir, está conectado exactamente con otros dos vértices. La Figura 1.4 ilustra el grafo C_{10} .

Definición 1.13. Un **grafo completo** es un grafo simple donde cada par de vértices está conectado por una arista (ver Figura 1.5).

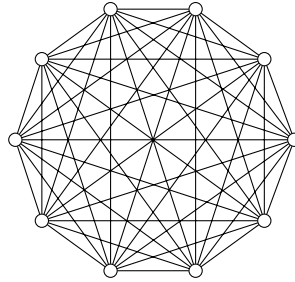


Figura 1.5: Grafo completo K_{10}

Un grafo completo de n vértices tiene $n(n-1)/2$ aristas, y se denota K_n . Es un grafo regular con todos sus vértices de grado $n-1$. La Figura 1.5 muestra el grafo K_{10} .

Definición 1.14. Un **grafo bipartito completo** $G(V_1 \cup V_2, E)$ es un grafo bipartito tal que $\forall u \in V_1, \forall v \in V_2 \{u, v\} \in E$. Es decir, un grafo bipartito completo está formado por dos conjuntos disjuntos de vértices y todas las posibles aristas que unen los vértices de un conjunto con los vertices del otro conjunto (ver Figura 1.6).

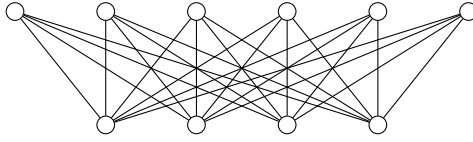


Figura 1.6: Grafo bipartito completo $K_{4,6}$

Un grafo bipartito completo con dos subconjuntos de cardinalidad $|V_1| = n$ y $|V_2| = m$, es denotado como $K_{n,m}$; su número de aristas es $n * m$. La Figura 1.6 muestra el grafo $K_{4,6}$.

Definición 1.15. Un *grafo estrella*, denotado por S_n , es el grafo bipartito completo $K_{1,n}$, es un árbol con un vértice soporte y n hojas (ver Figura 1.7).

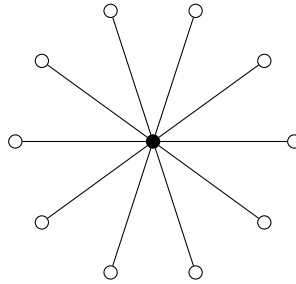


Figura 1.7: Grafo estrella S_{10}

La Figura 1.7 muestra el grafo S_{10} . Todo grafo S_n tiene n aristas y el vértice soporte tiene grado n .

Definición 1.16. El *grafo rueda*, denotado por W_n , es un grafo con n vértices que se forma conectando un único vértice a todos los vértices de un ciclo C_{n-1} (ver Figura 1.8).

La Figura 1.8 muestra el grafo W_{11} .

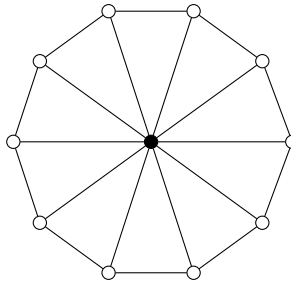


Figura 1.8: Grafo rueda W_{11}

1.2. Complejidad Computacional

La complejidad computacional es una rama de la teoría de la computación que estudia la dificultad teórica de resolver problemas, evaluando no solo la correctitud de un algoritmo, sino también los recursos necesarios para su ejecución. Los dos recursos más limitados en la producción de software o algoritmos son:

- El tiempo de ejecución (medido en cantidad de instrucciones elementales),
- La memoria necesaria para el almacenamiento de los datos (medida en términos de variables elementales).

Dado un algoritmo que resuelve un problema, es fundamental cuestionarse si es posible encontrar otro más eficiente para la misma tarea. El objetivo principal de la complejidad computacional es clasificar los problemas en función de su tratabilidad, evaluando la eficiencia de los algoritmos que los resuelven. Las preguntas clave que guían este análisis son: ¿Qué tan tratable es el problema? Y si lo es, ¿es el algoritmo lo suficientemente eficiente?

Si se demuestra que no existe un algoritmo mejor, se puede afirmar que el algoritmo disponible es eficiente. Un problema especialmente complejo se denomina "*NP-completo*", lo que implica que no se conoce un algoritmo eficiente para hallar una solución óptima. Demostrar que un problema es *NP-completo* es importante, ya que permite dejar de buscar soluciones óptimas inalcanzables y centrarse en encontrar soluciones aproximadas.

El desarrollo de la teoría de la complejidad ha permitido identificar problemas con características similares, que se agrupan en categorías para su estudio. Dos tipos principales de problemas son:

- *Problemas de optimización*: Se busca minimizar o maximizar una función objetivo dentro de un conjunto de soluciones factibles. Formalmente, se puede escribir como: *optimizar* $c(x)$ *sobre* x *en* $F(n)$, donde $c(x)$ es la función a optimizar y $F(n)$ es el conjunto de soluciones factibles.
- *Problemas de decisión*: Se trata de encontrar una respuesta de verdadero o falso.

Cualquier problema de optimización puede transformarse en un problema de decisión si se introduce un valor objetivo K y se pregunta si existe una solución factible que cumpla $c(x) \leq K$ o $c(x) \geq K$, según si se minimiza o maximiza, respectivamente. Para fines teóricos, es más conveniente trabajar con problemas de decisión.

En complejidad computacional, también se clasifican los problemas de acuerdo con:

- El estado del conocimiento sobre sus algoritmos (problemas abiertos o cerrados),
- Los recursos necesarios para resolverlos (problemas tratables o intratables).

Un problema se dice *cerrado* si se ha encontrado un algoritmo óptimo, y su complejidad ha sido demostrada con el orden de complejidad, denotado como $O()$. Este orden mide cómo crece el tiempo de ejecución de un algoritmo a medida que aumenta el tamaño de la entrada. Por ejemplo, un algoritmo con complejidad $O(n)$ indica que el tiempo de ejecución crece linealmente, mientras que $O(n^2)$ sugiere un crecimiento cuadrático.

Por otro lado, un problema es *tratable* si tiene solución para instancias grandes y las mejoras algorítmicas o de hardware amplían significativamente las instancias resolubles. En cambio, los problemas *intratables*, como las Torres de Hanói o el Problema del Viajante, solo pueden resolverse para instancias pequeñas en la práctica.

Para evaluar algoritmos de forma independiente del hardware, Alan Turing propuso un modelo conceptual, la Máquina de Turing Determinística, que permite medir los tiempos de ejecución contando el número de instrucciones básicas que ejecuta la máquina. La tesis de Turing-Church establece que si un problema puede resolverse en tiempo polinomial, es *tratable*; de lo contrario, es *intratable*.

En el análisis de complejidad se estudian principalmente problemas de decisión, ya que incluyen a los de optimización. Las clases más relevantes son:

- Problemas **P**: Aquellos que pueden resolverse en tiempo polinomial con una máquina determinista.

- Problemas **NP**: Problemas que pueden ser resueltos en tiempo polinomial en una máquina no determinista, pero no se conoce un algoritmo determinista que lo haga en tiempo polinomial.
- Problemas **NP-completos**: Problemas en NP para los cuales cualquier otro problema en NP puede reducirse en tiempo polinomial.

Para problemas de optimización, decimos que son NP, NP-completos o NP-hard si su problema de decisión asociado pertenece a esa categoría. Si se encontrara un algoritmo polinomial para un problema NP-completo, todos los problemas en NP podrían resolverse en tiempo polinomial.

Definición 1.17. *Un problema X se reduce en tiempo polinomial a un problema Y si las instancias de X pueden resolverse utilizando:*

- *Un número polinomial de pasos computacionales, más*
- *Un número polinomial de llamadas al algoritmo que resuelve Y .*

La relación entre las clases P y NP es estrecha: $P \subseteq NP$. Cualquier problema que pueda resolverse en tiempo polinomial con una máquina determinista también puede resolverse con una no determinista.

La teoría de NP-completitud se basa en las transformaciones polinomiales, que permiten convertir un problema P_1 en otro P_2 utilizando un algoritmo determinista en tiempo polinomial. Un problema NP-completo se define como uno que pertenece a NP, y cualquier otro problema en NP puede reducirse a él en tiempo polinomial.

Proposición 1.1. *[96] Si P_1 se transforma a P_2 y P_2 es NP-completo, entonces P_1 también es NP-completo.*

Esta proposición es crucial, ya que permite demostrar que un problema es NP-completo si se puede reducir otro problema NP-completo a él. Algunos problemas, aunque tan difíciles como los NP-completos, no pertenecen a NP. A estos problemas se les llama NP-hard o NP-duros, y suelen asociarse a problemas de optimización cuya versión de decisión es NP-completa.

Conjunto Dominante

2.1. Introducción

Los problemas de cobertura y partición en grafos son de los más investigados dentro de la optimización combinatoria y la teoría de grafos. Un problema muy estudiado de este tipo es el problema del conjunto dominante. El problema del conjunto dominante mínimo busca un conjunto dominante con la mínima cardinalidad. Un subconjunto de vértices en un grafo es un conjunto dominante si cada vértice que no está en ese subconjunto tiene al menos un vecino en él.

Definición 2.1. Sea $G = (V, E)$ un grafo, un conjunto $S \subseteq V$ es un conjunto dominante de G si para todo $v \in V$, o bien $v \in S$, o existe un vértice u en S tal que la arista $\{u, v\} \in E$.

Definición 2.2. El número de dominación de un grafo G , denotado como $\gamma(G)$, es la cardinalidad mínima de un conjunto dominante de G .

Un conjunto dominante de cardinalidad $\gamma(G)$ se denota como un $\gamma(G)$ -conjunto. Un conjunto dominante es *minimal* si al remover cualquiera de sus elementos, el conjunto resultante deja de ser dominante.

En la Figura 2.1, el vértice representado en negro es un ejemplo de un conjunto dominante minimal de cardinalidad uno para el grafo S_{10} . Por otro lado, en la Figura 2.2 se muestra otro ejemplo de un conjunto dominante minimal, el cual tiene cardinalidad tres.

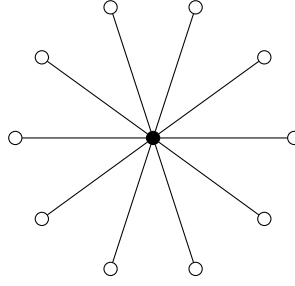


Figura 2.1: Grafo S_{10} con $\gamma(S_{10}) = 1$

El problema de dominación en grafos fue formalizado matemáticamente por Berge y Ore en [9] y [86], respectivamente. Actualmente, este tema ha sido detallado en los dos libros ampliamente conocidos de Haynes, Hedetniemi y Slater [60, 62]. La teoría de dominación en grafos es un área de creciente interés en matemática discreta y computación combinatoria, en particular, debido a una serie de aplicaciones importantes en la vida real. Dichas aplicaciones surgen, por ejemplo, en el análisis de redes sociales [17, 40, 73, 114], en la identificación eficiente de comunidades web [49], en bioinformática [61] y en sistemas biológicos [93, 112, 115], en problemas de ubicación de instalaciones en redes viales [60, 71], en la monitorización de sistemas eléctricos con el objetivo de minimizar las unidades de medición de estado [78], en redes eléctricas [63]. Diferentes versiones del problema de dominación en grafos se han utilizado para modelar la agrupación en redes inalámbricas [6, 113, 119, 120]. Las variaciones del problema de dominación y sus aplicaciones han sido ampliamente estudiadas en [60] y [62].

Existen algunos resultados conocidos que acotan el tamaño de un conjunto dominante mínimo, $\frac{n}{\Delta+1} \leq \gamma(G) \leq \frac{n}{2}$ o alternativamente, $\frac{d(G)+1}{3} \leq \gamma(G) \leq n - \Delta$ (ver [60]). A pesar de estas cotas, es NP-hard encontrar un conjunto dominante mínimo en un grafo, y sigue siendo así para grafos planares con $\Delta(G) = 3$ y para grafos planares regulares con $\Delta(G) = 4$ [53].

Se han propuesto algoritmos exactos (de tiempo exponencial) para este problema. Un algoritmo de ramificación y poda con un enfoque de medida y conquista para el problema del conjunto dominante fue propuesto en Fomin y otros [52]. El enfoque de medida y conquista ayuda a obtener buenas cotas superiores en el tiempo de ejecución al usar el método de ramificación y poda, mejorando frecuentemente las cotas conocidas hasta el momento. Este, ha sido usado exitosamente para el problema del conjunto dominante, ver Fomin y otros [52],

donde se obtuvo un tiempo de ejecución de $O(2^{0.610n})$. Van Rooij y Bodlaender en [111] han obtenido un algoritmo de tiempo $O(1.4969^n)$ utilizando un enfoque de medida y conquista, mejorando la cota obtenida en [52]. Algoritmos con tiempos de $O(1.4173^n)$, $O(1.4956^n)$ y $O(1.2303^{(1+\sqrt{1-2c})n})$ para encontrar un conjunto dominante mínimo para algunas clases de grafos (grafos cordales, grafos circulares y grafos densos, respectivamente) han sido propuestos en Gaspers y otros [54].

Como alternativa a los algoritmos exactos de tiempo exponencial, que no pueden proporcionar una solución para instancias prácticas de tamaño moderado, se han propuesto algoritmos voraces. Eubank y otros en [44], y Campan y otros en [25] presentan algoritmos heurísticos con un estudio experimental de su comportamiento práctico. Los autores no especifican el radio de aproximación ni la complejidad temporal de los algoritmos propuestos.

Chvátal en [32] describe un algoritmo de aproximación con un radio garantizado de $\ln\left(\frac{n}{OPT}\right) + 1 + \frac{1}{OPT}$ para el problema de la cobertura ponderada de conjuntos, donde OPT es el valor objetivo óptimo. En cada iteración, el algoritmo selecciona el vértice s que minimiza $\frac{w_s}{f(C \cup s) - f(C)}$, donde w_s es el peso del vértice s y $f(C) = |\cup_{s \in C} N[s]|$, hasta que $f(C) = |S|$. Este algoritmo puede usarse para resolver el problema del conjunto dominante. En particular, Parekh [89] ha aplicado este algoritmo voraz al problema de dominación, y mostró que la cardinalidad del conjunto dominante creado por ese algoritmo está acotada superiormente por $n+1-\sqrt{2m+1}$.

Existen algoritmos exactos de tiempo polinomial y de aproximación para algunas familias restringidas de grafos. En particular, se han descrito algoritmos de aproximación en tiempo polinomial para grafos de discos unitarios y grafos acotados por crecimiento (el problema sigue siendo NP-hard para tales grafos) en [68] y [85], respectivamente. Erlebach y Van Leeuwen [43] realizaron un estudio del problema para grafos de discos y propusieron un algoritmo polinomial para estos grafos. Alternativamente, un algoritmo polinomial de $2^{\log^* n}$ -aproximación para el conjunto dominante en grafos de discos fue propuesto por Gibson y Pirwani [55].

Existen resultados conocidos para grafos serie-paralelo, es decir, grafos con dos vértices dis-

tinguidos llamados terminales, formados recursivamente por dos operaciones de composición simples (ver [98]). Dichos grafos pueden usarse para modelar circuitos eléctricos en serie y paralelo. Para grafos serie-paralelo, Kikuno y otros [74] propusieron un algoritmo de tiempo lineal. Los llamados grafos k -degenerados fueron introducidos por Lick y White [79]. Dos algoritmos en tiempo lineal para encontrar un conjunto dominante de tamaño fijo en grafos degenerados se describen en Alon y Gutner [2].

Para árboles, Cockayne y otros [33] propusieron un algoritmo exacto con tiempo lineal. Siebertz [102] propuso un algoritmo voraz que aproxima conjuntos dominantes en grafos libres de biclques con una relación de aproximación de $O(t \ln(\gamma(G)))$ (grafos que no contienen $K_{t,t}$ como subgrafo inducido). Sin embargo, estos algoritmos no garantizan una buena aproximación para grafos generales.

También se han estudiado algunas generalizaciones para el problema del conjunto dominante mínimo. Por ejemplo, en el llamado problema del conjunto dominante ponderado mínimo (MWDS), a cada vértice se le asocia un número entero positivo, y el objetivo es encontrar un conjunto dominante con el peso más pequeño. Un algoritmo de búsqueda local para MWDS fue propuesto por Wang y otros en [116]. Ellos diseñaron un nuevo procedimiento de construcción rápida con cuatro reglas de reducción para reducir el tamaño de los grafos, y desarrollaron una estrategia de verificación de configuraciones para mejorar la búsqueda. Por otro lado, Guha y Khuller en [57], propusieron dos algoritmos heurísticos para aproximar un conjunto dominante conexo (un conjunto dominante en el cual el subgrafo inducido por los vértices de ese conjunto es conexo). Otra generalización del problema de dominación es el problema de k -dominación, que busca el llamado conjunto k -dominante, un conjunto de vértices tal que cada vértice del complemento de ese conjunto tiene al menos k vecinos en el conjunto anterior. Foerster en [50] propuso un algoritmo de $O(n(n + m))$ con un radio de aproximación de $\ln(\Delta + k) + 1$.

En este capítulo se propone un método de solución en dos etapas para el problema del CONJUNTO DOMINANTE MÍNIMO, que produce un algoritmo de aproximación en tiempo polinomial. En la primera etapa se crea un conjunto dominante, que no es necesariamente

óptimo (ni siquiera minimal), el cual se reduce en la segunda etapa de purificación. La Etapa 1 es un algoritmo voraz. En la Etapa 2, la reducción del conjunto dominante obtenido en la Etapa 1 se logra mediante el análisis del diagrama de flujo del algoritmo de la primera etapa, representado como un tipo especial de bosque. Este bosque se forma mediante un tipo especial de clúster del conjunto dominante creado en la primera etapa. Luego, la purificación de este conjunto dominante se lleva a cabo en la Etapa 2 utilizando un tipo especial de recorrido de cada árbol enraizado del bosque. El clúster puede realizarse de diferentes maneras. Diferentes métodos de agrupamiento o clusterización potencialmente producen una colección de distintos árboles enraizados. De igual manera, se pueden desarrollar diferentes métodos de recorrido y purificación para una estructura de clúster elegida. El mismo método de recorrido y purificación potencialmente dará un resultado diferente para una estructura de clúster dada. Por lo tanto, distintas combinaciones de clústeres y de recorrido/purificación dan lugar a diferentes algoritmos de purificación con distintas eficiencias. En este capítulo, proponemos métodos simples de clusterización y de recorrido/purificación, que dan lugar a un procedimiento con un tiempo de $O(n^3)$ o menor, mientras que la Etapa 1 se implementa en tiempo $O(n^3)$.

Además, se establecen algunos tipos de grafos para los cuales el algoritmo de la Etapa 1 ya entrega un conjunto dominante mínimo, y se derivan condiciones suficientes bajo las cuales el algoritmo completo entrega un conjunto dominante mínimo. También se propone un algoritmo exacto de enumeración implícita y un programa lineal entero (ILP, por sus siglas en inglés), y se evalúa su rendimiento en comparación con los algoritmos exactos previamente existentes en [111] y [70], los cuales también se implementaron en C++.

2.2. Propiedades básicas

En esta sección se enuncian algunas propiedades básicas sobre conjuntos dominantes. En [60] demuestran los tres teoremas siguientes:

Teorema 2.1. *Si $G(V, E)$ es un grafo conexo de orden n , entonces $\gamma(G) \geq \frac{n}{\Delta(G)+1}$.*

Teorema 2.2. *Si $G(V, E)$ es un grafo conexo, entonces $\gamma(G) \geq \frac{2r(G)}{3}$ y $\gamma(G) \geq \frac{d(G)+1}{3}$.*

Teorema 2.3. Si $G(V, E)$ es un grafo conexo de orden n , $s = |Supp(G)|$ y $l = |Leaf(G)|$. Entonces $s \leq \gamma(G) \leq n - l$.

El siguiente corolario es una consecuencia inmediata de los Teoremas 2.1, 2.2 y 2.3.

Corolario 2.1. Si $G(V, E)$ es un grafo conexo, entonces,

$$L = \max \left\{ \frac{n}{\Delta(G) + 1}, \frac{2r(G)}{3}, \frac{d(G) + 1}{3}, s \right\}$$

es una cota inferior para $\gamma(G)$.

Del Teorema 2.3, se obtiene el siguiente corolario.

Corolario 2.2. En un grafo G , existe al menos un conjunto dominante S con cardinalidad $\gamma(G)$ tal que $Supp(G) \subseteq S$ y $Leaf(G) \not\subseteq S$.

2.3. Heurística de dos Etapa

2.3.1. Etapa 1: Algoritmo *Greedy*

En esta sección, se presenta una descripción del algoritmo Greedy de la Etapa 1 y la construcción de los clústeres. Durante la construcción del conjunto dominante inicial S los clúster son generados. Como se mencionó brevemente en la introducción de este capítulo, el clúster C_i es un árbol enraizado, un subgrafo del grafo G asociado con alguna componente conexa del grafo inducido por el conjunto S . Inicialmente, un conjunto de clústeres $C_1, \dots, C_l$ forma una partición del conjunto S . Se denotará la unión de estos árboles como T . *Greedy* funciona en un número de iteraciones que está acotado superiormente por n . En cada iteración $h \geq 1$, se añade un vértice seleccionado especialmente, denotado por v_h , al conjunto dominante S_{h-1} de la iteración anterior, es decir, $S_h := S_{h-1} \cup v_h$, donde inicialmente $S_0 = \emptyset$. S_{h-1} es el conjunto dominante parcial de la iteración $h - 1$.

Definición 2.3. En cada iteración h , el grado activo de un vértice $v \in \overline{S_{h-1}}$ es $|N(v) \setminus [S_{h-1} \cup N(S_{h-1})]|$.

El vértice v_h es un vértice de $\overline{S_{h-1}}$ con el grado activo máximo. Cabe señalar que v_1 es un vértice con el grado máximo del grafo G . El algoritmo *Greedy* finaliza cuando S_h es un

conjunto dominante de G . En esa iteración, todos los vértices no dominados con grado activo 0 (si los hay) se incluyen en el conjunto S_h . A continuación, se presenta una descripción formal del algoritmo.

Algorithm 1 Algorithm Stage-One (*Greedy*)

Input: A graph G .
Output: A dominating set S of G .
 $h := 1$;
 $i := 1$;
 $S_0 := \emptyset$;
 $T^0 := \emptyset$;
{ iterative step }
while S_h is not a dominating set of graph G **do**
 $h := h + 1$;
 $v_h :=$ any vertex of $\overline{S_{h-1}}$ with maximum active degree;
 $S_h := S_{h-1} \cup \{v_h\}$;
 $[T^h, i] := \text{Cluster_Generation}(v_h, T^{h-1}, h, i)$;
end while

Implementación. Se representó el grafo $G = (V, E)$ mediante la matriz de adyacencia M , en la cual la entrada (i, j) es 1 si los vértices v_i y v_j son vecinos en el grafo G . La matriz de adyacencia se actualiza en cada iteración h según la selección del vértice v_h . Inicialmente, en la iteración 0, $M_0 := M$. De manera iterativa, algunas entradas 1 de la matriz M_{h-1} de la iteración $h - 1$ son reemplazadas por entradas 0 en la matriz M_h de la iteración $h \geq 1$. En particular, la matriz actualizada M_h se obtiene reemplazando una entrada 1 en la columna correspondiente a cada vecino del vértice v_h por un 0, y reemplazando todas las entradas 1 en la fila y la columna correspondientes al vértice v_h nuevamente por 0. De esta manera, todas las aristas incidentes a un vecino ya cubierto del vértice v_h y todas las aristas asociadas con el vértice v_h serán ignoradas a partir de la iteración h .

Lema 2.1. *En cada iteración h , el vértice v_h con el grado activo máximo se determina en tiempo $O(n^2)$.*

Demostración. Observe que, en cada iteración h , para cada vértice en $\overline{S_h}$, el grado activo se calcula sumando todas las entradas 1 en la fila correspondiente a ese vértice. Dado que cada fila contiene n entradas y hay menos de n filas que verificar, el cálculo de los grados activos de todos los elementos en $\overline{S_h}$ y la selección del máximo toma un tiempo de $O(n^2)$. ■

Teorema 2.4. *La Etapa 1 del algoritmo (*Greedy*) se ejecuta en un tiempo $O(n^3)$.*

Demostración. Recordemos que el número total de iteraciones (del bucle externo) en el algoritmo *Greedy* está acotado superiormente por n . En cada iteración h , la selección del vértice v_h toma un tiempo $O(n^2)$ (ver Lema 2.1). El costo de la actualización de la matriz de adyacencia en la iteración h también está acotado por $O(n^2)$. Por lo tanto, *Greedy* encuentra un conjunto dominante S en un tiempo $O(n^3)$. ■

La siguiente observación proporciona una condición suficiente de optimalidad para el algoritmo *Greedy*, y será útil en la estimación del radio de aproximación del algoritmo propuesto.

Observación 2.1. Si $|S| \leq 2$, entonces S es un conjunto dominante mínimo.

Demostración. Claramente, es suficiente considerar solo los dos casos, $\gamma(G) = 1$ y $\gamma(G) = 2$. De hecho, supongamos, primero, que $\gamma(G) = 1$. Entonces existe un vértice $v \in V$ adyacente a todos los demás vértices en V . En la iteración 1 del algoritmo *Greedy*, el grado activo del vértice v sería el grado máximo y este vértice será seleccionado como v_1 , y $S := \{v_1\}$ (los empates pueden ser claramente resueltos de forma arbitraria). De lo contrario, $\gamma(G) \geq 2$, mientras que el conjunto dominante devuelto por la Etapa 1 contiene como máximo 2 elementos, por lo que es óptimo. ■

Subclases de grafos favorables y desfavorables

Sea $\mathcal{F}_1$ la familia de grafos $W_r(S_{n_i}^i)$ con $i = \overline{1, r}$ y cualquier entero positivo r , donde $S_{n_i}^i$ es un grafo estrella con n_i vértices hoja, $|V(W_r(S_{n_i}^i))| = r + \sum_{i=1}^r n_i$, y los vértices hoja de $S_{n_i}^i$ tienen a lo sumo un vértice hoja en $S_{n_j}^j$ como vecino, con $i \neq j$. Se observa fácilmente que para cualquier grafo perteneciente a $\mathcal{F}_1$, $\gamma(\mathcal{F}_1) = r$ y el $\gamma(\mathcal{F}_1)$ -conjunto está compuesto por los r vértices soporte de las estrellas $S_{n_i}^i$. El algoritmo *Greedy* da un conjunto dominante mínimo. En la figura 2.2 se presenta un grafo representativo de la familia $\mathcal{F}_1$, donde los vértices negros forman la salida del algoritmo *Greedy*, el cual es óptimo.

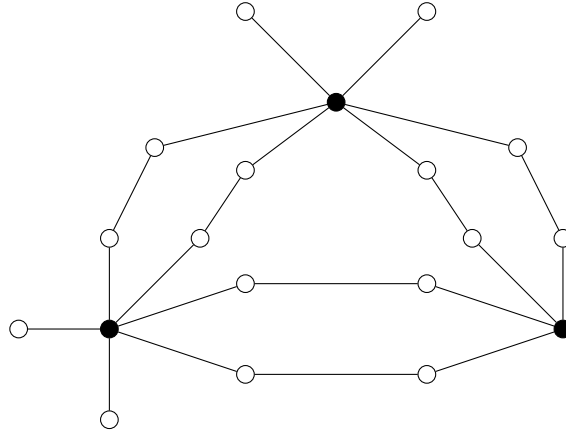


Figura 2.2: Grafo de la familia $\mathcal{F}_1$

La figura 2.3 muestra un ejemplo en el que el algoritmo *Greedy* de la Etapa 1 devuelve un conjunto dominante que no es minimal con una cardinalidad mayor que $n/2$.

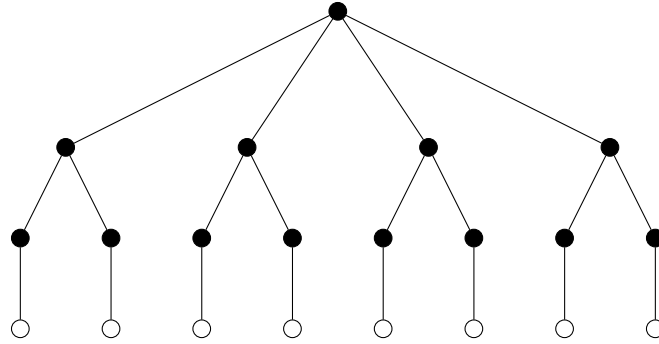


Figura 2.3: Solución proporcionada por Greedy (vértices negros)

Diagrama de flujo del algoritmo *Greedy* - clústeres y bosques

Es natural pensar en un procedimiento de purificación que elimine algunos vértices posiblemente redundantes del conjunto S , que no es necesariamente un conjunto dominante mínimo, ni siquiera un conjunto dominante minimal. En la Etapa 2 se reduce el conjunto dominante S de la Etapa 1 a un subconjunto $S^* \subseteq S$, que sigue siendo dominante, es minimal y contiene como máximo $n/2$ vértices.

Se dirá que el vértice $x \in S$ fue purificado si se elimina del conjunto S en la Etapa 2. Los vértices purificados se determinan en subgrafos especialmente formados del grafo G , los llamados clústeres. Cada clúster es un subgrafo del grafo G formado por un subconjunto de

vértices del conjunto S . Un clúster C es un árbol enraizado $T(C)$, y un conjunto de clústeres es formado mientras se ejecuta el algoritmo *Greedy* para una instancia particular del problema. Un árbol enraizado es un árbol (un grafo conexo y acíclico) en el que se ha designado un nodo especial llamado raíz. A partir de esta raíz, se establece una jerarquía en la que los demás nodos están organizados en niveles según su distancia a la raíz. La unión de todos los clústeres define un bosque T , un conjunto de árboles enraizados $T_1, \dots, T_k$, donde k es el número total de clústeres.

Los procedimientos de purificación, que se describen más adelante, se invocan para un clúster C_i y funcionan en varias iteraciones. Se denotará por T_i^h al árbol purificado correspondiente a la iteración h , y por T'_i al árbol purificado construido para el clúster C_i . T' denotará el bosque purificado completo, resultado de k llamadas independientes al procedimiento de purificación.

Ahora se introducen algunas definiciones necesarias que permitirán definir formalmente los clústeres. Sea S el resultado del algoritmo *Greedy* y sea $S = \{v_1, \dots, v_p\}$, para algún entero p , donde v_h es el vértice incluido en la iteración h . Sea $S(h)$ el conjunto de vértices no cubiertos por $v_1, \dots, v_{h-1}$ y cubiertos por v_h (nótese que los conjuntos $S(1), \dots, S(p)$ son disjuntos).

Diremos que un par ligado es un elemento del conjunto

$$P_S = \{\{v_h, v\} : v_h \in S, v \in S \cap S(h), h \in \{1, \dots, p\}\};$$

nótese que cada par ligado es una arista de G . La siguiente proposición se deduce inmediatamente de esta definición.

Proposición 2.1. *Sea $v_b, v_c \in S$, con $b < c$, tal que $\{v_b, v_c\} \in E$. Entonces, se tiene que $\{v_b, v_c\} \in P_S$ a menos que $\{v_a, v_c\} \in P_S$ para algún $v_a \in S$ con $a < b$.*

Sea $G(P_S)$ el subgrafo de G inducido por los pares (aristas de G) en P_S . Es decir, los vértices de $G(P_S)$ son aquellos involucrados en las aristas de P_S , y las aristas de $G(P_S)$ son las que pertenecen a P_S .

Nótese que $G(P_S)$ es un bosque de árboles enraizados, digamos $T_1, \dots, T_k$ para algún entero k . De hecho, no puede surgir un ciclo del tipo $\{v_a, v_b\}, \{v_b, v_c\}, \dots, \{v_h, v_a\}$ en $G(P_S)$, ya que,

según las definiciones anteriores, se tendría $a < b < c < \dots < h < a$, lo que es una contradicción. En particular, la raíz de T_i (para $i = 1, \dots, k$) es el vértice de T_i que fue agregado a S en la iteración más temprana del algoritmo *Greedy* sobre los vértices de T_i .

Esta forma de generar los clústeres intenta maximizar la profundidad/altura del árbol a construir. Este tipo de clúster solo se utilizará en el procedimiento de purificación PP_0 (Algoritmo 3). Más adelante se estudiarán otras variantes de clústeres y se justificará la creación de diferentes tipos de clústeres y procedimientos de purificación.

A primera vista, uno podría sugerir que *Greedy* entrega un conjunto dominante óptimo si no surge ningún par vinculado durante su ejecución, es decir, $P_S = \emptyset$. Aunque esto es cierto para la gran mayoría de los grafos, existen grafos para los cuales esto no se cumple. Como ejemplo, considere el grafo de la Figura 2.4. Se puede ver fácilmente que no surge ningún par vinculado y *Greedy* devuelve un conjunto de 3 vértices $\{1, 2, 3\}$. Pero $\gamma(G) = 2$. En general, si S no contiene ningún par vinculado, entonces S es un conjunto independiente. Sin embargo, la inexistencia de un par vinculado combinada con otra condición similar garantiza la optimalidad de *Greedy*, como se muestra a continuación.

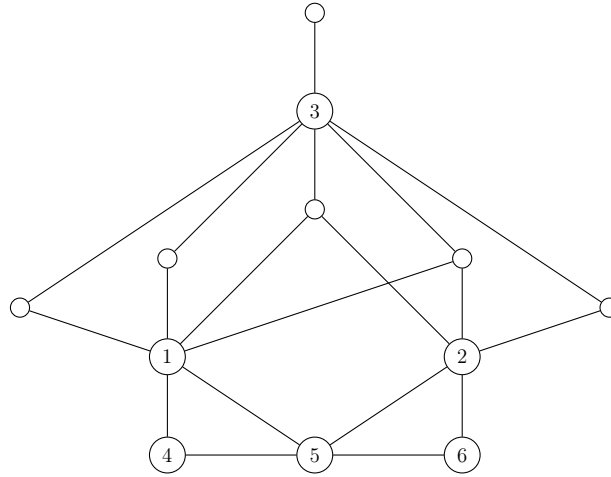


Figura 2.4: Grafo G , donde $S = \{1, 2, 3\}$ y el único $\gamma(G)$ -conjunto es $\{3, 5\}$

Recordemos que un *conjunto independiente* de un grafo es un conjunto de vértices que no son mutuamente no dyacentes. Las siguientes dos observaciones se deducen fácilmente de las definiciones.

Observación 2.2. *Se cumplen las siguientes afirmaciones:*

- (i) *S es un conjunto independiente de G si y solo si $P_S = \emptyset$;*
- (ii) *ningún vértice de $\bar{S}$ es adyacente a al menos dos vértices de S si y solo si existen dos vértices $v_a, v_b \in S$ y un vértice $w \in \bar{S}$ tal que $w \in S(a)$ y $\{w, v_b\} \notin E$.*

Observación 2.3. *$S \setminus V(G(P_S))$ es un conjunto independiente de G y cada vértice de $S \setminus V(G(P_S))$ tiene algún vecino en $V(G) \setminus S$.*

Proposición 2.2. *Si S es un conjunto independiente de G y ningún vértice de $\bar{S}$ es adyacente a al menos dos vértices de S , entonces S es un conjunto dominante mínimo de G .*

Demostración. La suposición implica que $\{\{v_1\} \cup N(v_1), \dots, \{v_p\} \cup N(v_p)\}$ es una partición de $V(G)$. Entonces, para cualquier conjunto dominante mínimo D de G , se tiene que $|D \cap (\{v_h\} \cup N(v_h))| \geq 1$ para cada $h \in \{1, \dots, p\}$. Se sigue que $|D| \geq |S|$, y por lo tanto, S debe ser un conjunto dominante mínimo de G . ■

2.3.2. Generalizaciones de clústeres y recorridos

Cada árbol T_i representa dependencias importantes en el conjunto dominante formado. Basándonos en estas dependencias, se puede purificar el conjunto dominante resultante de diferentes maneras. En particular, el orden en el que se purifican los vértices de cada clúster es importante y afecta el resultado del proceso de purificación. De hecho, se puede definir implícitamente una familia de algoritmos según el orden en que se consideran los vértices de cada clúster durante el proceso de purificación. Listados simples como de abajo hacia arriba y de arriba hacia abajo no necesariamente darán buenos resultados.

Una observación intuitiva y relevante es que un vértice en un clúster con un número considerable de descendientes inmediatos puede dejarse en el conjunto dominante mientras que todos sus descendientes inmediatos pueden ser purificados (a menos que alguno de esos descendientes tenga un vecino privado). Otra observación intuitiva es que, para un vértice dado de un clúster, importa el número de sus vecinos en el conjunto $\bar{S}$, los vértices con un número grande de estos vecinos pueden quedarse en el conjunto dominante. Estas observaciones intuitivas

pueden tenerse en cuenta de diferentes maneras en distintos métodos de recorrido/purificación. El estudio teórico y experimental de tales algoritmos de purificación se realiza en la siguiente sección de este capítulo.

Un clúster puede verse como un subgrafo dinámicamente formado del grafo G . Siempre que la arista $\{a, b\} \in E(G)$ esté en un clúster, tanto a como b deben haber sido incluidos en el conjunto dominante formado por *Greedy*. El inverso no es necesariamente cierto, es decir, no todas las dependencias (aristas) entre los vértices de un clúster están presentes en ese clúster (un clúster es un árbol).

Por ejemplo, para preservar una estructura de árbol, se podría simplemente incluir cada nuevo vértice como una nueva raíz de ese clúster, y existen muchas otras formas posibles de formar un clúster. Si ese vértice es vecino de dos o más vértices de diferentes clústeres, entonces todos esos clústeres se fusionarán basándose en esa dependencia (nuevamente, el nuevo vértice podría declararse como la raíz del nuevo clúster fusionado o la fusión podría realizarse de alguna otra manera). La estructura de clúster elegida afecta, en general, la eficiencia de un método de recorrido/purificación basado en esa estructura de clúster. Y, por supuesto, para una estructura de clúster fija, diferentes métodos de recorrido/purificación son posibles. Cada uno de ellos, en general, tendrá una eficiencia diferente.

Como se mencionó en la sección anterior, el clúster C_i es un árbol enraizado, un subgrafo del grafo G asociado con alguna componente conexo del grafo inducido por el conjunto S . Inicialmente, un conjunto de clústeres $\{C_1, \dots, C_l\}$ forma una partición del conjunto S . En un clúster, los vértices de cada componente conexa pueden representarse de diferentes maneras como árboles. Durante el recorrido de estos árboles, se verifican condiciones especiales y algunos vértices se omiten (purifican).

El objetivo es *purificar* este conjunto, es decir, omitir algunos vértices de ese conjunto de manera que el conjunto purificado siga siendo dominante. Una combinación de un método de clusterización particular y métodos de recorrido resulta en un procedimiento de purificación particular. A diferencia del procedimiento de clusterización propuesto en la sección anterior,

que está orientado a la creación de clústeres maximizando la profundidad/altura del árbol, aquí se propone una nueva metodología para crear clústeres que intenta maximizar la anchura del árbol. Este tipo de clúster se utilizará en los procedimientos de purificación PP_1 - PP_3 , que se describen en la siguiente sección.

Ahora se procede a explicar la nueva metodología de clusterización. Iterativamente, cada vértice siguiente v_h se incluye en el bosque parcial T^{h-1} de la iteración h , con $h = 1, \dots, k$, de la siguiente manera:

- Inicialmente, $T^0 := \emptyset$ y $T^1 := C_1 := v_1$.
- Para $h > 1$,
 - Si $N_{V(T^{h-1})}(v_h) = \emptyset$, entonces se crea un nuevo clúster $C_i := v_h$ y se actualiza T^h de la siguiente manera: $T^h := T^{h-1} \cup C_i$.
 - Si $N_{V(T^{h-1})}(v_h) = x$, donde $x \in T$ (es decir, $|N_{V(T^{h-1})}(v_h)| = 1$), entonces T^h es T^{h-1} complementado por el vértice v_h , que se convierte en un nuevo hijo del vértice x .
 - Si $|N_{V(T^{h-1})}(v_h)| > 1$, se crea un nuevo clúster C_i con raíz v_h y con los hijos en $N_{V(T^{h-1})}(v_h)$. En este caso, el clúster C_i incluye uno o más clústeres de T^{h-1} (estos clústeres desaparecen como clústeres individuales en el bosque T^h). Este tipo de unión puede generar ciclos (ver Figura 2.5). Para evitar esto, se elimina un número necesario de aristas de T^{h-1} incidentes con los vértices en $N_{V(T^{h-1})}(v_h)$.

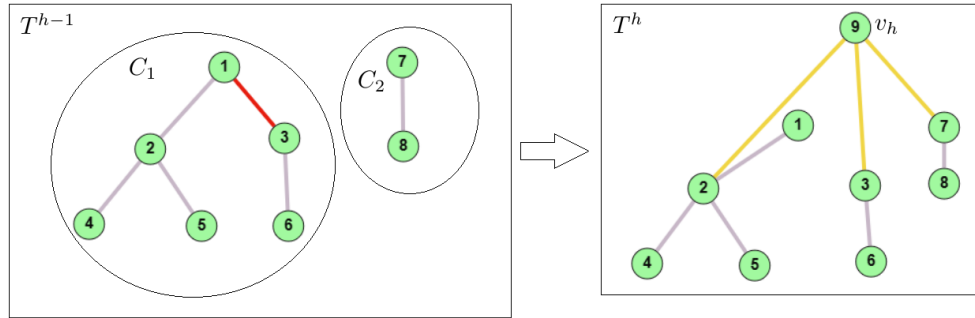


Figura 2.5: Bosques T^{h-1} y T^h (la arista roja en T^{h-1} es omitida en T^h)

A continuación, se presenta una descripción formal del procedimiento.

Algorithm 2 Cluster_Generation

Input: v_h , T^{h-1} , h , and number of clusters created i .
Output: Forest T^h and i .

```
if  $h = 1$  then
     $C_1 := \{v_1\}$ ;  $\{v_1$  is root of  $C_1\}$ 
     $T^1 := C_1$ ;
else
    if  $N_{V(T^{h-1})}(v_h) = \emptyset$  then
         $i := i + 1$ ;
         $C_i := \{v_h\}$ ;  $\{v_h$  is root of  $C_i\}$ 
         $T^h := T^{h-1} \cup C_i$ ;
    else
        if  $|N_{V(T^{h-1})}(v_h)| = 1$  then
             $x = N_{V(T^{h-1})}(v_h)$ 
             $T^h := T^{h-1} \cup \{x\}$  {insert  $v_h$  as a child of  $x$ };
        else
             $T^h := T^{h-1} \cup N_{V(T^{h-1})}(v_h)$  {insert  $N_{V(T^{h-1})}(v_h)$  as a child of  $v_h$  and update  $T^h$  };
        end if
    end if
end if
```

2.3.3. Etapa 2: Procedimiento de purificación PP_0

En esta sección se describe el procedimiento de purificación PP_0 . Recordemos que en las subsecciones 2.3.1 y 2.3.2 se argumentó que se podrían explotar y probar diferentes estructuras de clúster y métodos de recorrido; y que para el procedimiento de purificación PP_0 se utilizarán los clústeres creados maximizando la profundidad/altura, mientras que para los procedimientos de purificación PP_1 - PP_3 se utilizarán clústeres creados maximizando la anchura.

Procedimiento de purificación PP_0

Basados en la estructura de clúster particular que se define en la subsección 2.3.1, ahora se describe un procedimiento específico de recorrido y purificación (ver Algoritmo 3). Suponga que en la iteración h del procedimiento de purificación para el clúster C_i , $x \in V(G) \setminus \bar{S}$ y $v \in T_i^h$. Un vértice x es un *vecino semi-privado* del vértice v si v es el único vecino del vértice x en el bosque purificado actual $T'_1, \dots, T'_{i-1}, T^{h-1}_i, T_{i+1}, \dots, T_k$, subgrafo del bosque $T = G(P_S)$. Cabe señalar que un vecino semi-privado de un vértice v no es necesariamente un vecino privado de ese vértice en el grafo G .

En la iteración $h > 1$, se distinguen tres tipos de vértices en el árbol T_i^{h-1} : (1) los que ya han

sido purificados; (2) los que se establecen como no purificados en alguna iteración anterior, a estos vértices se les llama vértices *firμες*; y (3) los vértices restantes, aquellos para los cuales aún no se ha tomado ninguna decisión. Al último tipo de vértices se denominará *pendientes*; PF_i^h es el conjunto de vértices pendientes en la iteración h .

(0) *Inicialmente*, en la iteración 0, el procedimiento de purificación declara firmes todos los nodos hoja en el árbol T_i que tienen un vecino privado. Luego, en la iteración 0, procede a purificar las hojas no firmes en el árbol resultante. Si se purifica la hoja l , entonces su padre se establece como firme. El árbol resultante T_i^0 es el árbol purificado de la iteración inicial 0, y PF_i^0 es el conjunto resultante de los vértices que permanecen pendientes.

(1a) Antes de cualquier iteración $h \geq 1$, todos los vértices con vecinos semi-privados se establecen como firmes. Luego, el procedimiento aplica una de las dos reglas de purificación que se describen en los pasos (2) y (3).

(1b) En cada iteración $h \geq 1$, se busca el *vértice firme más lejos de la raíz y a la izquierda con un padre pendiente* en el árbol T_i^{h-1} . Se denota a este vértice firme como a^h . Si no existe un vértice a^h , equivalentemente, no existe ningún vértice pendiente en el árbol T_i^{h-1} , entonces el procedimiento de purificación retorna el árbol purificado $T'_i = T_i^{h-1}$ y se detiene.

(2) La *primera regla de purificación* purifica dos vértices pendientes b y c del camino (a, b, c, d) del árbol actual T_i^{h-1} en la iteración h , donde $a = a^h$. Se llamará a (a, b, c, d) como *cuádruple*. Note que en tal cuádruple, el vértice a ya es firme y los vértices b y c aún están pendientes en la iteración h ; ni el vértice b ni el vértice c pueden tener un vecino semi-privado por (1a). Por lo tanto, los vértices b y c son purificados y el vértice d se establece como firme a menos que haya sido previamente establecido como firme.

(3) La *segunda regla de purificación* se aplica si no existe un cuádruple, es decir, no hay dos vértices pendientes consecutivos en el árbol T_i^{h-1} . Luego, el procedimiento elige tres vértices consecutivos (a, b, c) en un camino en ese árbol, tales que, nuevamente, $a = a^h$ y b es un vértice aún pendiente en la iteración h . Se llamará a (a, b, c) como *trío*. Note que el vértice a es firme, el vértice b está pendiente y el vértice c podría haber sido

establecido como firme o purificado en una iteración anterior de otra rama. Esta regla purifica el vértice intermedio pendiente b .

- (4) Si en la iteración h , ni la primera ni la segunda regla de purificación pueden aplicarse, es decir, no existe ningún vértice pendiente en el árbol T_i^h , entonces el procedimiento retorna el árbol purificado $T'_i = T_i^h$ y se detiene.

Tenga en cuenta que en cada iteración h , los cuádruples y tríos se consideran como ramas ascendentes en el árbol T_i^h . Es decir, en un cuádruple (a, b, c, d) , $a = a^h$ es el vértice fijo de nivel más alto (el más alejado de la raíz), y d es el vértice de nivel más bajo (el más cercano a la raíz), en particular, a puede ser una hoja y d puede ser la raíz.

Algorithm 3 Procedure Purify PP_0 .

Input: Tree T_i and a graph G .

Output: The purified tree T'_i .

$T_i^0 := T_i$;

$PF_i^0 := V(T_i)$;

for any leaf $l \in T_i$ **do**

if l has no semi-private neighbor **then** { set l as purified and set its parent-node l^- firm }

$T_i^0 := T_i^0 \setminus \{l\}$;

$PF_i^0 := PF_i^0 \setminus \{l, l^-\}$;

else { set l firm }

$PF_i^0 := PF_i^0 \setminus \{l\}$;

end if

end for

 Iterative step:

$h := 0$;

while $PF_i^h \neq \emptyset$ **do**

$h := h + 1$;

for any vertex $v \in PF_i^h$ **do**

if x has a semi-private neighbor **then**

$PF_i^h := PF_i^{h-1} \setminus \{x\}$; { set x firm }

end if

end for

$a^h :=$ the highest level leftmost firm vertex with a pending parent in tree T_i^{h-1}

if there exists no vertex $a^h \in T_i^{h-1}$ **then**

 return $T'_i := T_i^{h-1}$ and stop;

end if { there exists vertex a^h }

if there exists quadruple (a^h, b, c, d) **then** {Note that in such quadruple, vertices b and c are in PF_i^{h-1} }

$T_i^h := T_i^{h-1} \setminus \{b\}$;

$PF_i^h := PF_i^{h-1} \setminus \{b\}$;

if c has a semi-private neighbor **then**

$PF_i^h := PF_i^h \setminus \{c\}$; { set c firm }

else

$T_i^h := T_i^h \setminus \{c\}$;

$PF_i^h := PF_i^h \setminus \{c, d\}$; { set d firm }

end if

else {there exists no quadruple}

if there exists trio (a^h, b, c) **then** {Note that in such trio, vertex $b \in PF_i^{h-1}$ }

$T_i^h := T_i^{h-1} \setminus \{b\}$;

$PF_i^h := PF_i^{h-1} \setminus \{b\}$;

end if

end if

 return $T'_i := T_i^h$ and stop;

end while

Como se muestra en el Lema 2.2, un v3rtice purificado en la iteraci3n h a partir del v3rtice a^h , no se har3 firme en otra cadena ascendente de una iteraci3n posterior. Por lo tanto, ning3n v3rtice purificado aparecer3 en el conjunto dominante devuelto por el procedimiento.

Lema 2.2. *Si PP_0 purifica el v3rtice $b \in T_i^{h-1}$ en la iteraci3n h , entonces ese v3rtice no puede ser firme en ninguna iteraci3n posterior.*

Demostraci3n. Sea el v3rtice b purificado desde el v3rtice a^h en la iteraci3n h . Para llegar a una contradicci3n, supongamos que el v3rtice b se hace firme desde el v3rtice a^g en la iteraci3n $g > h$. Por la construcci3n del procedimiento, deben existir dos v3rtices intermedios entre los

vértices a^g y b , ya que de lo contrario el vértice b no podría hacerse firme. Pero entonces el nivel del vértice a^g sería mayor que el del vértice a^h , y por lo tanto, el procedimiento no podría haber seleccionado el vértice a^h en la iteración h como el vértice firme de nivel más alto. ■

Lema 2.3. *Para cualquier clúster C_i y para cualquier vértice purificado $p \in T_i$, el algoritmo PP_0 deja un hijo o al padre del vértice p en el árbol purificado resultante T'_i . Además, al eliminar todos estos vértice del árbol T'_i , el vértice p quedará sin ser dominado. Por lo tanto, si el conjunto de vértices entregado por el procedimiento es un conjunto dominante, entonces también es un conjunto dominante minimal.*

Demostración. La primera afirmación se obtiene directamente de la construcción del algoritmo. Probemos la segunda. Todo vértice purificado del árbol T_i es el padre o el hijo de un vértice firme. Consideremos dos vértices purificados adyacentes, digamos b y c , de manera que b es el padre de un vértice firme $a = a^h$ y c es el hijo de otro vértice firme d ; aquí d es el más cercano a la raíz, y (a, b, c, d) es un camino (dirigido) en el árbol T_i en la correspondiente rama ascendente. Si el vértice a es una hoja, entonces tiene un vecino semi-privado, por lo que si el vértice a es purificado, el conjunto resultante no será dominante. Si el vértice a tiene un hijo que es una hoja, entonces esa hoja es purificada, por lo que si el vértice a también es purificado, el conjunto resultante nuevamente no será dominante. Si ninguno de los hijos del vértice a es una hoja, entonces al menos uno de ellos, digamos x , debería haber sido purificado, ya que de lo contrario el vértice a no podría haber sido establecido como un vértice firme. Además, un hijo del vértice x también debería haber sido purificado, ya que de lo contrario el vértice a también habría sido purificado.

Se tiene que el vértice a no puede ser purificado sin dejar algún vértice purificado sin dominar. Un razonamiento similar se puede aplicar al vértice d , lo que prueba la segunda afirmación. La tercera afirmación es una consecuencia directa de la anterior. ■

A continuación se proporciona una descripción formal del algoritmo general de dos etapas. Para los siguientes procedimientos de purificación, se sigue la misma idea de este algoritmo general de dos etapas.

Algorithm 4 The overall algorithm

Input: graph G

Output: a dominating set S^* of graph G , $S^* \subseteq S$.

Call Greedy to form dominating set S and construct the forest T consisting of the trees $T_1, \dots, T_k$;

$S^* := S \setminus V(G(P_S))$;

for $i = 1, \dots, k$ **do**

 Call *Procedure Purify*(T_i, G) to form purified trees T'_i ;

$S^* := S^* \cup V(T'_i)$;

end for

Cuestiones de implementación

Los vértices del bosque T adyacentes al vértice $x \in V(G)$ se representan como un árbol de búsqueda binaria balanceado $l^0(x)$ (los nodos de estos árboles están ordenados según sus índices). Cada vértice es una estructura de datos de registro con punteros. Cada vez que un vértice $v \in V(T_i^h)$ es purificado en la iteración h , se elimina el nodo que representa a dicho vértice en todos los árboles de búsqueda binaria que contienen un nodo que representa al vértice v . De esta manera, en cada iteración h , para cada vértice $x \in \bar{S}$, el número de nodos en el árbol $l^h(x)$ es igual al número de vecinos no purificados del vértice x . El número de nodos en estos árboles disminuye gradualmente a medida que se purifican nuevos vértices. Al mismo tiempo, por construcción, en ninguna iteración h el árbol $l^h(x)$ puede quedar vacío para ningún vértice x .

Es un hecho conocido que el número de vértices en un conjunto dominante mínimo está acotado superiormente por $\frac{n}{2}$, es decir, $\gamma(G) \leq \frac{n}{2}$ para todo grafo G de orden n , y esta cota es ajustada (ver el ejemplo en la Figura 2.6). En el Teorema 2.5, se demuestra que la misma cota es válida para el conjunto dominante obtenido por el algoritmo de dos etapas (*Greedy* + PP_0). Primero veamos el ejemplo y luego el Teorema 2.5.

El producto corona de los grafos G_1 y G_2 , $G_1 \odot G_2$, se define como el grafo obtenido a partir

de los grafos G_1 y G_2 tomando una copia de G_1 y n_1 copias de G_2 , uniendo por una arista cada vértice de la $i^{\text{ésima}}$ copia de G_2 con el $i^{\text{ésimo}}$ vértice de G_1 (n_1 es el orden de G_1). Existen grafos del producto corona de la forma $G = H \odot K_1$, para los cuales la cota superior $\frac{n}{2}$ es ajustada; aquí H es un grafo arbitrario y K_1 es un grafo completo de orden uno. Se puede ver fácilmente que el Algoritmo 4 alcanza esta cota superior para tales grafos.

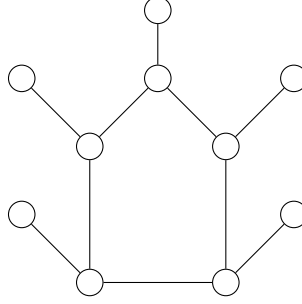


Figura 2.6: Graph $C_5 \odot K_1$

Teorema 2.5. *El procedimiento de purificación PP_0 entrega un conjunto dominante minimal S^* en tiempo $O(n^2 \log n)$ y $|S^*| \leq \frac{n}{2}$.*

Demostración. Primero, mostremos que el conjunto S^* obtenido por el procedimiento de purificación es dominante. En efecto, en cualquier iteración h , el árbol actualizado $l^h(x)$, para cada vértice $x \in \overline{S_{h\text{máx}}}$, contiene al menos un vértice, es decir, x está dominado por al menos un vértice del bosque purificado. Además, el conjunto S^* es minimal, ya que cada vértice firme tiene un vecino semi-privado.

En cuanto a la complejidad temporal, sea n_1 el número de vértices en $V \setminus V(T)$ y n_2 el número de vértices en el bosque T (donde $n_1 + n_2 \leq n$). El número de árboles $l^{h-1}(x)$ para $x \in V \setminus V(T_1 \cup \dots \cup T_k)$, es n_1 , y el número de vértices en cada árbol está acotado por n_2 . En una iteración h : (i) el vértice a^h puede determinarse en tiempo $O(n_2 \log n_2)$; (ii) se requiere tiempo $O(\log n_2)$ para localizar y eliminar el vértice que representa a un vértice purificado v en cada uno de los n_1 árboles binarios, mientras que hay como máximo dos vértices purificados. Por lo tanto, el costo total en la iteración h es $O(n \log n)$. El número total de iteraciones para purificar todos los árboles $T_1, \dots, T_k$ es, aproximadamente, $\frac{n_2}{3}$. Entonces, la complejidad temporal del procedimiento de purificación PP_0 es $O(n^2 \log n)$. ■

Como consecuencia de los resultados anteriores, se obtiene inmediatamente el siguiente corolario:

Corolario 2.3. *El algoritmo de dos etapas es correcto y se puede ejecutar en un tiempo $O(n^3)$.*

Estudio comparativo

La heurística descrita por Parekh en [89] extiende iterativamente el conjunto de vértices formado hasta el momento con un vértice que cubre el máximo número de vértices aún no cubiertos, hasta obtener un conjunto dominante. Este algoritmo es esencialmente el mismo que el algoritmo Greedy. Parekh no proporciona ni el radio de aproximación en el peor caso ni el análisis de la complejidad temporal. Sin embargo, ofrece la cota $n + 1 - \sqrt{2m + 1}$ sobre el número total de vértices en el conjunto dominante generado. Se puede verificar fácilmente que la prueba de esta cota en [89] se basa esencialmente en el hecho de que las aristas incidentes en los vértices pendientes de cada vértice seleccionado v_h se eliminan repetidamente en la heurística, lo que da lugar al término $-\sqrt{2m + 1}$. El mismo argumento se puede aplicar a la secuencia de nuestros grafos $G_0, G_1, \dots, G_{h_{\text{máx}}}$ en el algoritmo Greedy.

El algoritmo en [89] deja vértices redundantes en el conjunto dominante resultante, al igual que el algoritmo Greedy, que, en general, no es minimal. En los siguientes ejemplos se ilustra el rendimiento del algoritmo anterior y del algoritmo de dos etapas aquí propuesto. En la Figura 2.7A, se muestra un grafo con $n = 15$ vértices y $m = 24$ aristas. La solución generada por el algoritmo en [89] para este grafo consta de nueve vértices (los vértices grises y negros en la Figura 2.7A), mientras que el algoritmo de dos etapas obtiene un conjunto dominante óptimo con seis vértices (los vértices negros en la Figura 2.7). Para el ejemplo de la Figura 2.7B, el algoritmo en [89] obtiene un conjunto dominante con cardinalidad 13, mientras que el algoritmo de dos etapas nuevamente obtiene una solución óptima con 9 vértices (nótese que, usando estos ejemplos, se pueden construir fácilmente las correspondientes familias de grafos).

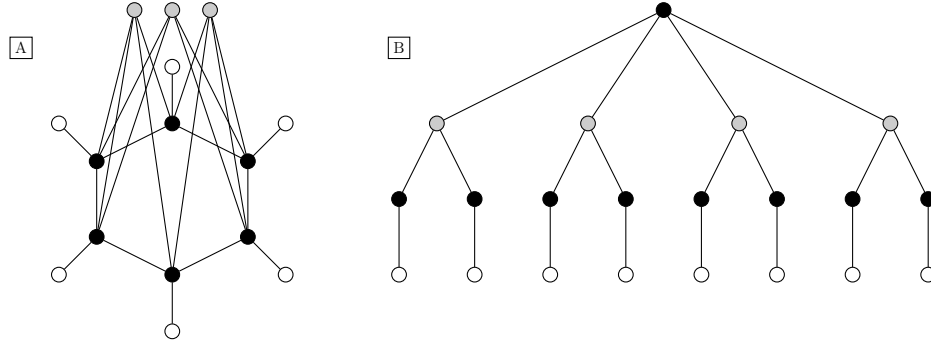


Figura 2.7: Solución obtenida por el algoritmo en [89] y el Algoritmo 4

Campan y otros [25] comparan el rendimiento práctico de cinco algoritmos de aproximación polinomial, tres de los cuales fueron propuestos anteriormente en Eubank y otros [44]. Basado en este estudio experimental, uno de los algoritmos, denominado como **Alg. 4**, resultó ser el que daba los mejores resultados. Ni el radio de aproximación ni la complejidad temporal de ninguno de los algoritmos considerados fue mostrado. No obstante, comparemos nuevamente el rendimiento de este último algoritmo y y el Algoritmo 4 en ejemplos simples.

Formemos una familia de grafos $\mathcal{T}(G, H)$ de la siguiente manera. Inicialmente, se aplica el producto corona $G \odot H$ a dos grafos arbitrarios G y H . Luego, para cada vértice $v \in G$, se agrega un grafo camino P_2 de vértices u y w , introduciendo la arista (v, u) . El grafo $\mathcal{T}(P_2, P_4)$ se muestra en la Figura 2.8A. Para un grafo de esta familia, el **Alg. 4** obtiene un conjunto dominante con cardinalidad $n(t + 1)$, donde n es el orden de G y t es la cardinalidad de la solución dada por el **Alg. 4** para el grafo H . El Algoritmo 4 obtiene un conjunto dominante óptimo de cardinalidad $2n \leq n(\gamma(H) + 1) \leq n(t + 1)$.

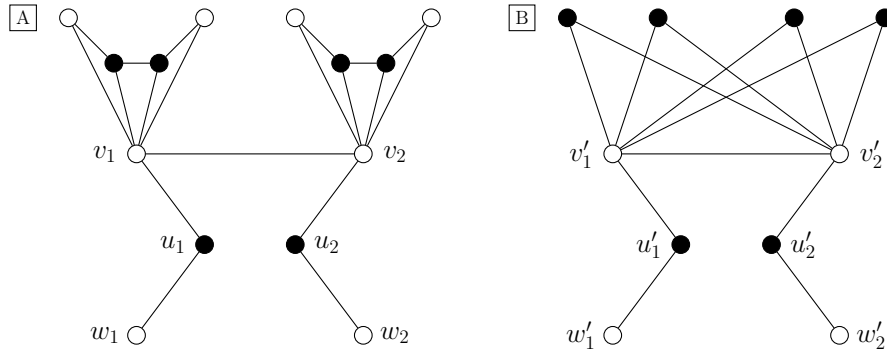


Figura 2.8: Solución del **Alg. 4** para $\mathcal{T}(P_2, P_4)$ y $\mathcal{T}'(P_2, N_4)$

Formemos otra familia de grafos $\mathcal{T}'(G, N_p)$ a partir del grafo G de orden n y el grafo sin aristas N_p de orden p , donde $p > n$. Se conecta cada vértice de G con todos los vértices de N_p mediante una arista, y para cada vértice $v' \in G$ se agrega un grafo camino P_2 de vértices u' y w' introduciendo la arista $\{v', u'\}$. Cabe destacar que el grafo resultante es de orden $3n + p$. El grafo $\mathcal{T}'(P_2, N_4)$ de la familia $\mathcal{T}'(G, N_p)$ se ilustra en la Figura 2.8B. Para un grafo de orden n de esta familia, el **Alg. 4** obtiene un conjunto dominante con una cardinalidad $p + n$, una magnitud mayor que $\frac{3n+p}{2}$. El Algoritmo 4 obtiene un conjunto dominante de cardinalidad $s + n \leq \frac{3n+p}{2} < p + n$, donde s es la cardinalidad de la solución dada por ese algoritmo para el grafo G . Recordemos que G es un grafo arbitrario utilizado para formar la familia $\mathcal{T}'(G, N_p)$.

Una generalización comúnmente estudiada del problema de dominación es el problema de k -dominación. El número de k -dominación de un grafo $G(V, E)$, $\gamma_k(G)$, es la cardinalidad mínima de un conjunto $S \subset V$ tal que cualquier vértice en $V \setminus S$ sea adyacente a al menos k vértices de S [27]. Cabe señalar que para $k = 1$, el problema de 1-dominación es el mismo que el problema de dominación. Foerster presenta en [50], un algoritmo heurístico para $k \geq 1$ y muestra que su radio de aproximación es $\ln(\Delta + k) + 1$, que coincide con el límite para el algoritmo *Greedy* (ver Ecuación 2.4), es decir, el algoritmo de [50] tiene el mismo radio de aproximación que el algoritmo *Greedy* para el problema de dominación clásico. En cuanto a las heurísticas mencionadas anteriormente de [89], el algoritmo anterior no necesariamente entrega un conjunto dominante minimal, y hay familias de grafos para las cuales el conjunto dominante formado por el algoritmo en [50] contiene más de $\frac{n}{2}$ vértices (ver ejemplo en la Figura 2.7B).

Recientemente, Wang y otros en [117] y [116] han propuesto heurísticas para la versión ponderada (MWDS) del problema de dominación. Utilizan repetidamente una función para elegir aleatoriamente el siguiente vértice que se añade al conjunto dominante actual. El algoritmo de [117] inicialmente reduce el tamaño del problema MWDS y luego, en la segunda etapa, construye la primera aproximación al problema MWDS haciendo elecciones aleatorias. En la tercera etapa, la salida de la segunda etapa se reduce eliminando algunos vértices redundantes aplicando una especie de procedimiento de purificación. Debido a la forma probabilística de la selección de cada siguiente vértice, las heurísticas no tienen una relación de aproximación

en el peor de los casos (determinística), pero según los resultados experimentales reportados, tienen un buen comportamiento práctico. Sin embargo, la etapa de purificación sigue dejando vértices redundantes (en particular, aquellos sin vecinos privados). Como un ejemplo ilustrativo simple, en la Figura 2.9 se representa un grafo para el cual el conjunto dominante generado por el algoritmo *Greedy* es 1, 2, 3. El procedimiento de purificación PP_0 elimina el vértice redundante 1, que quedará en el conjunto dominante construido por el algoritmo propuesto en [117].

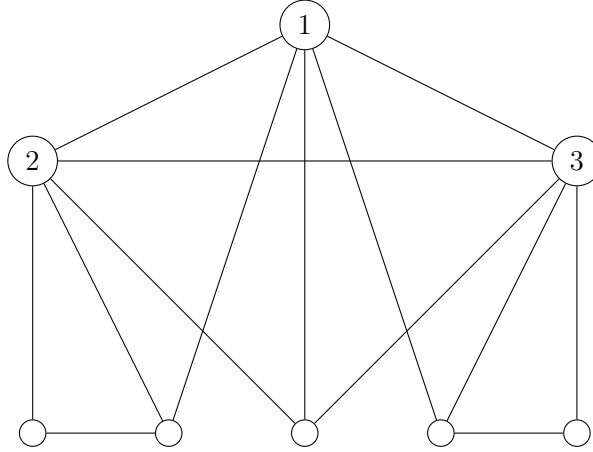


Figura 2.9: Grafo G con $\gamma(G) = 2$

Radios de aproximación

En esta subsección, se derivan cuatro radios de aproximación para el algoritmos de dos etapas. Según la Observación 2.1, todos los grafos G analizados aquí tendrán $\gamma(G) > 2$. Como ya se mencionó, el algoritmo en [50] obtiene un conjunto k -dominante con una relación de aproximación menor que

$$\frac{\ln\left(\frac{nk}{\gamma(G)}\right)}{\gamma(G)\ln\left(\frac{\gamma(G)}{\gamma(G)-1}\right)} + 1.$$

Ahora, al llevar a cabo un análisis similar al de [50], se puede ver que, para $k = 1$, el algoritmo *Greedy* logra un radio de aproximación similar

$$\rho \leq \frac{\ln\left(\frac{n}{\gamma(G)}\right)}{\gamma(G)\ln\left(\frac{\gamma(G)}{\gamma(G)-1}\right)} + 1. \quad (2.1)$$

El siguiente resultado es bien conocido y se demuestra en [62]:

$$\gamma(G) \geq \frac{n}{\Delta + 1}. \quad (2.2)$$

Dado que $\frac{1}{\gamma(G) \ln \left(\frac{\gamma(G)}{\gamma(G)-1} \right)} \leq 1$ y (2.1), se tiene que

$$\rho \leq \frac{\ln \left(\frac{n}{\gamma(G)} \right)}{\gamma(G) \ln \left(\frac{\gamma(G)}{\gamma(G)-1} \right)} + 1 \leq \ln \left(\frac{n}{\gamma(G)} \right) + 1. \quad (2.3)$$

Así, utilizando (2.2) y (2.3), obtenemos un radio de aproximación con invariantes n y Δ .

$$\rho \leq \ln(\Delta + 1) + 1. \quad (2.4)$$

A continuación, se deriva un radio de aproximación del Algoritmo 4 únicamente en términos del grado máximo de los vértices. Recordemos que S^* es un conjunto dominante minimal devuelto por el Algoritmo 4 y que, por el Teorema 2.5, se cumple que $\frac{n}{\Delta+1} \leq |S^*| \leq \frac{n}{2}$. Dado que $|S^*| \geq \gamma(G)$, se obtiene que un radio de aproximación para el Algoritmo 4 es

$$\rho = \frac{|S^*|}{\gamma(G)} \leq \frac{\frac{n}{2}}{\frac{n}{\Delta+1}} = \frac{\Delta + 1}{2}. \quad (2.5)$$

Usando los radios de aproximación (2.4) y (2.5), un radio de aproximación general para el Algoritmo 4 es:

$$\rho \leq \begin{cases} \frac{\Delta+1}{2}, & \text{if } 1 \leq \Delta \leq 4 \\ \ln(\Delta + 1) + 1, & \text{otherwise.} \end{cases}$$

Note que se necesita verificar que $\frac{\Delta+1}{2} \leq \ln(\Delta + 1) + 1$. Sea $f(\Delta) = e^{\frac{\Delta-1}{2}} - \Delta - 1$ una función con $\Delta \in \mathbb{R}$. Es fácil ver que la función f es 0 cuando $\Delta \approx 4,3567$. Dado que existe un cambio en el signo de la función f cuando $\Delta = 4,3567$, tenemos que $f(\Delta) > 0$ cuando $\Delta > 4$. De lo contrario, $f(\Delta) \leq 0$ cuando $\Delta \leq 4$.

Usando un análisis similar al descrito anteriormente, se obtiene un límite general más fuerte en términos del valor óptimo $\gamma(G)$.

$$\rho \leq \begin{cases} \frac{\Delta+1}{2}, & \text{if } n \geq \gamma(G)e^{\frac{\Delta-1}{2}} \\ \ln \left(\frac{n}{\gamma(G)} \right) + 1, & \text{otherwise.} \end{cases}$$

Note que

$$\ln\left(\frac{n}{\gamma(G)}\right) + 1 \leq \ln(\Delta + 1) + 1.$$

Primero, es necesario probar que $\frac{\Delta+1}{2} \leq \ln\left(\frac{n}{\gamma(G)}\right) + 1$ si y solo si $n \geq \gamma(G)e^{\frac{\Delta-1}{2}}$.

$$\frac{\Delta-1}{2} \leq \ln\left(\frac{n}{\gamma(G)}\right) \iff e^{\frac{\Delta-1}{2}} \leq \frac{n}{\gamma(G)} \iff \gamma(G)e^{\frac{\Delta-1}{2}} \leq n.$$

Consecuentemente, si $n \geq \gamma(G)e^{\frac{\Delta-1}{2}}$, entonces $\rho \leq \frac{\Delta+1}{2}$, y $\rho \leq \ln\left(\frac{n}{\gamma(G)}\right) + 1$ en caso contrario. Nuevamente, observe que los radios de aproximación derivados anteriormente tienen una evidente ventaja sobre el conocido anteriormente, ya que emplea únicamente parámetros invariantes del grafo y se pueden calcular fácilmente.

2.3.4. Otros procedimientos de purificación PP_1 - PP_4

Dado un conjunto de clústeres formados durante la ejecución del algoritmo en la Etapa 1, ahora se desea purificar el bosque T . Para el procedimiento de purificación PP_0 , en la Etapa 1, durante la construcción del conjunto dominante inicial $S = v_1, \dots, v_k$, se inserta el vértice v_h en el bosque actual T^{h-1} como un hijo de uno de los vértices insertados previamente $v \in T^{h-1}$ ($\{v, v_h\} \in E(G)$). El vértice $v \in C$ ($C \in T^{h-1}$) es el que está más alejado de la raíz en su clúster. En caso de que la raíz de otro clúster de T^{h-1} sea adyacente al vértice v_h (en el grafo G), esta raíz se convertirá en un hijo de v_h . Luego, ten en cuenta que los clústeres con tales raíces se fusionarán con el clúster anterior C en un nuevo clúster más grande.

Así, en el procedimiento PP_0 , los clústeres se generan en profundidad, resultando en clústeres con gran profundidad. Trabajar con clústeres profundos tiene algunas ventajas, pero también algunos inconvenientes. En el procedimiento de purificación PP_0 , en la Etapa 2, cada clúster es recorrido de abajo hacia arriba, desde una hoja hasta la raíz, y se identifican cuádruples (a, b, c, d) o tríos (a, b, c) que forman subpartes del camino correspondiente. Dado un cuádruple (a, b, c, d) , los vértices (b, c) son purificados a menos que tales vértices posean un *vecino semi-privado*. Para un trío, se verifica una condición similar solo para el vértice b . Como se describió anteriormente, dos vértices pueden ser purificados al mismo tiempo. Sin embargo, esto podría no ser posible, ya sea porque no exista un cuádruple (los caminos

ascendentes pueden ser demasiado cortos) o/y porque un vértice candidato podría poseer un vecino semi-privado.

Por otro lado, podría ser beneficioso fijar primero todos los vértices con un vecino semi-privado (un vértice fijo permanece sin purificar en el conjunto dominante resultante). También podría ser beneficioso dejar sin purificar un vértice que posea un número considerable de descendientes inmediatos sin vecinos semi-privados, en caso de que todos sus descendientes inmediatos sean purificados. De igual manera, podría ser útil dejar un vértice sin purificar si el número de sus vecinos en $\bar{S}$ es lo suficientemente grande. Las consideraciones anteriores y otras similares fueron tomadas en cuenta en cuatro nuevos procedimientos de purificación.

Los primeros tres procedimientos de purificación, PP_1 - PP_3 , hacen firmes todos los vértices del bosque T con vecinos privados en la iteración 0. Si el conjunto formado de esta manera es dominante, todos se detienen. De lo contrario, se distinguen tres subconjuntos disjuntos del conjunto S que forman una partición de S , tal como se define a continuación:

- (1) Conjunto de vértices firmes F^h (F^0 es el conjunto de vértices de T con vecinos privados; en la iteración $h > 0$, F^h es el conjunto de vértices de T con vecinos *semi-privados*).
- (2) Conjunto de vértices pendientes P^h , formado por vértices aún no firmes que son vecinos de un vértice en F^h .
- (3) Conjunto de los vértices restantes, U^h .

Iterativamente, una vez que F^h está determinado, se forman los conjuntos P^h y U^h . Los dos parámetros de purificación para cada vértice del clúster se definen a continuación.

El *conjunto de cobertura externa* de un vértice $v \in T^{h-1}$ en la iteración h , $OCS(v, h)$, es el conjunto de vecinos de v en $V \setminus S$, que no son adyacentes a ningún vértice firme del conjunto F^{h-1} ; es decir,

$$OCS(v, h) = N_{\overline{T^{h-1}}}(v) \setminus N_{\bar{S}}(F^{h-1}).$$

El *conjunto de cobertura interna* de un vértice $v \in T^{h-1}$ en la iteración h , $ICS(v, h)$, es el conjunto de vecinos aún no firmes del vértice v en T^{h-1} , es decir,

$$ICS(v, h) = N_{T^{h-1}}(v) \setminus F^{h-1}.$$

En términos generales, los vértices con altos valores de $|OCS(v, h)|$ y $|ICS(v, h)|$ tienden a no ser purificados; por el contrario, los descendientes directos de tales vértices con bajos índices de cobertura externa e interna tienden a ser purificados. Un balance global de purificación de un vértice v es

$$PB(v, h) = \alpha|OCS(v, h)| + \beta|ICS(v, h)|,$$

donde $0 \leq \alpha, \beta \leq 1$.

Las características distintivas de los procedimientos de purificación se definen a continuación.

Procedimiento de purificación PP_1

En la iteración $h > 0$, se recorre cada clúster $C \in T$ de manera ascendente por niveles. Para cada vértice no hoja x en un nivel dado del clúster C , si x es firme, sus hijos sin vecinos semi-privados son purificados. Si x no es firme y no tiene hijos firmes, se establece como firme y sus hijos sin vecinos semi-privados son purificados, mientras que los hijos con vecinos semi-privados se establecen como firmes. Si x tiene un hijo firme, el padre de x se establece como firme (a menos que ya lo sea) y x es purificado (ver Algoritmo 5).

Procedimiento de purificación PP_2

En la iteración $h > 0$, el vértice no firme x con el valor máximo de $PB(x, h)$ se establece como firme (para cada vértice no firme v , los conjuntos $OCS(v, h)$ e $ICS(v, h)$ se actualizan respectivamente). El procedimiento se detiene en la iteración h si $U^h = \emptyset$ y produce como salida el conjunto dominante F^h (ver Algoritmo 6).

Procedimiento de purificación PP_3

En la iteración $h > 0$, un vértice no firme x con el valor mínimo de $PB(x, h)$ es purificado, y los vértices de T^{h-1} con vecinos semi-privados de $\overline{T^{h-1}}$ se establecen como firmes (para cada vértice no firme v , los conjuntos $OCS(v, h)$ e $ICS(v, h)$ se actualizan respectivamente). El procedimiento se detiene en la iteración h si $U^h = \emptyset$ y produce como salida el conjunto dominante F^h (ver Algoritmo 7).

Algorithm 5 Procedure Purify 1 (PP_1).

Input: Forest $T = \{C_1, C_2, \dots, C_k\}$.

Output: S^* . { minimal dominating set }

 {set vertices with private neighbors}

$F^0 := \{\text{set of vertices from } T \text{ with private neighbors}\};$

if F^0 is dominating set **then**

$S^* := Firm;$

else

for any $C_i \in T$ **do** { in order }

$C_i^0 := C_i;$

$h := 0;$

 {Travel C_i by levels in ascending order}

for any $x \notin \text{leaf set in level } j \text{ of } C_i$ **do**

if $x \in F^h$ **then**

$C_i^{h+1} := C_i^h \setminus \{\text{children not a firm of } x \text{ do not have semi-private neighbors}\};$

else

if x has non firm children **then**

$F^{h+1} := F^h \cup \{x\};$

$C_i^{h+1} := C_i^h \setminus \{\text{children of } x \text{ do not have semi-private neighbors}\};$

else

if parent of x is not in F^h **then**

$F^{h+1} := F^h \cup \{\text{parent of } x\};$

$C_i^{h+1} := C_i^h \setminus \{x\};$

else

$C_i^{h+1} := C_i^h \setminus \{x\};$

end if

end if

end if

$h := h + 1;$

end for

end for

$S^* := F^h;$

end if

Algorithm 6 Procedure Purify 2 (PP_2).

Input: Forest $T = \{C_1, C_2, \dots, C_k\}$.
Output: S^* . { minimal dominating set }
 $F^0 := \{\text{set of vertices from } T \text{ with private neighbors}\};$
 $P^0 := \emptyset;$
if F^0 is dominating set **then**
 $S^* := F^0;$
else
 $h := 0;$
 for any $C_i \in T$ **do** { in order }
 while $V(C_i) \setminus (F^h \cup P^h) \neq \emptyset$ **do**
 $v := \text{any vertex with the maximum PB}(v, h) \text{ in set } V(C_i) \setminus F^h;$
 $F^{h+1} := F^h \cup \{v\};$
 $P^{h+1} := P^h \cup (N_{V(C_i)}(v) \setminus F^h);$
 $h := h + 1;$
 end while
 if F^h is dominating set **then**
 $S^* := F^h;$
 break;
 end if
 end for
end if

Algorithm 7 Procedure Purify 3 (PP_3).

Input: Forest $T = \{T(C_1), T(C_2), \dots, T(C_k)\}$.
Output: S^* . { minimal dominating set }
 $F^0 := \{\text{set of vertices from } T \text{ with private neighbors}\};$
 $P^0 := \emptyset;$
if F^0 is dominating set **then**
 $S^* := F^0;$
else
 $h := 0;$
 for any $T(C_i) \in T$ **do** { in order }
 while $T(C_i) \setminus (F^h \cup P^h) \neq \emptyset$ **do**
 $v := \text{any vertex with the minimum PB}(v, h) \text{ in set } T(C_i) \setminus F^h;$
 $P^{h+1} := P^h \cup \{v\};$
 $F^{h+1} := F^h \cup \{\text{all vetex in set } T \setminus (F^h \cup P^{h+1}) \text{ that have semi-private neighbor}\};$
 $h := h + 1;$
 end while
 if F^h is dominating set **then**
 $S^* := F^h;$
 break;
 end if
 end for
end if

Procedimiento de purificación PP_4

Una de las buenas intuiciones en el procedimiento de purificación PP_0 es la siguiente: podría ser conveniente que, para cualesquiera de los nodos adyacentes a, b de T_i tales que el nodo a sea una hoja de T_i , el nodo a sea procesado antes que el nodo b . Entonces, siguiendo esta intuición, parece que el núcleo es el siguiente: podría ser conveniente que, para cualesquiera vértices adyacentes a, b de S tales que a se haya añadido a S en la Etapa 1 después que b , el vértice a sea procesado antes que el vértice b . De hecho, el vértice a podría tener un *vecino privado* en $V \setminus S$ respecto al vértice b . Esto conduce a la siguiente alternativa de purificación.

Sea $S = \{v_1, \dots, v_p\}$, donde los vértices $\{v_1, \dots, v_p\}$ están ordenados en el orden en el que tales vértices fueron añadidos a S en la Etapa 1 y $S^* = S$; para $h = p, \dots, 1$, si $N[v_h] \subseteq (S^* \setminus \{v_h\}) \cup N(S^* \setminus \{v_h\})$, entonces se purifica al vértice v_h del conjunto S^* . Ver Algoritmo 8.

Algorithm 8 Procedure Purify 4 (PP_4)

Input: A graph $G(V, E)$ and a dominating set S .
Output: A dominating set S^* .
 $S^* := \{v_1, \dots, v_p\}$; {vertices of S in the order they were added by *Greedy*.}
 $h := p$;
 { iterative step }
while $h > 0$ **do**
 if $N[v_h] \subseteq (S^* \setminus \{v_h\}) \cup N(S^* \setminus \{v_h\})$ **then**
 $S^* := S^* \setminus \{v_h\}$; {we purify vertex v_h }
 end if
 $h := h - 1$;
end while

Radio de aproximación para PP_1 - PP_4

Por la Observación 2.1, si $|S| \leq 2$ entonces S es un conjunto dominante mínimo. Por lo tanto, todos los grafos G analizados aquí serán tales que $\gamma(G) > 2$. Nótese que G es un grafo conexo, y todo $x \in S^*$ tiene al menos un vecino semi-privado (S^* es el resultado de los procedimientos de purificación). Entonces $|S^*| \leq n/2$.

A continuación, se deriva un radio de aproximación para los procedimientos de purificación PP_1 - PP_4 . Según [60], $\gamma(G) \geq \frac{n}{\Delta+1}$. Entonces $\frac{n}{\Delta+1} \leq |S^*| \leq \frac{n}{2}$. Dado que $|S^*| \geq \gamma(G)$, se

obtiene que un radio de aproximación para la purificación es:

$$\rho = \frac{|S^*|}{\gamma(G)} \leq \frac{\frac{n}{2}}{\frac{n}{\Delta+1}} = \frac{\Delta+1}{2}. \quad (2.6)$$

En la subsección 2.3.3 se demuestra que el algoritmo Greedy logra un radio de aproximación $\rho \leq \ln(\Delta+1) + 1$. Usando estos radios de aproximación, se obtiene un radio de aproximación general para los procedimientos de purificación, que es esencialmente el mismo que el obtenido para el procedimiento PP_0 .

$$\rho \leq \begin{cases} \frac{\Delta+1}{2}, & \text{if } 1 \leq \Delta \leq 4 \\ \ln(\Delta+1) + 1, & \text{otherwise.} \end{cases} \quad (2.7)$$

2.4. ILP formulation

La formulación ILP del problema de dominación que se propone aquí es similar a las de [45] y [36] para el problema del conjunto dominante conexo y el problema de dominación independiente ponderada, respectivamente. Dado un grafo $G = (V, E)$ de orden n , la matriz de vecindades de G , $A = (a_{ij})_{n \times n}$, se define de la siguiente manera: $a_{ij} = a_{ji} = 1$ si $\{i, j\} \in E(G)$ o $i = j$, y $a_{ij} = a_{ji} = 0$ en caso contrario. Dado un conjunto dominante de cardinalidad $\gamma(G)$ e $i \in V(G)$, se definen las variables de decisión de la siguiente manera:

$$x_i = \begin{cases} 1, & \text{if } i \in S \\ 0, & \text{otherwise,} \end{cases} \quad (2.8)$$

El problema del conjunto dominante se puede formular de la siguiente manera:

$$\text{mín} \sum_{i=1}^n x_i = \gamma(G), \quad (2.9)$$

$$\text{subject to : } \sum_{j=1}^n a_{ij} x_j \geq 1, \quad x_j \in \{0, 1\}, \quad \forall i \in V(G), \quad (2.10)$$

Aquí, se desea minimizar la función objetivo 2.9. Se tienen n restricciones de tipo 2.10, garantizando que para cada $j \in V(G)$, al menos uno de los vértices en S debe ser adyacente a j o $j \in S$.

2.5. Algoritmo de enumeración implícita

El algoritmo de enumeración implícita encuentra un $\gamma(G)$ -conjunto enumerando subconjuntos de soluciones especialmente determinados. La cota superior inicial U se obtiene del tamaño de una solución factible a partir del algoritmo de aproximación de dos etapas descrito en las secciones anteriores. Dada la cota inferior L del Corolario 2.1 y la cota superior U , se restringe la búsqueda de soluciones factibles en el rango $[L, U]$ utilizando búsqueda binaria.

Las soluciones de tamaño $\nu \in [L, U]$ se generan y se prueba su factibilidad en función de una lista de prioridad especialmente formada. Los tamaños de las soluciones factibles se derivan mediante búsqueda binaria realizada en el intervalo $[L, U]$. Típicamente, se elige $(L + 3U)/4$ como el punto de partida en la búsqueda. Para cada solución creada σ de tamaño ν , se verifica la condición de factibilidad, es decir, se verifica si la solución forma un conjunto dominante. A continuación, se describe el marco general del algoritmo; y en la siguiente subsección se especifica cómo se crean las listas de prioridad.

Sea σ la solución actual de tamaño ν (inicialmente, σ es la solución factible proporcionada por el algoritmo de dos etapas):

- Si la solución σ es factible, entonces el límite superior actual U se actualiza a ν . El algoritmo procede con el siguiente valor de prueba (menor) ν del intervalo $[L, \nu)$ obtenido mediante el procedimiento de búsqueda binaria. Si todos los valores de prueba ν ya han sido verificados, entonces σ es una solución óptima. El algoritmo devuelve la solución σ y se detiene.
- Si la solución actual σ de tamaño ν no es factible, entonces se llama a *Procedure_Next*(ν) y se prueba la siguiente solución σ de tamaño ν de la lista de prioridad correspondiente.
- Si *Procedure_Next*(ν) devuelve *NULL*, es decir, ya se probaron todas las soluciones de tamaño ν y ninguna de ellas fue factible, el límite inferior actual L se actualiza a ν . El algoritmo continúa con el siguiente valor de prueba (más grande) ν del intervalo $[\nu, U)$ derivado mediante el procedimiento de búsqueda binaria. Si ya se probaron todos los valores de prueba ν , entonces σ es una solución óptima. El algoritmo devuelve la

solución σ y se detiene.

2.5.1. *Procedure_Next*(ν) y *Procedure_Priority_LIST*()

A continuación, se describe en qué orden se consideran las soluciones de un tamaño dado ν durante el proceso de enumeración. Para cada valor de prueba $\nu \in [L, U]$, las soluciones de tamaño ν se generan en un orden de prioridad especial que está diseñado para facilitar una convergencia más rápida hacia una solución factible. Para esto, se emplean dos procedimientos: *Procedure_Priority_LIST*() y *Procedure_Next*(ν). La subrutina *Procedure_Priority_LIST*() genera una lista de prioridad de vértices. Los primeros vértices incluidos en esta lista, potencialmente dominan un mayor número de vértices aún no dominados. Basado en el orden determinado por *Procedure_Priority_LIST*(), *Procedure_Next*(ν) determina la solución $\sigma_h(\nu)$ del tamaño de prueba actual ν en la iteración h . Primero se describe *Procedure_Priority_LIST*().

Ni los vértices del conjunto $Supp(G)$, ni los del conjunto $Leaf(G)$ son considerados al formar la lista de prioridad. Los vértices en $Supp(G)$ deben ser incluidos en cualquier conjunto dominante, por lo que forman parte de cualquier solución considerada durante el proceso de enumeración; por lo tanto, no es necesario incluir ninguno de los vértices del conjunto $Leaf(G)$ en ninguna solución formada (ver Corolario 2.2).

La lista de prioridad se forma en función de sus *grados activos*. La noción de grado activo fue introducida en [83]. El grado activo de un vértice se determina dinámicamente en cada iteración r . Sea $LIST_r$ la lista de prioridad de la iteración r en *Procedure_Priority_LIST*() (inicialmente, $LIST_0 := \emptyset$). El grado activo de un vértice v en la iteración r es el número de vecinos del vértice v en el conjunto $V(G) \setminus (LIST_{r-1} \cup Supp(G))$ sin contar los vecinos del vértice v que son adyacentes a un vértice en $LIST_{r-1} \cup Supp(G)$.

Supongamos que S es una solución factible (un conjunto dominante). La lista de prioridad de vértices se forma de manera iterativa a partir de los vértices del conjunto $S \setminus (Supp(G) \cup Leaf(G))$ ordenados en orden decreciente de sus grados activos. Los vértices restantes, es decir, aquellos del conjunto $V \setminus (S \cup Supp(G) \cup Leaf(G))$, se insertan de manera iterativa en

orden decreciente de sus grados en el grafo G . A continuación, se presenta una descripción formal del procedimiento.

Algorithm 9 Procedure_Priority_LIST

Input: A graph $G(V, E)$ and a feasible solution (dominating set) S .

Output: Priority list $LIST := LIST_r$.

$r := 0$;

$LIST_r := \emptyset$;

$S_r := S \setminus (Supp(G) \cup Leaf(G))$;

{ iterative step }

while $S_r \neq \emptyset$ **do**

$r := r + 1$;

$v_r :=$ any vertex of set S_{r-1} with the maximum active degree;

$S_r := S_{r-1} \setminus \{v_r\}$;

$LIST_r := LIST_{r-1} \cup \{v_r\}$;

end while

$D_r := V \setminus (S \cup Leaf(G))$;

while $D_r \neq \emptyset$ **do**

$r := r + 1$;

$v_r :=$ any vertex of set D_{r-1} with the maximum degree;

$D_r := D_{r-1} \setminus \{v_r\}$;

$LIST_r := LIST_{r-1} \cup \{v_r\}$;

end while

Ahora se describe $Procedure_Next(\nu)$ que, de manera iterativa, verifica si la solución formada actualmente $\sigma_h(\nu)$ es factible. Cabe señalar que la factibilidad de cada solución generada de un tamaño dado puede verificarse en tiempo $O(n)$. Si es factible, devuelve esa solución y continúa con el siguiente valor de prueba ν ; el procedimiento se detiene si ya se han considerado todos los tamaños de prueba. Para cada nuevo valor de prueba ν , la primera solución de ese tamaño contiene todos los vértices del conjunto $Supp(G)$ complementados por los primeros $\nu - |Supp(G)|$ vértices de la lista $LIST_r$ proporcionada por $Procedure_Priority_LIST()$.

Si la solución $\sigma_h(\nu)$ no es factible, la siguiente solución $\sigma_{h+1}(\nu)$ de tamaño ν se obtiene a partir de la solución $\sigma_h(\nu)$ mediante un intercambio de vértices de la siguiente manera. Sea $v \notin \sigma_h(\nu)$ el siguiente vértice de la lista de prioridad, y sea $v' \in \sigma_h(\nu)$ el vértice con el grado activo más pequeño. El vértice v' es sustituido por el vértice v en la solución $\sigma_{h+1}(\nu)$ (ten en cuenta que el grado activo de v debe ser menor que el de v'):

$$\sigma_{h+1}(\nu) := (\sigma_h(\nu) \setminus \{v'\}) \cup \{v\}.$$

Algorithm 10 Procedure *_Next_Aux*

Input: ν' , $\sigma_h(\nu')$, $LIST$, $\text{index_}LIST$, $Supp(G)$, h .
Output: $\sigma^*(\nu)$ or NIL .
if $\nu' = 0$ **then**
 if $Supp(G) \cup \sigma_h(\nu')$ is feasible solution **then**
 $\sigma^*(\nu) := Supp(G) \cup \sigma_h(\nu')$;
 return $\sigma^*(\nu)$;
 end if
 return $\sigma_{h-1}(\nu')$;
end if
 $i := \text{index_}LIST$;
while $i \leq |LIST|$ **do**
 Add $LIST[i]$ at the end of $\sigma_h(\nu')$;
 $\sigma_{h+1}(\nu') = \text{Procedure_Next_Aux}(\nu' - 1, \sigma_h(\nu'), LIST, i + 1, Supp(G), h + 1)$
 if $Supp(G) \cup \sigma_h(\nu')$ is feasible solution **then**
 $\sigma^*(\nu) := Supp(G) \cup \sigma_h(\nu')$;
 return $\sigma^*(\nu)$;
 else
 Remove the last element in $\sigma_h(\nu')$;
 $i := i + 1$;
 end if
end while
return NIL ;

Algorithm 11 Procedure *_Next*(ν)

Input: ν , $LIST$, $Supp(G)$.
Output: $\sigma^*(\nu)$ or NIL .
 $h := 0$;
 $\text{index_}LIST := 0$;
 $\nu' := \nu - Supp(G)$;
 $\sigma_h(\nu) := \emptyset$;
return Procedure *_Next_Aux*(ν' , $\sigma_h(\nu)$, $LIST$, $\text{index_}LIST$, $Supp(G)$, h);

2.5.2. Algoritmo BDS

Ahora se puede dar una descripción formal del algoritmo de enumeración implícita.

Algorithm 12 Algorithm_BDS

Input: A graph G .
Output: A $\gamma(G)$ -set S .
 $Supp(G) :=$ Set of support vertex of graph G ;
 $Leaf(G) :=$ Set of leaf vertex of graph G ;
 $L := \max\{\frac{n}{\Delta(G)+1}, \frac{2r}{3}, \frac{d+1}{3}, |Supp|\}$;
 $S := \sigma$; {Feasible solution proposed in [83]}
 $U := |S|$;
 $\nu := \lfloor (L + 3U)/4 \rfloor$;
 $LIST := Procedure_Priority_LIST(G, S)$;
 { iterative step }
while $U - L > 1$ **do**
 if $Procedure_Next(\nu)$ returns NIL **then**
 $L := \nu$;
 $\nu := \lfloor (L + 3U)/4 \rfloor$;
 else { A feasible solution was found ($\sigma_h(\nu)$) }
 $U := \nu$;
 $\nu := \lfloor (L + 3U)/4 \rfloor$;
 $LIST := Procedure_Priority_LIST()$;
 end if
end while

Sea $s = |Supp(G)|$ y $l = |Leaf(G)|$. A continuación, se proporciona una cota en el peor caso para el tiempo de ejecución del algoritmo de enumeración implícita.

Teorema 2.6. *La complejidad temporal del Algoritmo 12 en el peor caso es*

$$O\left(n \log(U - L) \binom{n - s - l}{\nu - s}\right).$$

Demostración. Dado que la búsqueda binaria se realiza en el intervalo $[L, U]$, el número total de iteraciones externas en el Algoritmo 12 (es decir, el número de diferentes tamaños ν) es, como máximo, $\log(U - L)$. Para un tamaño dado ν , el número de soluciones generadas de ese tamaño está acotado por $\binom{n-s-l}{\nu-s}$. En efecto, en el peor de los casos, se consideran todas las soluciones de cardinalidad ν . Según el Corolario 2.2, el conjunto $Supp(G)$ forma parte de todas las soluciones generadas, mientras que ningún vértice hoja pertenece a ninguna solución creada. Por lo tanto, $\binom{n-s-l}{\nu-s}$ es una cota superior sobre el número de soluciones de tamaño ν que genera el algoritmo.

Para establecer la factibilidad de la solución $\sigma_h(\nu)$, $Procedure_Next(\nu)$ verifica si cada vértice $x \in V(G)$ está en $\sigma_h(\nu)$ o si es adyacente a un vértice en $\sigma_h(\nu)$, lo cual claramente toma tiempo $O(n)$. Resumiendo lo anterior, se tiene la cota general $O\left(n \log(U - L) \binom{n-s-l}{\nu-s}\right)$ sobre el costo del algoritmo. ■

El teorema anterior se puede complementar con un análisis intuitivo que refleje el comportamiento en el peor de los casos del algoritmo exacto *BDS*. Para el propósito de esta estimación, supongamos que $\nu = \lfloor (U + L)/2 \rfloor$, teniendo en cuenta que el número máximo de combinaciones se alcanza para este ν particular. También se puede expresar ν en términos de n como $\nu = \lfloor (\frac{n}{2} + 1)/2 \rfloor = \frac{n}{4}$ utilizando $U = n/2$, $L = 1$, $s = 0$ y $l = 0$. En el caso de que exista una solución factible de tamaño $\nu = n/4$, entonces se tiene $O\left(n \log\left(\frac{n}{2} - 1\right) \binom{n}{n/4}\right)$.

Si no existe tal solución, el algoritmo aumenta el valor actual de ν . Entonces, en la siguiente iteración, el algoritmo enumerará las posibles soluciones con $\nu = n/3$, $O\left(n \log\left(\frac{n}{2} - 1\right) \binom{n}{n/3}\right)$. De manera similar, existe una solución factible de tamaño $n/3$ o no. Si no la hay, el algoritmo continúa con las soluciones de tamaño $\frac{5}{12}n$, y así sucesivamente.

El análisis anterior refleja un comportamiento muy bruto en el peor de los casos del algoritmo, pero no transmite su rendimiento práctico (ver Sección 2.6 para más detalles).

2.6. Resultados experimentales

En esta sección se describen los experimentos computacionales para conjuntos dominantes y algoritmos propuestos. Se implementaron los algoritmos en C++ utilizando el sistema operativo Windows 10 para 64 bits en una computadora personal con Intel Core i7-9750H (2.6 GHz) y 16 GB de RAM DDR4. Se generaron las instancias del problema utilizando diferentes métodos pseudoaleatorios para generar los grafos. El orden y el tamaño de cada instancia se generaron aleatoriamente utilizando la función *random()* dentro de los rangos respectivos. Se añadió cada nueva arista entre dos vértices aún no adyacentes de forma aleatoria hasta alcanzar el tamaño correspondiente.

Se analizaron 1316 instancias de referencia disponibles en [90]. Un análisis completo de los resultados para estas instancias se puede encontrar en [90]. Debido a las limitaciones de espacio, aquí se resumen los resultados para 45 instancias seleccionadas aleatoriamente en las Tablas 2.1 y 2.3, donde S denota el conjunto dominante inicial encontrado por *Greedy*, y S^* es el conjunto dominante purificado devuelto por los procedimientos de purificación

correspondientes. Las columnas marcadas con % representan el porcentaje de reducci3n del n3mero de v3rtices del conjunto dominante devueltos por los procedimientos de purificaci3n PP_1 - PP_4 con respecto a PP_0 .

La Tabla 2.1 presenta el porcentaje de mejora por los procedimientos de purificaci3n PP_1 - PP_4 para las 45 instancias de muestra con tama1os entre 800 y 15000.

V	E	<i>Greedy</i>	Procedimientos de Purificaci3n									
			PP_0		PP_1		PP_2		PP_3		PP_4	
			S	S*	S*	%	S*	%	S*	%	S*	%
840	310396	7	5	4	20.00	4	20.00	5	0.00	4	20.00	
1150	1185	500	470	431	8.30	430	8.51	430	8.51	428	8.94	
1650	1711	717	660	614	6.97	609	7.73	611	7.42	607	8.03	
1700	1740	725	681	634	6.90	632	7.20	633	7.05	632	7.20	
1900	1977	802	754	702	6.90	698	7.43	703	6.76	697	7.56	
2150	2287	909	850	787	7.41	781	8.12	784	7.76	781	8.12	
2300	2407	966	907	842	7.17	835	7.94	841	7.28	835	7.94	
2450	2588	1054	987	901	8.71	895	9.32	898	9.02	895	9.32	
2650	2744	1147	1059	981	7.37	975	7.93	977	7.74	975	7.93	
2800	2853	1225	1137	1048	7.83	1048	7.83	1044	8.18	1040	8.53	
2950	3011	1277	1186	1091	8.01	1085	8.52	1089	8.18	1085	8.52	
3300	3342	1440	1339	1232	7.99	1219	8.96	1221	8.81	1219	8.96	
3400	3518	1480	1382	1273	7.89	1268	8.25	1269	8.18	1268	8.25	
3500	3530	1537	1423	1305	8.29	1292	9.21	1296	8.92	1292	9.21	
3650	3705	1589	1464	1367	6.63	1351	7.72	1356	7.38	1351	7.72	
3950	4093	1688	1575	1464	7.05	1453	7.75	1455	7.62	1453	7.75	
4200	4202	1832	1707	1584	7.21	1564	8.38	1565	8.32	1564	8.38	
4550	4589	1963	1841	1701	7.60	1710	7.12	1701	7.60	1695	7.93	
4850	4915	2111	1955	1809	7.47	1790	8.44	1796	8.13	1789	8.49	
4950	5048	2164	2005	1865	6.98	1852	7.63	1853	7.58	1849	7.78	
5150	5288	2204	2044	1899	7.09	1888	7.63	1890	7.53	1888	7.63	
5300	5313	2293	2136	1999	6.41	1980	7.30	1982	7.21	1980	7.30	
5450	5489	2369	2211	2067	6.51	2052	7.19	2055	7.06	2052	7.19	
5550	5562	2424	2255	2084	7.58	2069	8.25	2074	8.03	2069	8.25	
5700	5770	2463	2287	2130	6.86	2115	7.52	2119	7.35	2114	7.56	
5950	6047	2614	2439	2256	7.50	2243	8.04	2246	7.91	2243	8.04	
6000	6010	2614	2427	2241	7.66	2237	7.83	2232	8.03	2225	8.32	
6200	6310	2694	2490	2304	7.47	2309	7.27	2289	8.07	2282	8.35	
6400	6466	2769	2582	2400	7.05	2395	7.24	2388	7.51	2383	7.71	
6500	6586	2804	2623	2440	6.98	2421	7.70	2424	7.59	2421	7.70	
6750	6870	2952	2721	2522	7.31	2494	8.34	2499	8.16	2494	8.34	
6850	6933	2969	2785	2579	7.40	2566	7.86	2566	7.86	2563	7.97	
7000	7038	3043	2832	2620	7.49	2623	7.38	2611	7.80	2601	8.16	
7100	7135	3067	2851	2646	7.19	2639	7.44	2633	7.65	2621	8.07	
7250	7357	3080	2879	2671	7.22	2654	7.82	2656	7.75	2654	7.82	

V	E	Greedy	Procedimientos de Purificación									
			PP ₀		PP ₁		PP ₂		PP ₃		PP ₄	
			S	S*	S*	%	S*	%	S*	%	S*	%
7350	7474	3177	2944	2744	6.79	2717	7.71	2722	7.54	2717	7.71	
7450	7497	3217	2980	2771	7.01	2772	6.98	2757	7.48	2751	7.68	
7550	7557	3299	3093	2844	8.05	2827	8.60	2835	8.34	2821	8.79	
7650	7696	3331	3095	2875	7.11	2849	7.95	2853	7.82	2846	8.05	
7700	7716	3332	3096	2875	7.14	2861	7.59	2862	7.56	2855	7.78	
7800	7804	3395	3176	2947	7.21	2933	7.65	2938	7.49	2924	7.93	
7950	7982	3480	3260	3022	7.30	3000	7.98	3004	7.85	3000	7.98	
8000	8126	3466	3215	2987	7.09	2985	7.15	2969	7.65	2957	8.02	
9900	24411032	16	14	13	7.14	13	7.14	13	7.14	13	7.14	
15000	56111357	14	12	12	0	12	0	12	0	12	0	

Tabla 2.1: Resultados de PP_0 - PP_4 para instancias de muestra

La tabla 2.2 presenta el porcentaje de mejora logrado por los procedimientos de purificación PP_1 - PP_4 con respecto a PP_0 , en promedio, para las 1316 instancias de referencia probadas.

PP_1	PP_2	PP_3	PP_4
9.17	9.48	8.88	9.63

Tabla 2.2: Promedio de purificación sobre todas las instancias

En promedio, sobre las 1316 instancias probadas, la mejor solución entre las cuatro soluciones generadas por los procedimientos de purificación PP_1 - PP_4 fue un 9.68 % más pequeña que el conjunto dominante devuelto por PP_0 . Los procedimientos de purificación PP_1 - PP_4 resultaron ser más eficientes para instancias densas, con una mejora de aproximadamente el 40 % de los vértices; sin embargo, en términos absolutos, el número de vértices purificados para instancias poco densas fue mayor.

En la Tabla 2.3, se presentan resultados para 45 instancias de referencia disponibles en [90] para las cuales se conocen las soluciones óptimas obtenidas por el Algoritmo BDS y la formulación ILP. Esta tabla compara la calidad de las soluciones obtenidas por los procedimientos de purificación PP_1 - PP_4 frente a las soluciones óptimas correspondientes y los límites superiores. El análisis comparativo completo para las 500 instancias de referencia con soluciones óptimas conocidas se puede encontrar en [90], y también está reflejado en la Figura 2.10. En todas las instancias probadas, las soluciones, en promedio, contenían 1/7 de la cantidad de

vértices del mejor límite superior conocido U , lo que representa una mejora del 85.71 %. Notablemente, entre las 500 instancias con el valor óptimo conocido, el algoritmo de dos etapas generó una solución óptima para el 46.33 % de las instancias, mientras que el error promedio de las soluciones no óptimas creadas fue de 1.01 vértices.

La cota superior U se debe a los siguientes resultados conocidos y propuestos en [62].

Teorema 2.7. *Si $G(V, E)$ es un grafo conexo de orden n , con grado mínimo $\delta(G)$ y grado máximo $\Delta(G)$, entonces $\gamma(G) \leq \frac{n}{2}$, $\gamma(G) \leq n - \Delta(G)$, y $\gamma(G) \leq \frac{n(\ln(\delta(G)+1))}{\delta(G)+1}$. Por lo tanto,*

$$U = \min \left\{ \frac{n}{2}, n - \Delta(G), \frac{n(\ln(\delta(G) + 1))}{\delta(G) + 1} \right\} \quad (2.11)$$

es una cota superior válida para $\gamma(G)$.

V	E	$\gamma(G)$	PP_1	PP_2	PP_3	PP_4	U
			$ S_1^* $	$ S_2^* $	$ S_3^* $	$ S_R^* $	
100	2687	6	7	7	7	7	26
106	3744	4	4	4	4	4	18
126	4651	6	6	6	6	6	30
128	6331	4	4	4	4	4	13
134	5359	7	7	7	7	7	32
138	7470	4	5	5	5	5	13
140	6710	6	6	6	6	6	24
144	7267	5	5	5	5	5	25
148	8664	3	3	3	4	3	16
150	7950	5	5	5	5	5	27
154	8304	6	6	6	6	6	24
170	11640	4	4	4	4	4	14
172	10665	6	6	6	6	6	26
176	9978	7	7	7	7	7	39
182	12100	6	6	6	6	6	25
194	13675	5	5	5	5	5	34
196	14229	6	6	6	6	6	26
198	12936	7	7	7	7	7	42
200	16389	4	4	4	4	4	19
204	15169	5	5	5	5	5	35
222	20370	4	4	4	4	4	18
224	16923	7	7	7	7	7	43
234	22709	4	4	4	4	4	21
238	20902	5	5	5	5	5	39
264	27175	5	5	5	6	5	30
326	37878	7	7	7	7	7	60
328	45703	4	4	4	4	4	24

$ V $	$ E $	$\gamma(G)$	PP_1	PP_2	PP_3	PP_4	U
			$ S_1^* $	$ S_2^* $	$ S_3^* $	$ S_R^* $	
330	43714	6	6	6	6	6	35
332	41661	5	5	5	5	5	56
342	47142	7	7	7	7	7	36
344	50364	4	4	4	4	4	26
370	58531	4	4	4	4	4	28
400	65545	6	6	6	6	6	37
458	77296	7	7	7	7	7	79
506	95091	6	6	6	7	6	91
538	125734	5	5	5	5	5	47
612	140918	7	7	7	8	7	101
714	198539	5	5	5	5	5	117
724	199141	7	7	7	7	7	124
790	243725	5	5	5	5	5	130
794	240592	7	7	7	7	7	138
840	310396	4	4	4	5	4	67
844	306681	7	7	7	7	7	72
870	296197	5	7	7	7	7	145
1098	533220	5	5	5	5	5	81

Tabla 2.3: Comparación entre los resultados de PP_1 - PP_4 , $\gamma(G)$, y U

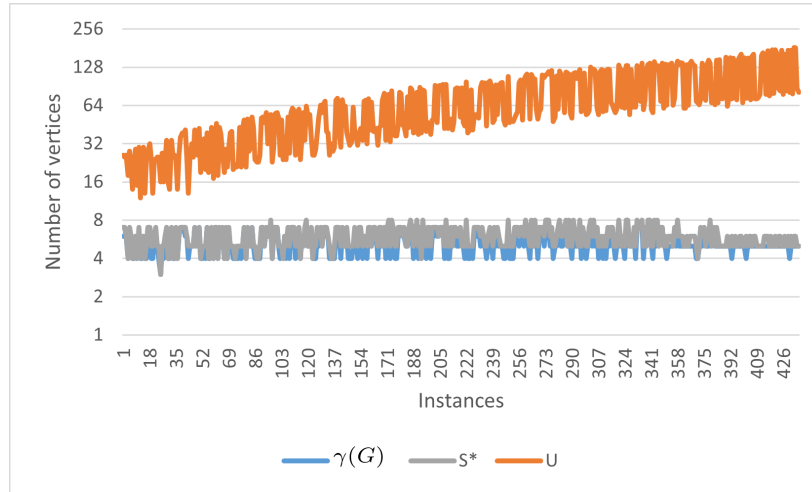
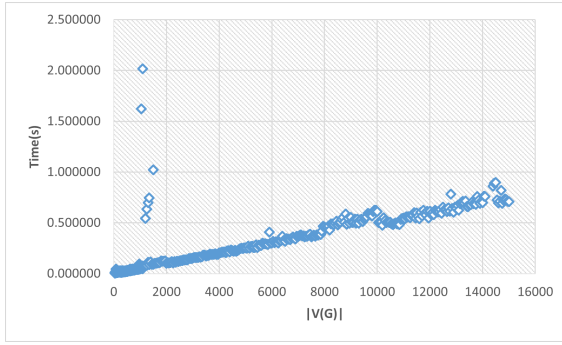
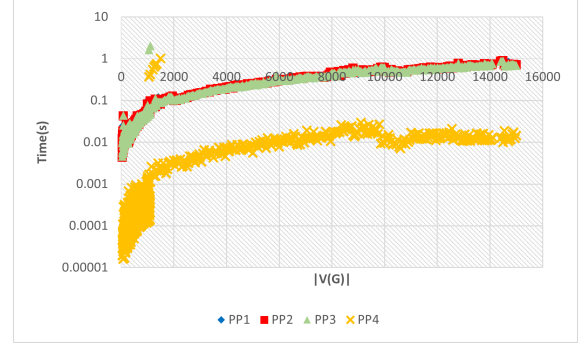


Figura 2.10: Resultados de los procedimientos de purificación ($\min\{S_i^*\}$), $\gamma(G)$, y U

En cuanto a los tiempos de ejecución, para las instancias de referencia más grandes con 15000 vértices, los cuatro procedimientos finalizaron en menos de dos segundos. La Figura 2.11 representa los tiempos de ejecución para todas las instancias probadas.



(a) Tiempos máximos de ejecución entre PP_{1-4} para todas las instancias.

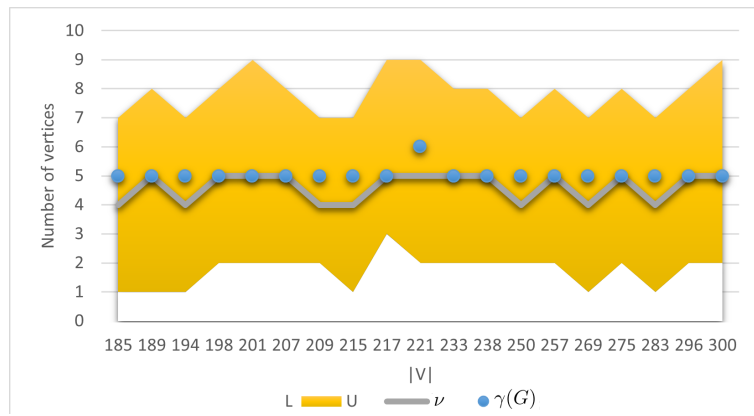


(b) Tiempos de ejecución por procedimiento

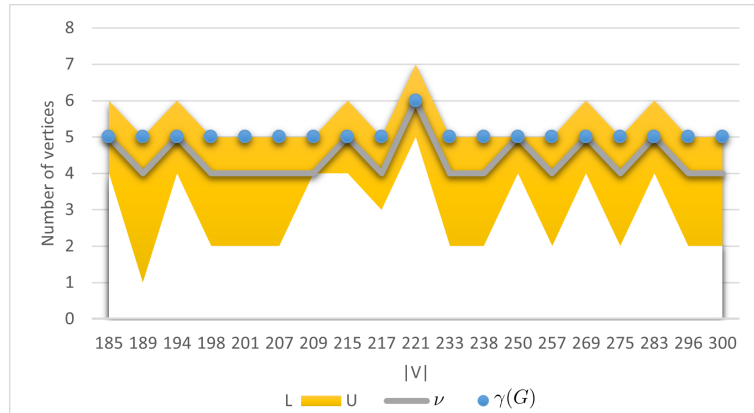
Figura 2.11: Tiempo de los procedimientos de purificación

En lo que sigue de esta sección, los algoritmos exactos de [111], [70], la formulación ILP de la Sección 2.4 y el algoritmo exacto propuesto se referirán como MSC^1 , MSC^2 , ILP y BDS, respectivamente. Para estos algoritmos, se analizaron más de 1100 instancias de las disponibles públicamente en [90]. Un resumen completo de los resultados para estas instancias se puede encontrar en [90]. Se eligieron 72 de estas instancias al azar y se reportan más adelante los resultados para ellas.

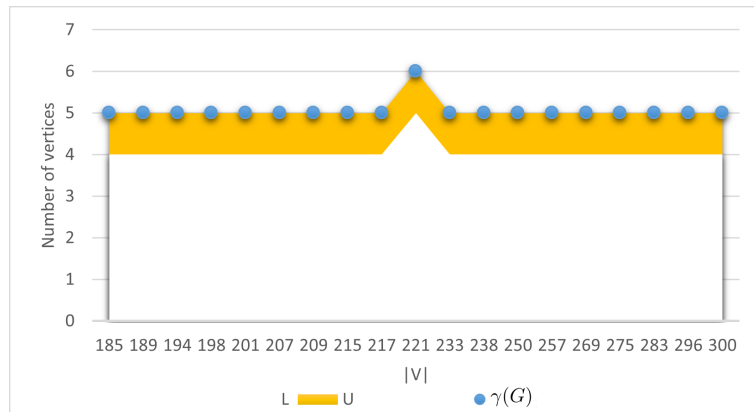
Parte de la entrada para el algoritmo BDS es una solución factible que puede ser obtenida mediante cualquier heurística. Dado que los resultados del algoritmo de dos etapas 3 proporciona la mejor aproximación que se conoce, se eligió este algoritmo en particular para generar las entradas del algoritmo BDS. En la Figura 2.12 se representa cómo se actualizan las cotas inferior y superior durante las tres primeras iteraciones del algoritmo BDS y cómo el número de nodos en las soluciones factibles generadas se reduce paso a paso para 19 instancias de tamaño medio y densidad media.



(a) Iteración 1



(b) Iteración 2



(c) Iteración 3

Figura 2.12: Cota inferior y superior del intervalo de búsqueda para el algoritmo BDS en tres iteraciones consecutivas, y $\gamma(G)$

La Tabla 2.4 presenta los resultados para cuasi-árboles de tamaño pequeño con una densidad de aproximadamente 0.2, donde la formulación ILP resultó ser la de mejor rendimiento debido a la pequeña cantidad de variables en el modelo lineal. Los dos algoritmos MSC resultaron ser

más rápidos que el algoritmo BDS, ya que los conjuntos de vértices utilizados en los primeros dos algoritmos son muy pequeños y las reducciones realizadas tienen costos computacionales bajos.

No.	$ V(G) $	$ E(G) $	Tiempo				Cota Inferior				$\gamma(G)$	Cota Superior	
			ILP	BDS	MSC^1	MSC^2	$\frac{n}{\Delta(G)+1}$	$\frac{d+1}{3}$	$\frac{2r}{3}$	$ Supp(G) $		$ S $	$n - \Delta(G)$
1	50	286	0.16	1.89	0.61	0.23	2	1	1	2	6	6	33
2	60	357	0.11	3.13	0.71	0.29	2	1	1	1	6	7	40
3	70	524	0.12	7.04	1.46	0.37	2	1	1	1	6	6	45
4	80	678	0.13	12.59	2.81	0.79	2	1	1	1	6	7	53
5	90	842	0.17	390.90	5.18	2.31	3	1	1	1	7	8	63
6	100	1031	0.21	803.74	8.31	3.01	2	1	1	2	7	7	67
7	101	1051	2.17	804.20	8.69	3.17	3	1	1	1	7	7	70
8	106	1154	1.78	1143.92	10.82	2.93	2	1	1	1	7	8	71
9	113	1306	1.29	1594.92	16.25	4.27	3	1	1	1	7	7	77
10	117	1398	2.56	3364.4	19.95	5.33	3	1	1	2	8	9	84
11	121	1493	3.22	2240.49	23.11	4.86	3	1	1	1	7	7	87

Tabla 2.4: Resultados para grafos con densidad $\approx 0,2$.

Para los grafos de densidad media donde los cuatro algoritmos exactos lograron detenerse con una solución óptima, los algoritmos BDS e ILP proporcionaron mejores tiempos de ejecución que los algoritmos MSC. La formulación ILP resultó en tiempos peores que el algoritmo MSC^2 para instancias con más de 295 vértices, y el algoritmo BDS fue más rápido que la formulación ILP para instancias a partir de 255 vértices, véase la Tabla 2.5 y la Figura 2.13 a continuación.

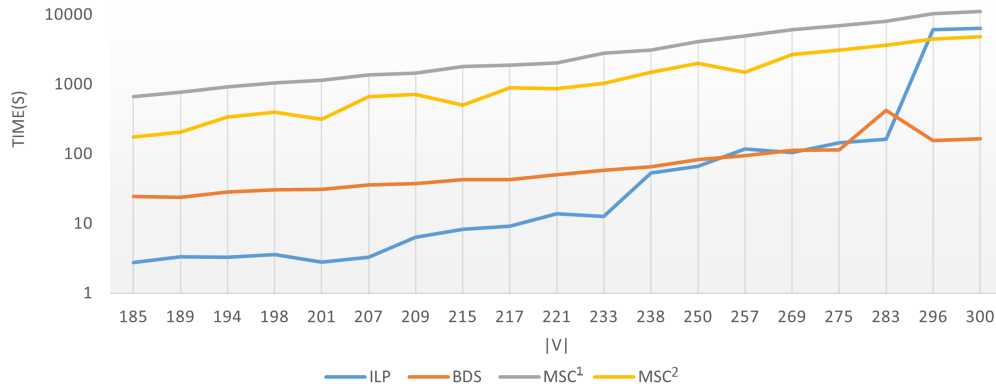


Figura 2.13: Tiempos de ejecución para los cuatro algoritmos exactos (se aplicó escala logarítmica).

No.	$ V(G) $	$ E(G) $	Tiempo				Cota Inferior				$\gamma(G)$	Cota Superior	
			ILP	BDS	MSC^1	MSC^2	$\frac{n}{\Delta(G)+1}$	$\frac{d+1}{3}$	$\frac{2r}{3}$	$ Supp(G) $		$ S $	$n - \Delta(G)$
1	185	6849	2.75	24.66	666.83	176.18	1	1	1	1	5	6	93
2	189	7147	3.35	23.85	770.36	206.02	1	1	1	1	5	7	91
3	194	7529	3.29	28.29	915.36	335.88	1	1	1	1	5	6	96
4	198	7842	3.61	30.53	1041.98	395.03	2	1	1	1	5	6	101
5	201	8081	2.78	31.03	1136.67	313.15	2	1	1	2	5	7	103
6	207	8569	3.28	35.94	1361.77	659.35	2	1	1	1	5	6	105
7	209	8735	6.40	37.35	1454.44	715.74	1	2	1	1	5	5	104
8	215	9243	8.25	42.91	1810.09	499.96	1	1	1	1	5	6	103
9	217	9415	9.16	42.88	1892.35	890.39	1	1	1	3	5	6	107
10	221	9765	13.73	49.91	2011.15	866.37	2	1	1	1	6	7	115
11	233	10852	12.74	57.83	2793.59	1028.85	2	1	1	1	5	6	118
12	238	11322	53.59	65.05	3093.64	1485.16	2	1	1	1	5	6	123
13	250	12491	66.09	83.19	4065.44	1988.04	1	2	1	1	5	5	125
14	257	13199	118.24	93.98	4919.85	1478.89	2	1	1	1	5	6	131
15	269	14459	104.05	112.609	6102.11	2677.25	1	1	1	1	5	6	134
16	275	15111	143.66	114.446	6887.98	3082.72	2	1	1	1	5	6	144
17	283	16002	163.58	418.72	8040.4	3649.57	1	1	1	1	5	6	141
18	296	17505	6071.5	155.861	10269.4	4467.34	2	1	1	1	5	6	152
19	300	17981	6318.52	165.301	11035.8	4811.58	2	1	1	1	5	7	151

Tabla 2.5: Resultados para grafos con densidad $\approx 0,5$.

La Tabla 2.6 presenta los resultados para instancias más grandes con una densidad $\approx 0,8$, donde los algoritmos MSC no lograron detenerse dentro del límite de tiempo de 8 horas; los algoritmos BDS e ILP se completaron en menos de 6 minutos para la instancia más grande con 1098 vértices. Estas instancias son densas, y la formulación ILP resulta en tiempos de ejecución más cortos (ver Figura 2.14). Estos grafos densos son cuasi-completos, por lo que $\gamma(G)$ es muy pequeño, oscilando entre 4 y 5.

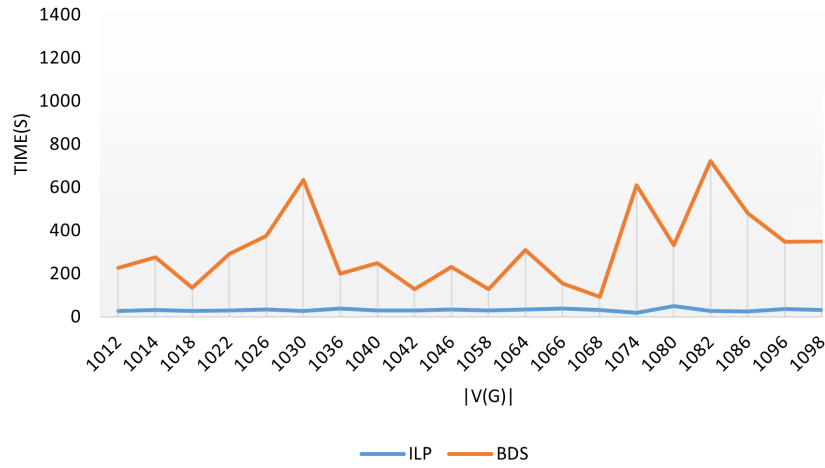


Figura 2.14: BDS vs. ILP para grafos densos

No.	V	E	Tiempo		Cota Inferior				$\gamma(G)$	Cota Superior	
			ILP	BDS	$\frac{n}{\Delta(G)+1}$	$\frac{d+1}{3}$	$\frac{2r}{3}$	$ Supp(G) $		S	$n - \Delta(G)$
1	1012	452294	26.27	227.34	1	2	2	2	5	6	76
2	1014	454116	31.72	275.10	1	2	2	2	5	6	79
3	1018	406867	27.23	135.86	1	2	2	2	5	7	175
4	1022	461340	29.27	291.41	1	2	2	2	5	6	80
5	1026	413355	34.74	375.12	1	2	2	2	5	7	176
6	1030	416585	27.07	633.91	1	2	2	2	5	7	164
7	1036	474219	38.03	198.76	1	2	2	2	5	6	86
8	1040	477909	29.73	248.36	1	2	2	2	5	6	83
9	1042	426451	28.48	127.95	1	2	2	2	5	7	174
10	1046	483492	32.97	231.50	1	2	2	2	5	6	78
11	1058	439752	29.10	128.97	1	2	2	2	5	7	162
12	1064	500420	33.63	310.40	1	2	2	2	5	6	83
13	1066	446505	37.56	155.74	1	2	2	2	5	6	173
14	1068	504205	31.27	92.06	1	2	2	2	4	6	81
15	1074	453307	19.43	610.30	1	2	2	2	5	7	174
16	1080	515722	49.56	333.09	1	2	2	2	5	6	79
17	1082	460117	26.83	723.12	1	2	2	2	5	7	183
18	1086	463560	24.05	480.90	1	2	2	2	5	7	182
19	1096	531243	35.54	347.71	1	2	2	2	5	6	87
20	1098	533220	31.33	349.20	1	2	2	2	5	6	81

Tabla 2.6: Resultados para grafos con densidad $\approx 0,8$

2.7. Conclusiones

Este capítulo abordó la resolución eficiente del problema del conjunto dominante mínimo, un problema NP-hard. Las soluciones presentadas, basadas en métodos heurísticos, demostraron mejoras significativas respecto a los enfoques tradicionales y exactos en términos de eficiencia y tiempo de cómputo. Se expuso un enfoque de dos etapas, en el cual un algoritmo voraz inicial fue seguido por un proceso de purificación del conjunto dominante obtenido. Aunque el primer algoritmo resultó ser similar a otros del estado del arte, la etapa de purificación mejoró notablemente la calidad de la solución en la práctica. Este enfoque no solo fue eficiente en términos computacionales, sino que también se identificaron casos específicos de grafos en los que el algoritmo proporcionó directamente la solución óptima.

Además, se introdujeron variantes del algoritmo de purificación que superaron al método inicial en eficiencia, mostrando un rendimiento destacado en pruebas con numerosas instancias del problema. Para las instancias donde se conocía la solución óptima, el algoritmo de dos etapas logró alcanzar dicha solución en un alto porcentaje de los casos y, en los demás, el error fue mínimo.

Por último, se formuló el problema como un programa lineal entero (ILP) y se propuso un algoritmo exacto de enumeración implícita. En conjunto, este capítulo proporcionó enfoques robustos y eficientes para la resolución del problema del conjunto dominante mínimo, demostrando tanto la aplicabilidad práctica de los algoritmos heurísticos como su capacidad para resolver problemas de gran tamaño de manera eficiente, con soluciones cercanas al óptimo en la mayoría de los casos.

Conjunto Dominante Global

3.1. Introducción

Un conjunto de vértices de un grafo G es un conjunto dominante global si es un conjunto dominante tanto del grafo G como de su complemento. El objetivo de este capítulo es encontrar un conjunto dominante global con la mínima cardinalidad, un conjunto dominante global *óptimo*. El problema de dominación global (GDP, por sus siglas en inglés) fue introducido por [97] y [20]. El GDP es conocido por ser NP -hard [20]. En este capítulo se muestra que el GDP sigue siendo NP -hard para grafos planares y grafos split. También se proporcionan algunas familias de grafos para los cuales las heurísticas que se proponen aquí obtienen una solución óptima. Estas familias de grafos, que contienen grafos estrella como subgrafos, tienen aplicaciones prácticas inmediatas en redes sociales [41].

En este capítulo, se presenta además, una formulación de programación lineal entera (ILP) para el GDP y un algoritmo de enumeración implícita (IE) alternativo. Se prueba el modelo ILP utilizando los solvers *intlinprog* y *CPLEX*. Basados en las soluciones óptimas obtenidas por los métodos exactos para las 1000 instancias de referencia existentes con hasta 8000 vértices en [91] y en las cotas superiores previamente conocidas para el problema, se estimó la eficiencia de las tres heurísticas que se proponen aquí. Las heurísticas incluyen una etapa final de *purificación* que reduce el tamaño de las soluciones factibles entregadas por las

heurísticas. Esta etapa logró reducir el tamaño de las soluciones factibles obtenidas por las heurísticas básicas en más del 10 %, en promedio. Se probaron las heurísticas en 2284 instancias de referencia con hasta 14000 vértices [91] y en 6 nuevas instancias de gran tamaño con hasta 25000 vértices. Las soluciones factibles para las instancias más grandes se obtuvieron en menos de dos minutos, mientras que las heurísticas se detuvieron en unos pocos segundos para todas las instancias de tamaño moderado con hasta 1000 vértices. Notablemente, para aproximadamente el 56.75 % de las 1000 instancias para las cuales se obtuvieron soluciones óptimas, al menos una de las heurísticas también generó una solución óptima. Además, el error absoluto promedio para las instancias restantes fue de un solo vértice.

Se establecieron los siguientes hechos para juzgar el comportamiento asintótico en el peor de los casos de las heurísticas propuestas. Para las instancias de tamaño moderado con hasta 1100 vértices, obtuvieron radios de aproximación en el rango $[1,0135, 1,25]$, donde el promedio fue de 1,1691. Para las instancias de mayor tamaño con hasta 8000 vértices, los radios de aproximación correspondientes esencialmente disminuyeron hasta 1.0003, con un promedio de 1.0050. Como se puede ver en la Figura 3.11, asintóticamente, el radio de aproximación se acercó a un número muy cercano a 0. Esto nos hace creer que las soluciones factibles entregadas por las heurísticas están muy cerca del óptimo para las instancias de gran tamaño para las cuales no se pueden obtener soluciones óptimas en la actualidad.

Una descripción detallada del estudio experimental se puede encontrar en la Sección 3.6. En las Secciones 3.2 y 3.3 se dan definiciones básicas, se hace un breve resumen de las cotas superiores e inferiores conocidas anteriormente para el problema, y se estudian cuestiones de complejidad computacional para grafos generales y algunos tipos de grafos restringidos. En la Sección 3.4 se da una formulación ILP del problema de dominación global y también se describe un algoritmo exacto. En las Secciones 3.5.1, 3.5.2 y 3.5.3 se describen los algoritmos heurísticos y se establecen familias de grafos para las cuales proporcionan soluciones óptimas. El procedimiento de purificación se describe en la Sección 3.5.4.

3.2. Propiedades básicas

Se inicia esta sección con algunas nociones y notaciones necesarias. Sea $G = (V, E)$ un grafo simple de orden $|V| = n$ y tamaño $|E| = m$. Para un conjunto no vacío $D \subseteq V$ y un vértice $v \in V$, $N_D(v)$ denota el conjunto de vecinos de v en D , y $N_D[v] = N_D(v) \cup v$. El grado de v en D se denotará por $d_D(v) = |N_D(v)|$. De manera similar, $\overline{N}_D(v)$ denota el conjunto de no vecinos de v en D , $\overline{N}_D[v] = \overline{N}_D(v) \cup v$, y $\overline{d}_D(v)$ denota la cardinalidad del conjunto $\overline{N}_D(v)$ (es decir, $\overline{d}_D(v) = |\overline{N}_D(v)|$).

Para simplificar, se usará frecuentemente $N(v)$, $N[v]$, $d(v)$, $\overline{N}(v)$ y $\overline{N}[v]$ en lugar de $N_V(v)$, $N_V[v]$, $d_V(v)$, $\overline{N}_V(v)$ y $\overline{N}_V[v]$, respectivamente. Por lo tanto, $N_A(B) = \bigcup_{v \in B} N_A(v)$ y $N_A[B] = \bigcup_{v \in B} N_A[v]$, donde A y B son subconjuntos de V . Para simplificar, si $A = V$, entonces $N(B) = \bigcup_{v \in B} N(v)$ y $N[B] = \bigcup_{v \in B} N[v]$.

Definición 3.1. *Un subconjunto no vacío $D \subseteq V$ se llama conjunto dominante global de G (o GDS, por sus siglas en inglés) si D es un conjunto dominante tanto de G como de $\overline{G}$, donde $\overline{G}$ es el grafo complemento del grafo G .*

Definición 3.2. *El número de dominación global, $\gamma_g(G)$, es el número de vértices en un conjunto dominante global de cardinalidad mínima.*

Un conjunto dominante global con cardinalidad $\gamma_g(G)$ será llamado como $\gamma_g(G)$ -conjunto.

Si $G_1, G_2, \dots, G_r$ con $r \geq 2$, son las componentes conexas de un grafo G , entonces cualquier GDS mínimo de G está formado por un conjunto dominante mínimo de cada subgrafo G_i . Esto implica que,

$$\gamma_g(G) = \sum_{i=1}^r \gamma(G_i).$$

Además, se sabe que $\gamma_g(G) = \gamma_g(\overline{G})$.

Ahora se resumen brevemente las cotas inferiores y superiores conocidas para el problema y se obtiene una cota inferior general L utilizando los siguientes resultados conocidos.

Teorema 3.1 ([60]). *Si $G(V, E)$ es un grafo de orden n , entonces*

- $\gamma(G) \geq \frac{n}{\Delta(G)+1},$
- $\gamma(G) \geq \frac{2r(G)}{3},$
- $\gamma(G) \geq \frac{d(G)+1}{3},$ y
- $\gamma(G) \geq |Supp(G)|.$

Teorema 3.2 ([37]). *Si $G(V, E)$ es un grafo, $\gamma_g(G) \geq \gamma(G)$.*

El siguiente resultado es una consecuencia inmediata de los Teoremas 3.1 y 3.2:

Corolario 3.1. *Sea $G(V, E)$ un grafo de orden n . Entonces*

$$L = \max \left\{ \frac{n}{\Delta(G)+1}, \frac{2r(G)}{3}, \frac{d(G)+1}{3}, |Supp(G)| \right\}$$

es una cota inferior de $\gamma_g(G)$.

La cota superior general U se obtiene utilizando los siguientes resultados conocidos.

Teorema 3.3 ([20]). *Si $G(V, E)$ es un grafo con grado máximo $\Delta(G)$, entonces $\gamma_g(G) \leq U_1 = \min\{\Delta(G), \Delta(\overline{G})\} + 1$.*

Teorema 3.4 ([20]). *Para cualquier grafo $G(V, E)$ de grado mínimo $\delta(G)$, $\gamma_g(G) \leq U_2 = \delta(G) + 2$ si $\delta(G) = \delta(\overline{G}) \leq 2$; de lo contrario $\gamma_g(G) \leq U_2 = \max\{\delta(G), \delta(\overline{G})\} + 1$.*

Los Teoremas 3.3 y 3.4 se resumen en el siguiente corolario:

Corolario 3.2. *Sea $G(V, E)$ un grafo. Entonces $U = \min\{U_1, U_2\}$ es una cota superior de $\gamma_g(G)$.*

3.3. Resultados de NP-complejidad

Como ya se señaló, el concepto de dominación global fue introducido en [97]. Dutton, en [20], introdujo el llamado problema de dominación por factores y demostró que el problema de dominación global puede reducirse al problema de dominación, el cual es conocido por ser NP-hard [53]. Demos algunas definiciones de [20] y luego presentemos un esbozo de esta reducción intuitiva.

Digamos que el grafo $G = (V, E)$ puede descomponerse en k -factores ($k \geq 2$), $G_1, \dots, G_k$, $G_i = (V_i, E_i)$ si para cada grafo G_i , $V_i = V$ y el conjunto de todas las aristas $E_1, \dots, E_k$ forma una partición del conjunto E .

Definición 3.3. *Dada una descomposición del grafo $G = (V, E)$ en los k -factores $G_1, \dots, G_k$, $D \subseteq V$ se llama un conjunto dominante por factores si D es un conjunto dominante para cada G_i , $i = 1, \dots, k$.*

Definición 3.4. *El número de dominación por factores, $\gamma(G_1, \dots, G_k)$, es el número de elementos del conjunto dominante por factores más pequeño.*

Ahora consideremos una 2-factorización de un grafo completo K_n , y cualquier subgrafo $G = (V, E)$ con $|V| = n$. Sea $G_1 = G$ y $G_2 = \bar{G}$. Entonces, note que G_1, G_2 es una 2-factorización del grafo K_n . A partir de esto, se obtiene inmediatamente que el problema de dominación por factores para el grafo K_n con $k = 2$, es equivalente al problema de dominación global en G . Observe que el problema de dominación por factores con $k = 1$ es el problema de dominación estándar, que es NP-hard. Por lo tanto, el problema de dominación por factores también es NP-hard para cualquier $k \geq 2$, en particular, para $k = 2$.

Teorema 3.5 ([20]). *El problema de dominación global es NP-hard.*

En la parte restante de esta sección, se muestra que el problema de dominación global en tipos restringidos de gráficos sigue siendo NP-hard. En particular, se consideran los grafos split y planar [81, 82]. Se usará D (D_g) para un conjunto dominante (global) mínimo.

3.3.1. Grafo Split

Un *grafo split* es un grafo en el cual los vértices pueden ser particionados en un clique C y un conjunto independiente I (ver [35, 51, 72, 103]). Sea G un grafo split conexo y sea C un clique máximo en G . Dado que el grafo G es conexo, existe una arista desde cualquier vértice del conjunto I hacia un vértice del conjunto C . En particular, se cumple el siguiente lema.

Lema 3.1 ([14]). *Supongamos que G es un grafo split conexo con un clique máximo C y un conjunto independiente I . Entonces, cualquier conjunto dominante mínimo D del grafo G es un subconjunto (no necesariamente propio) del conjunto C .*

Lema 3.2. *Supongamos que G es un grafo split conexo con un clique máximo C , un conjunto independiente I y un conjunto dominante mínimo D . Si $D = C$, entonces $D_g = D$.*

Demostración. Dado que $D = C$ y C es máximo, para cada vértice $y \in I$, debe existir un vértice $z \in C$ que no esté asociado con el vértice y . Entonces, el vértice z dominará al vértice y en el complemento $\overline{G}$. Por lo tanto, D también es un conjunto dominante global. ■

Teorema 3.6. *Dado un grafo split conexo G , cualquier conjunto dominante global para este grafo se forma ya sea por el conjunto dominante de G o por el conjunto dominante de G y uno (o más) vértices del conjunto I . Por lo tanto, el problema de dominación global en el grafo G se reduce al problema de dominación en ese grafo, y por ende es NP-hard.*

Demostración. Consideremos dos tipos básicos de un grafo split G . En el primer tipo de grafo split G , cada vértice $v \in C$ está asociado con al menos un vértice de grado uno (pendiente) $x \in I$ (es decir, x es un vecino privado de v). Para tal G , solo hay un conjunto dominante mínimo $D = C$. El conjunto D también es un conjunto dominante global para el grafo G según el Lema 3.2, y la primera afirmación del teorema se sostiene.

Supongamos ahora que G es del segundo tipo, es decir, hay un vértice en el conjunto I asociado con dos o más vértices en el conjunto C , ver la Figura 3.1. Note que, dado que el grafo G es conexo, cada vértice en el conjunto I está asociado con al menos un vértice del conjunto C . Sea $u \in C$ un vértice sin vecino privado en el conjunto I (en el grafo de la Figura 3.1, el clique C está formada por los cuatro vértices del rectángulo en la parte inferior derecha de esa figura, este clique contiene un vértice de tipo v y tres vértices de tipo u). Es fácil ver que un conjunto dominante mínimo D del grafo G es un subconjunto propio del conjunto C (ver Lema 3.1). De hecho, dado que hay un vértice en el conjunto I asociado con dos o más vértices de tipo u en el conjunto C , el conjunto D está formado por los vértices del clique C pero omitiendo uno (o más) vértices de tipo u .

Note que estos vértices de tipo u no serán dominados por el conjunto D en el complemento $\overline{G}$. Por lo tanto, para formar un conjunto dominante global, se necesita añadir al conjunto D uno (o más) vértices del conjunto I , que no sean adyacentes a los vértices de tipo u mencionados en $C \setminus D$.

Así queda demostrado que el problema del conjunto dominante global en grafos split se reduce al problema de dominación para ambos tipos de grafos split. Este último problema se conoce

como NP-hard [14]. ■

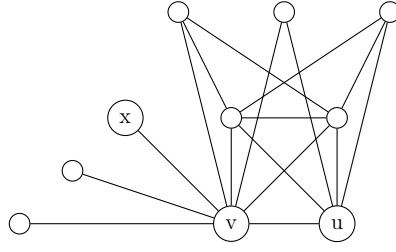


Figura 3.1: Grafo split con vértices de tipo x y u

3.3.2. Grafo Planar

En [42] se estudió el problema de dominación global en grafos planares. En particular, se obtuvieron cotas superiores para tipos especiales de grafos planares. Sin embargo, la complejidad del problema no se ha abordado aún, ni en la referencia mencionada ni en ninguna otra que conozcamos. Aquí se demuestra que el problema de dominación global en grafos planares es NP-hard.

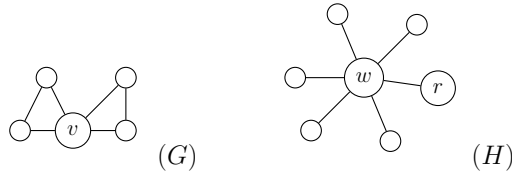


Figura 3.2: Grafo planar H (grafo estrella) y G

Para empezar, se define el *producto enraizado* de dos grafos. Dado un grafo G con n vértices y un grafo (árbol) H con el vértice raíz r , el producto enraizado $G \circ_r H$ es el grafo obtenido de los grafos G y H de la siguiente manera. Cada vértice v del grafo G se extiende con una copia del árbol H , es decir, se asocia el vértice v con el vértice r , extendiendo de esta manera el grafo G . En la Figura 3.2 se ilustra un grafo plano G y el árbol H . La Figura 3.3 ilustra el producto enraizado de estos dos grafos.

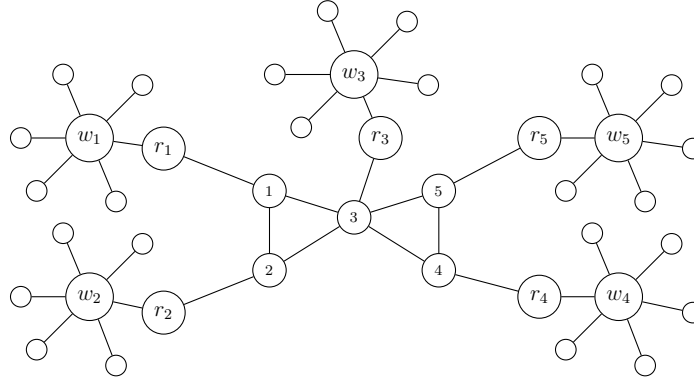


Figura 3.3: Grafo $G \circ_r H$

Teorema 3.7. *Dado un grafo planar G de orden n y un grafo estrella H de orden mayor que dos con soporte w , el grafo $G \circ_r H$ es planar. Además, el conjunto dominante global del grafo $G \circ_r H$ es el conjunto dominante del grafo G aumentado con los n vértices soporte de las n copias del grafo H . Por lo tanto, el problema de dominación global en el grafo G es NP -hard.*

Demostración. Dado que ambos grafos G y H son planares, el producto enraizado de estos grafos claramente produce otro grafo planar. Se demostrará la parte restante del teorema. Sea D un conjunto dominante mínimo del grafo G . Es fácil ver que $D \cup \{w_1, w_2, \dots, w_n\}$ es un conjunto dominante global del grafo $G \circ_r H$. De hecho, cada vértice de la i -ésima copia H_i del grafo H es dominado por el vértice w_i , por lo que $D \cup \{w_1, w_2, \dots, w_n\}$ es un conjunto dominante del grafo $G \circ_r H$.

$D \cup \{w_1, w_2, \dots, w_n\}$ también es un conjunto dominante del complemento del grafo $G \circ_r H$, ya que el vértice w_i dominará cualquier vértice del complemento que no esté en H_i , mientras que todos los vértices del grafo H_i serán dominados por cualquier w_j , $j \neq i$. La Figura 3.3 representa el producto enraizado $G \circ_r H$ de los dos grafos de la Figura 3.2, donde el conjunto D está formado por el vértice $\{v\}$.

Note ahora que $D \cup \{w_1, w_2, \dots, w_n\}$ es un conjunto dominante mínimo para el grafo $G \circ_r H$, ya que claramente al menos un vértice de cada grafo H_i debe incluirse en cualquier conjunto dominante de ese grafo. Por lo tanto, $D \cup \{w_1, w_2, \dots, w_n\}$ es el conjunto dominante global mínimo para el grafo $G \circ_r H$, ya que coincide con el conjunto dominante mínimo de ese grafo. La última afirmación del teorema se sigue del hecho de que el problema del conjunto dominante en grafos planares es NP -hard [42]. ■

3.4. Algoritmos exactos

3.4.1. ILP formulation of the problem

Dado un grafo simple no dirigido $G = (V, E)$ de orden n , su matriz de vecindad cerrada $A = (a_{ij})_{n \times n}$ es una matriz de 0-1 definida de la siguiente manera: $a_{ij} = a_{ji} = 1$ si $\{i, j\} \in E(G)$ o $i = j$, y $a_{ij} = a_{ji} = 0$ en caso contrario.

Dada un conjunto dominante global D de cardinalidad $\gamma_g(G)$ e $i \in V(G)$, se definen las variables de decisión de la siguiente manera:

$$x_i = \begin{cases} 1, & \text{if } i \in D \\ 0, & \text{otherwise,} \end{cases} \quad (3.1)$$

El problema del conjunto dominante global puede ser formulado de la siguiente manera:

$$\min \sum_{i=1}^n x_i, \quad (3.2)$$

$$\text{subject to : } \sum_{j=1}^n a_{ij}x_j \geq 1, \forall i \in V(G), \quad (3.3)$$

$$\sum_{j=1}^n (1 - a_{ij})x_j \geq 1, \forall i \in V(G), \quad (3.4)$$

$$x_i \in \{0, 1\}, \forall i \in V(G), \quad (3.5)$$

Aquí se desea minimizar la función objetivo 3.2. Se tienen $2n$ restricciones (3,3 y 3,4), garantizando que para cada $i \in V(G)/D$, existe al menos un vértice en D que es adyacente a i mientras que otro no lo es.

3.4.2. Algoritmo de enumeración implícita

A partir de aquí, una *solución factible* es un conjunto dominante global, mientras que cualquier subconjunto de vértices del conjunto V será denominado simplemente como una solución. Una solución factible con la cardinalidad mínima es una *solución óptima*.

El algoritmo de enumeración implícita descrito aquí es similar al algoritmo para el problema clásico de dominación propuesto en [69] y la Sección 2.5. Este algoritmo comienza con la mejor solución factible obtenida a través de una de las heurísticas H1, H2 y H3 (ver Secciones 3.5.1, 3.5.2 y 3.5.3). Usando esta solución, se determina la cota superior inicial U para el tamaño de una solución factible. Para cada instancia probada, la cota inferior L se obtiene ya sea utilizando la relajación de la formulación ILP del problema o directamente a partir del Corolario 3.1. Empleando estas cotas inferiores y superiores, la búsqueda se restringe a soluciones con un tamaño en el rango $[L, U]$ usando búsqueda binaria.

Un procedimiento diseñado especialmente enumera soluciones de cada tamaño $\nu \in [L, U]$ en un orden de prioridad. Las soluciones de tamaño ν son consideradas en este orden y se verifica su factibilidad. Los tamaños se derivan mediante búsqueda binaria realizada en el intervalo $[L, U]$. Este procedimiento y la lista de prioridad se describen en la subsección 2.5.1.

A continuación, se describe el marco general del algoritmo exacto. Nótese que es esencialmente el mismo que se propone en la Sección 2.5 y que puede utilizarse para cualquier variante del problema de dominación, ajustando la condición de factibilidad. Se denota por σ la solución actual de tamaño ν ; como ya se mencionó, σ se establece inicialmente como la mejor solución factible proporcionada por cualquiera de las heurísticas H1, H2 o H3. El algoritmo verifica las siguientes condiciones utilizando búsqueda binaria.

- Si la solución σ es un conjunto dominante global, entonces la cota superior actual U se actualiza a ν . El algoritmo continúa con el siguiente valor de prueba (menor) ν del intervalo $[L, \nu)$ obtenido mediante el procedimiento de búsqueda binaria. Si ya se han probado todos los ν de prueba, σ es una solución óptima, se devuelve y el algoritmo se detiene.
- Si la solución actual σ de tamaño ν no es un conjunto dominante global, entonces se llama a *Procedure_Next*(ν) para generar la siguiente solución σ del mismo tamaño ν .
- Si *Procedure_Next*(ν) devuelve *NULL*, todas las soluciones de tamaño ν ya fueron probadas, ninguna de ellas siendo factible, la cota inferior actual L se actualiza a ν . El algoritmo procede con el siguiente valor de prueba (más grande) ν del intervalo $[\nu, U)$.

derivado del procedimiento de búsqueda binaria. Si todos los valores de prueba ν ya fueron probados, σ es una solución óptima, se devuelve y el algoritmo se detiene.

Una descripción formal del algoritmo es la siguiente.

Algorithm 13 Algorithm_IE

Input: A graph G .
Output: A $\gamma_g(G)$ -set D .
 $L := \text{máx}\{\frac{n}{\Delta(G)+1}, \frac{2r}{3}, \frac{d+1}{3}, |Supp(G)|\};$
 $D := \sigma;$ {the best feasible solution delivered by the algorithms 14, 15, and 16}
 $U := |D|;$
 $\nu := \lfloor (L + 3U)/4 \rfloor;$
 $LIST := Procedure_Priority_LIST(G, D);$
{ iterative step }
while $U - L > 1$ **do**
 if $Procedure_Next(\nu, LIST, G)$ returns NIL **then**
 $L := \nu;$
 $\nu := \lfloor (L + 3U)/4 \rfloor;$
 else { A feasible solution was found ($\sigma_h(\nu)$) }
 $D := \sigma_h(\nu);$
 $U := \nu;$
 $\nu := \lfloor (L + 3U)/4 \rfloor;$
 $LIST := Procedure_Priority_LIST(G, D);$
 end if
end while
return $D;$

La diferencia entre este algoritmo y el propuesto en la Sección 2.5 para el problema del conjunto dominante radica en la condición de factibilidad y en el criterio de parada del procedimiento $Procedure_Next()$, que consisten en verificar que la solución generada sea un conjunto dominante global y no simplemente un conjunto dominante. Por lo tanto, todas las propiedades demostradas en la Sección 2.5 también se cumplen para este algoritmo.

3.5. Algoritmos Heurísticos

3.5.1. Heurística H1

La heurística H1 primero construye un conjunto dominante global auxiliar D en un número de iteraciones; se denota por D_h al conjunto de vértices ya construido en la iteración h . Dado el conjunto D_h se definen los siguientes conjuntos: A_h es el conjunto de todos los vértices de G no dominados en la iteración h , es decir, el conjunto de vértices de $V(G) \setminus D_h$ sin un vecino en

D_h , $N(A_h) \cap D_h = \emptyset$; B_h es el conjunto de vértices no dominados en $\overline{G}$, es decir, cada vértice dentro de este conjunto es adyacente a todos los vértices de D_h ; $C_h = V \setminus (D_h \cup A_h \cup B_h)$. Nótese que los conjuntos D_h , A_h , B_h y C_h son disjuntos. Se deduce que C_h es el conjunto de vértices *globalmente dominados* en la iteración h , es decir, el conjunto de vértices $x \in V \setminus D_h$ tal que existe $y \in D_h$ con $\{x, y\} \in E$ y $z \in D_h$ con $\{x, z\} \notin E$. El siguiente lema se sigue de esto.

Lema 3.3. *Los conjuntos de vértices A_h , B_h , C_h y D_h son disjuntos y $A_h \cup B_h \cup C_h \cup D_h = V$.*

El *grado activo global* en la iteración $h \geq 1$ del vértice $v \in V \setminus D_{h-1}$, $GAD_h(v)$, es el número de vecinos de v en A_{h-1} más el número de no vecinos de v en B_{h-1} , es decir,

$$GAD_h(v) = |N_{A_{h-1}}[v] \cup \overline{N}_{B_{h-1}}[v]|. \quad (3.6)$$

Entonces, el $GAD_h(v)$ es el número de vértices que aún no están dominados en la iteración h , y que serían dominados por el vértice v tanto en G como en $\overline{G}$.

La heurística H1 comienza en la iteración 0 determinando dos vértices u y v con

$$\{u, v\} = \operatorname{argmax}_{x, y \in V} |(N[x] \cup N[y]) \setminus (N[x] \cap N[y])| \quad (3.7)$$

Note que $\gamma_g(G) \geq 2$. En otras palabras, comienza con un par de vértices que dominan globalmente el mayor número posible de vértices. Así que, se establece $D_0 = \{u, v\}$. Entonces, $C_0 = (N[u] \cup N[v]) \setminus (N[u] \cap N[v])$, $A_0 = V \setminus (N[u] \cup N[v])$ y $B_0 = V \setminus (D_0 \cup C_0 \cup A_0)$.

Lema 3.4. *Si $A_0 = B_0 = \emptyset$ entonces*

$$(N[u] \cup N[v]) \setminus (N[u] \cap N[v]) = V \setminus \{u, v\}, \quad (3.8)$$

y $D_0 = \{u, v\}$ es una solución óptima.

Demostración. La Ecuación (3.8) se sigue inmediatamente de la condición $A_0 = B_0 = \emptyset$ y del hecho de que $|D_0| = 2$. Según la Ecuación (3.8), el conjunto de vértices $V \setminus \{u, v\}$ está particionado en dos conjuntos (disjuntos) $N[u]$ y $N[v]$. Esto implica que D_0 es un conjunto dominante global. Alternativamente, $A_0 = B_0 = \emptyset$ da como resultado $C_0 = V \setminus D_0$ según el Lema 3.3. Entonces, D_0 es mínimo, ya que un conjunto dominante global no puede contener menos de 2 vértices. ■

Si la condición en el Lema 3.4 no se cumple, el algoritmo continúa con la iteración 1. Se detiene en la iteración $h \geq 1$ si $A_{h-1} = B_{h-1} = \emptyset$. De lo contrario, determina el vértice $v_h \in V \setminus D_{h-1}$ con el mayor grado activo global y establece $D_h := D_{h-1} \cup v_h$. Los conjuntos A_h , B_h y C_h se actualizan correspondientemente, como se muestra a continuación en el pseudo-código.

Algorithm 14 H1

```

Input: A graph  $G(V, E)$ .
Output: A global dominating set  $D_h$ .
 $h := 0$ ;
 $\{u, v\} := \operatorname{argmax}_{x, y \in V} |(N[x] \cup N[y]) \setminus (N[x] \cap N[y])|$ ;
 $D_0 := \{u, v\}$ ;
 $C_0 := (N[u] \cup N[v]) \setminus (N[u] \cap N[v])$ ;
 $A_0 := V \setminus (N[u] \cup N[v])$ ;
 $B_0 := V \setminus (D_0 \cup C_0 \cup A_0)$ ;
    { iterative step }
while  $A_h \neq \emptyset \vee B_h \neq \emptyset$  do
     $h := h + 1$ ;
     $v_h :=$  a vertex in the set  $V \setminus D_{h-1}$  with the maximum  $GAD_h(v_h)$ ;
     $D_h := D_{h-1} \cup \{v_h\}$ ;
     $C_h := C_{h-1} \cup N_{A_{h-1}}(v_h) \cup \overline{N}_{B_{h-1}}(v_h)$ ;
    if  $A_{h-1} \neq \emptyset$  then
         $A_h := A_{h-1} \setminus N_{A_{h-1}}(v_h)$ ;
    end if
    if  $B_{h-1} \neq \emptyset$  then
         $B_h := B_{h-1} \setminus \overline{N}_{B_{h-1}}(v_h)$ ;
    end if
end while

```

Proposición 3.1. *La heurística H1 encuentra una solución factible óptima si se detiene en la iteración h con $|D_h| \leq 3$, $h = 0, 1$.*

Demostración. Se debe demostrar que $|D_h| = \gamma_g(G)$ siempre que $|D_h| \leq 3$. Primero, observe que $|D_h|$ no puede ser 1. Supongamos que $|D_h| = 2$, es decir, $h = 0$ y el conjunto D_0 contiene dos vértices, digamos u y v . Dado que el algoritmo se detuvo con el conjunto D_0 , debe cumplirse que $A_0 = B_0 = \emptyset$, lo que implica la ecuación (3.8) y que D_0 es un conjunto dominante global por el Lema 3.4, y la proposición se cumple para $h = 0$. Si ahora $|D_h| = 3$, $h = 1$, entonces no existe ningún par de vértices $\{u, v\}$ que satisfaga la ecuación 3.8 (por la construcción). Por lo tanto, $\gamma_g(G) \geq 3$. Al mismo tiempo, dado que el algoritmo se ha detenido en la iteración 1, $A_1 = B_1 = \emptyset$, y por el Lema 3.3, $C_1 = V \setminus D_1$. Por lo tanto, D_1 es una solución factible y la proposición queda demostrada. ■

Ahora se define una familia particular de grafos. Un grafo conexo $G = (V, E)$ de esta familia, de orden n , está compuesto por dos subgrafos H_1 y H_2 de orden $m_i + 1$, cada uno de los cuales contiene el grafo estrella S_{m_i} como subgrafo, con $V(H_1) \cup V(H_2) = V(G)$ y $V(H_1) \cap V(H_2) = \emptyset$, y con centro v_i ($i = 1, 2$), un vértice con grado máximo m_i en el grafo S_{m_i} . En la Figura 3.4 se ilustra dicho grafo, donde cada línea discontinua representa una arista que puede existir o no. Nótese que cada H_i también puede ser un grafo completo.

Teorema 3.8. *La heurística H1 encuentra una solución óptima para cualquier grafo G de la familia especificada anteriormente.*

Demostración. Dado el grafo G , en la primera iteración, la heurística H1 seleccionará los dos vértices v_1, v_2 (marcados en verde en la Figura 3.4), tales que $|(N[v_1] \cup N[v_2]) \setminus (N[v_1] \cap N[v_2])|$ sea máximo. Observemos que $(N[v_1] \cup N[v_2]) \setminus (N[v_1] \cap N[v_2]) = V \setminus \{v_1, v_2\}$ y, por lo tanto, $A_0 = B_0 = \emptyset$. Por lo tanto, la heurística H1 devolverá un conjunto dominante global $D = \{v_1, v_2\}$ en la iteración 0, el cual es óptimo según el Lema 3.4. ■

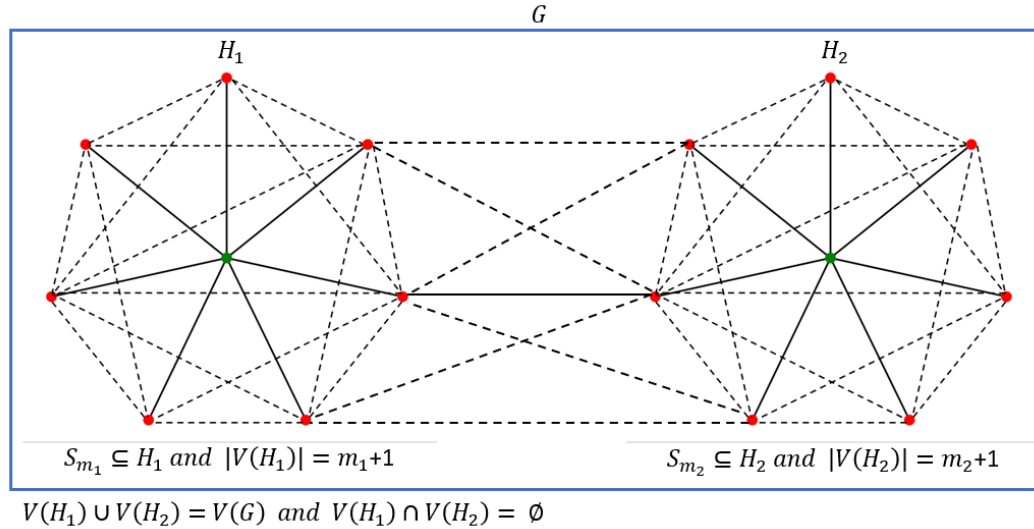


Figura 3.4: Grafo G

La Figura 3.5 muestra un grafo con $|D| = 3$. En la iteración $h = 0$, $N[u] \cup N[v] \setminus (N[u] \cap N[v]) = \{v, u, 1, 3, 4, 5, 6\}$. En consecuencia, H1 elige $D_0 = \{u, v\}$. Luego, $C_0 = \{1, 3, 4, 5, 6\}$, $A_0 = \{v_1, 2\}$ y $B_0 = \emptyset$. En la iteración $h = 1$, se puede elegir ya sea el vértice v_1 o el vértice 2. Entonces, $D_1 = \{u, v, v_1\}$ o $D_1 = \{u, v, 2\}$, ambos siendo un conjunto dominante global de cardinalidad mínima 3.

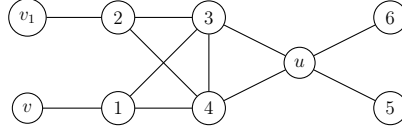


Figura 3.5: Grafo con $\gamma_g(G) = 3$

Proposición 3.2. *La heurística H1 se ejecuta en un máximo de $|A_0| + |B_0|$ iteraciones y su complejidad temporal es $O(n^3)$.*

Demostración. Primero, recordemos que el número total de iteraciones del bucle externo en la heurística H1 está acotado por $|A_0| + |B_0|$. Observemos que, en la iteración $h = 0$, para cada par de vértices $\{x, y\}$ en $V(G)$ se calcula $|(N[x] \cup N[y]) \setminus (N[x] \cap N[y])|$ sumando todas las entradas 1 en las filas correspondientes a los vértices x, y , siempre que ambas entradas no sean simultáneamente 1. Dado que cada fila contiene n entradas y hay n^2 pares de vértices que verificar, el cálculo de $|(N[x] \cup N[y]) \setminus (N[x] \cap N[y])|$ para todos los pares de vértices $\{x, y\}$ en $V(G)$ y la selección del máximo toma un tiempo $O(n^3)$.

Por el Lema 3.3, se tiene que $|A_0| + |B_0| \leq n$. Entonces, el número total de iteraciones del bucle externo está acotado por n .

En cada iteración $h \geq 1$, para cada vértice en $V \setminus D_{h-1}$, se calcula el grado activo global contando todas las entradas 1 y 0 en la fila correspondiente a ese vértice, donde los vértices correspondientes a entradas 1 que pertenezcan a A_h y aquellos correspondientes a entradas 0 que pertenezcan a B_h . Dado que cada fila contiene n entradas y hay menos de n filas para verificar, el cálculo del grado activo global de todos los elementos en $V \setminus D_{h-1}$ y la selección del máximo toma un tiempo $O(n^2)$.

Por lo tanto, la heurística H1 encuentra un conjunto dominante global D en un tiempo $O(n^3)$.

■

3.5.2. Heurística H2

La heurística H2, en cada iteración, complementa el conjunto de vértices actual D_h con dos vértices, v_h y u_h , definidos de la siguiente manera: v_h es un vértice que domina el máximo número de vértices aún no dominados en el grafo G , es decir, tiene el máximo número de vértices adyacentes entre aquellos que aún no tienen un vecino en el conjunto D_h . De manera similar, el vértice u_h es aquel que domina el máximo número de vértices aún no dominados en

el grafo $\overline{G}$, es decir, tiene el máximo número de vértices no adyacentes entre aquellos que no son adyacentes a un vértice en el conjunto D_h . En otras palabras, $v_h \in V \setminus D_h$ es un vértice con

$$|N_{A_h}[v_h]| = \max_{v \in V \setminus D_h} |N_{A_h}[v]|.$$

De manera similar, $u_h \in V \setminus D_h$ es un vértice con

$$|\overline{N}_{B_h}[u_h]| = \max_{v \in V \setminus D_h} |\overline{N}_{B_h}[v]|.$$

A continuación se presenta una descripción detallada del algoritmo.

Algorithm 15 H2

Input: A graph $G(V, E)$.
Output: A global dominating set D_h .
 $D_0 := \emptyset$;
 $C_0 := \emptyset$;
 $A_0 := V(G)$;
 $B_0 := V(G)$;
{ iterative step }
 $h := 0$;
while $A_h \neq \emptyset \vee B_h \neq \emptyset$ **do**
 $h := h + 1$;
 $v_h := u_h := NIL$;
 if D_{h-1} is not a dominant set of G **then**
 $v_h :=$ a vertex in set $V \setminus D_{h-1}$ such that $N_{A_h}[v]$ be maximum;
 $A_h := A_{h-1} \setminus N_{A_{h-1}}(v_h)$;
 $B_h := B_{h-1} \setminus \overline{N}_{B_{h-1}}(v_h)$;
 end if
 $D_h := D_{h-1} \cup \{v_h\}$;
 if D_h is not a dominant set of $\overline{G}$ **then**
 $u_h :=$ a vertex in set $V \setminus D_h$ such that $\overline{N}_{B_h}[u]$ be maximum;
 $D_h := D_h \cup \{u_h\}$;
 $A_h := A_h \setminus N_{A_h}(u_h)$;
 $B_h := B_h \setminus \overline{N}_{B_h}(u_h)$;
 end if
 $C_h := N(D_h) \cap \overline{N}(D_h)$;
end while

Proposición 3.3. *La heurística H2 se ejecuta en un máximo de $|A_0| + |B_0|$ iteraciones y su complejidad temporal es $O(n^3)$.*

Demostración. Primero, recordemos que el número total de iteraciones del bucle externo en la heurística H2 está acotado por $|A_0| + |B_0|$. Por el Lema 3.3, se tiene que $|A_0| + |B_0| \leq n$. En cada iteración h , para cada vértice v en $V \setminus D_{h-1}$, se calcula $|N_{A_h}[v]|$ contando todas las entradas 1 en la fila correspondiente a ese vértice, de manera que las entradas 1 pertenezcan

a A_h . Para cada v rtice u en $V \setminus D_{h-1}$, se calcula $|\overline{N}B_h[u]|$ contando todas las entradas 0 en la fila correspondiente a ese v rtice, de manera que las entradas 0 pertenezcan a B_h . Dado que cada fila contiene n entradas y hay menos de n filas por verificar, el c lculo de $|NA_h[v]|$ y $|\overline{N}B_h[u]|$ para todos los elementos en $V \setminus D_{h-1}$ y la selecci n del m ximo toma un tiempo de $O(n^2)$. Por lo tanto, la heur stica H2 encuentra un conjunto dominante global D en un tiempo $O(n^3)$. ■

En la Figura 3.6 (a) y (b), se dan ejemplos de cuando la heur stica H2 obtiene una soluci n  ptima y no  ptima, respectivamente. En (a) se muestra el grafo Petersen, donde $\gamma_g(G) = 4$ y H2 entrega una soluci n  ptima $\{1, 2, 3, 4\}$. En (b) se muestra un grafo G donde el conjunto de v rtices negros representan una soluci n  ptima, y $\{v_1, v_2, v_3, v_4\}$ es el conjunto dominante global obtenido por H2.

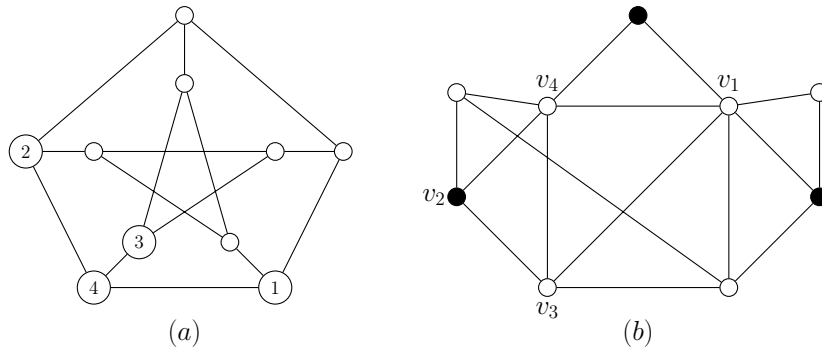


Figura 3.6: (a) Grafo Petersen; (b) grafo G

Se puede observar f cilmente que las cotas superiores U_2 y U_1 introducidas en la Secci n 3.2 (Teoremas 3.3 y 3.4) se alcanzan, por ejemplo, para el grafo completo K_n y el grafo de Petersen representado en la Figura 3.6(a), respectivamente. Al mismo tiempo, para ambos tipos de grafos, la heur stica H2 devuelve una soluci n  ptima (ver la Figura 3.6(a), que ilustra el grafo de Petersen y la soluci n  ptima generada por el algoritmo para ese grafo).

A continuaci n, se define una familia m s amplia de grafos para los cuales la heur stica encuentra el  ptimo. Sea G con $|V(G)| = n$ y H_i ($i = 1, 2, \dots, n$) grafos conectados, todos de orden no menor que dos, y sea $G' = G \circ_{v_i} H_i$ el grafo compuesto por estos $n + 1$ grafos, en el cual cada v rtice del grafo G est  asociado con un v rtice especialmente determinado v_i del grafo H_i (el v rtice v_i se elige de un $\gamma(H_i)$ -conjunto). Las Figuras 3.7 y 3.8 ilustran un

ejemplo de tales grafos compuestos, donde cada H_i es el grafo de Petersen, y G es un grafo conexo. En la Figura 3.9 se ilustra otro grafo de este tipo, donde los H_i s son diferentes.

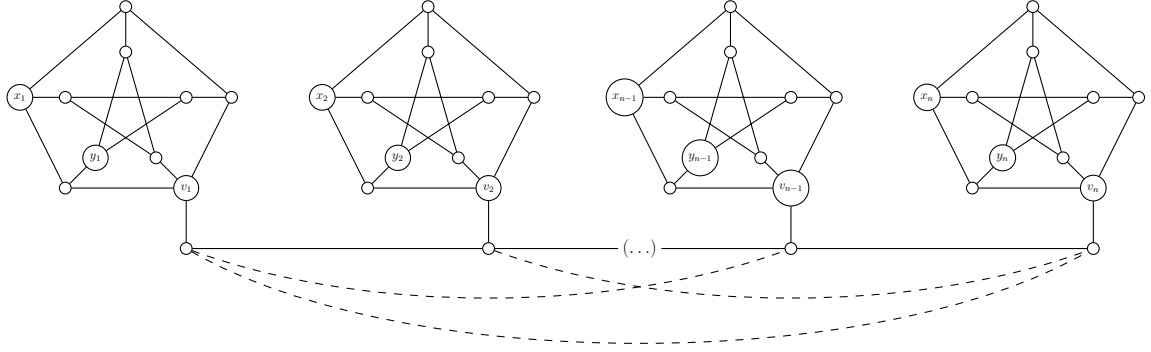


Figura 3.7: Grafo $G' = G \circ_v H$, donde H es el grafo de Petersen.

Proposición 3.4. Dado el grafo $G' = G \circ_{v_i} H_i$ con $|V(G)| = n \geq 2$ y $v_i \in \gamma(H_i)$ -conjunto, el $\gamma_g(G')$ -conjunto es $\cup_{i=1}^n \gamma(H_i)$ -conjunto.

Demostración. Como es fácil de ver, $n \geq 2$ implica que cualquier par de vértices de tipo v_i forman un conjunto dominante del grafo $\overline{G'}$, mientras que $v_i | i = \overline{1, n}$ forma un conjunto dominante del grafo G . Por lo tanto, el conjunto $\bigcup_{i=1}^n \gamma(H_i)$ es un conjunto dominante global de G' . ■

Por ejemplo, si los H_i son grafos de Petersen, entonces, según la proposición anterior, $\gamma_g(G') = 3n$ (ver Figura 3.7). Si, además, G es un grafo P_n o C_n , entonces la heurística H2 encontrará una solución óptima. Para el grafo G' con $G = P_3$, $\gamma(G') = 3\gamma(H_i) = 9$, ver Figura 3.8. Nótese que como el $\gamma_g(G')$ -conjunto y el $\gamma(G')$ -conjunto son idénticos, se alcanza la cota inferior del Teorema 3.2.

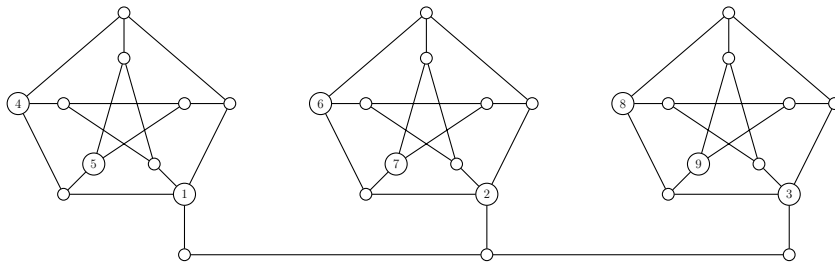


Figura 3.8: Grafo $G' = P_3 \circ_v H$, donde H es el grafo de Petersen

El siguiente teorema trata sobre una familia más general de grafos:

Teorema 3.9. Sea H_i , $i = \overline{1, n}$, un grafo conexo de orden $m_i + 1$ que contiene el grafo estrellado S_{m_i} como subgrafo, tal que $m_1 \geq m_2 \geq \dots \geq m_i \geq \dots \geq m_{n-1} \geq m_n > \Delta(G)$. Sea, además, v_i el centro del grafo S_{m_i} (un vértice con grado máximo m_i en ese grafo), y sea G un grafo conexo de orden n . Entonces, la heurística H2 encuentra una solución óptima para el grafo $G' = G \circ_{v_i} H_i$.

Demostración. Claramente, v_i es un conjunto dominante del grafo H_i y el $\gamma_g(G')$ -conjunto es $\cup_{i=1}^n v_i$ (ver la Proposición 3.4 y la Figura 3.9). Dado que $\Delta(G)$ es menor que todos los m_i , la heurística H2 en la primera iteración seleccionará los dos vértices de tipo v_i con los grados más altos, es decir, $\{v_1, v_2\}$, que forman un conjunto dominante del grafo $\overline{G'}$. En cada iteración siguiente $h = 2, \dots, n-1$, la heurística H2 añade el vértice v_{i+1} a la solución actual y devuelve un conjunto dominante global $D = \{v_i | i = \overline{1, n}\}$ en la iteración $n-1$, que es óptimo según la Proposición 3.4. ■

La Figura 3.9 muestra un ejemplo de grafo $G' = G \circ_{v_i} H_i$, donde el número de dominación global es $n = |V(G)|$ y la heurística H2 obtiene una solución óptima eligiendo los vértices verdes.

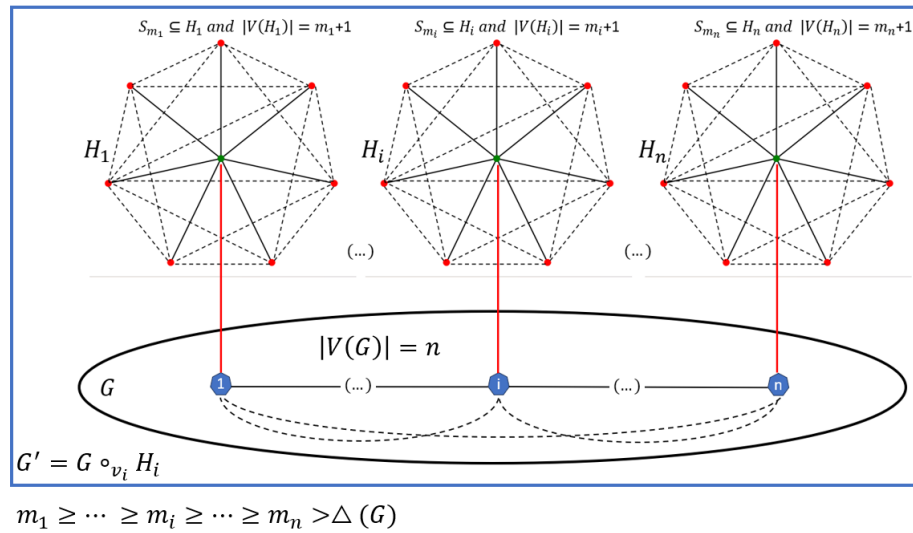


Figura 3.9: Grafo $G' = G \circ_{v_i} H_i$

3.5.3. Heurística H3

En la heurística H3, los conjuntos de vértices D_h , A_h , B_h y C_h , y la iteración inicial 0 con los vértices u y v se definen como en la heurística H1. Las partes iterativas de las heurísticas

H1 y H2 se combinan en esta heurística. Si se cumple la condición $A_{h-1} = B_{h-1} = \emptyset$ (ver Observación 3.4), entonces el algoritmo se detiene en la iteración $h \geq 1$. De lo contrario, determina los vértices v_h, v'_h, u_h :

- v_h es un vértice en $V \setminus D_{h-1}$ con el máximo grado activo global;
- $v'_h \in V \setminus D_{h-1}$ es un vértice con $N_{V \setminus N[D_{h-1}]}[v'_h] = \max_{v \in V \setminus D_{h-1}} N_{V \setminus N[D_{h-1}]}[v]$;
- $u_h \in V \setminus D_{h-1}$ es un vértice con $\overline{N}_{V \setminus \overline{N}[D_{h-1}]}[u_h] = \max_{v \in V \setminus D_{h-1}} \overline{N}_{V \setminus \overline{N}[D_{h-1}]}[v]$.

Sea

$$GAD(v'_h, u_h) = |N_{A_{h-1}}[\{v'_h, u_h\}] \cup \overline{N}_{B_{h-1}}[\{v'_h, u_h\}]|.$$

Si $GAD(v_h) \geq GAD(v'_h, u_h)/2$, entonces $D_h := D_{h-1} \cup \{v_h\}$; de lo contrario, $D_h := D_{h-1} \cup \{v'_h, u_h\}$. Si $A_{h-1} = \emptyset$, el vértice v'_h no se requiere, y si $B_{h-1} = \emptyset$, el vértice u_h tampoco es requerido. Los conjuntos A_h, B_h y C_h se actualizan correspondientemente. A continuación se proporciona una descripción detallada de la heurística.

Algorithm 16 H3

Input: A graph $G(V, E)$.
Output: A global dominating set D_h .
 $h := 0$;
 $\{u, v\} :=$ two vertex with the largest possible cardinality of the set
 $(N[u] \cup N[v]) \setminus (N[u] \cap N[v])$;
 $D_0 := \{u, v\}$;
 $C_0 := (N[u] \cup N[v]) \setminus (N[u] \cap N[v])$;
 $A_0 := V \setminus (N[u] \cup N[v])$;
 $B_0 := V \setminus (D_0 \cup C_0 \cup A_0)$;
{ iterative step }
while $A_h \neq \emptyset \vee B_h \neq \emptyset$ **do**
 $h := h + 1$;
 $v_h :=$ a vertex in the set $V \setminus D_{h-1}$ with the maximum $GAD(v_h)$;
 if $A_h \neq \emptyset$ **then**
 $v'_h :=$ a vertex in the set $V \setminus D_{h-1}$ such that $N_{V \setminus N[D_{h-1}]}[v'_h]$ be maximum;
 end if
 if $B_h \neq \emptyset$ **then**
 $u_h :=$ a vertex in the set $V \setminus D_{h-1}$ such that $\overline{N}_{V \setminus \overline{N}[D_{h-1}]}[u_h]$ be maximum;
 end if
 if $GAD(v_h) \geq GAD(v'_h, u_h)/2$ **then**
 $D_h := D_{h-1} \cup \{v_h\}$;
 $C_h := C_{h-1} \cup N_{A_{h-1}}(v_h) \cup \overline{N}_{B_{h-1}}(v_h)$;
 $A_h := A_{h-1} \setminus N_{A_{h-1}}(v_h)$;
 $B_h := B_{h-1} \setminus \overline{N}_{B_{h-1}}(v_h)$;
 else
 $D_h := D_{h-1} \cup \{v'_h, u_h\}$;
 $C_h := C_{h-1} \cup N_{A_{h-1}}(v'_h, u_h) \cup \overline{N}_{B_{h-1}}(v'_h, u_h)$;
 $A_h := A_{h-1} \setminus N_{A_{h-1}}(v'_h, u_h)$;
 $B_h := B_{h-1} \setminus \overline{N}_{B_{h-1}}(v'_h, u_h)$;
 end if
end while

La siguiente proposición se deduce fácilmente y se puede probar de manera similar a las Proposiciones 3.2 y 3.3.

Proposición 3.5. *La heurística H3 se ejecuta en un máximo de $|A_0| + |B_0|$ iteraciones y su complejidad temporal es $O(n^3)$.*

3.5.4. Procedimiento de purificación

Los procedimientos de purificación han demostrado ser útiles para la reducción del conjunto de vértices en una solución factible no óptima para el problema clásico de dominación. En esta sección se aplica un tipo similar de purificación para el problema de dominación global.

Sea $D = \{v_1, \dots, v_p\}$ un conjunto dominante global generado por H1, H2 o H3, donde los vértices se incluyeron en este orden particular en dicho conjunto. El conjunto purificado inicial D^* se forma de la siguiente manera. Inicialmente, $D^* := D$; de manera iterativa, para $i = p, \dots, 1$: si $pn(v_i, D^*, G) = \emptyset$ y $pn(v_i, D^*, \overline{G}) = \emptyset$, entonces $D^* := D^* \setminus v_i$, es decir, se purifica el vértice v_i . Ver Algoritmo 17.

Algorithm 17 Procedure Purify_GDS

Input: A graph $G(V, E)$ and a global dominating set S .

Output: A global dominating set S^* .

$S^* := \{v_1, \dots, v_p\}$; {vertices of S in the order they were added by H1, H2 or H3.}

$h := p$;

{ iterative step }

while $h > 0$ **do**

if $pn(v_i, D^*, G) = \emptyset \wedge pn(v_i, D^*, \overline{G}) = \emptyset$ **then**

$S^* := S^* \setminus \{v_h\}$; {we purify vertex v_h }

end if

$h := h - 1$;

end while

El conjunto D^* así formado, es un conjunto dominante global minimal del grafo G . De hecho, por la construcción del procedimiento, la eliminación de cualquier vértice en el conjunto D^* entregado por el procedimiento de purificación, daría como resultado una solución no factible. Es fácil ver que este procedimiento se puede implementar en un tiempo de $O(pn^2) = O(n^3)$ como el descrito en la subsección 2.3.4.

3.6. Resultados experimentales

En esta sección se describen los experimentos computacionales para el problema de dominación global. Se utilizó el solver *intlinprog* de MATLAB y *CPLEX* de IBM ILOG para resolver el modelo ILP. Se implementaron los algoritmos en C++. Los algoritmos fueron probados con 2284 instancias de referencia disponibles en [91] con hasta 14000 vértices, que fueron generadas para el problema clásico de dominación. También se crearon 6 instancias adicionales de gran tamaño con hasta 25000 vértices para probar las heurísticas. Los códigos de las tres heurísticas y la formulación ILP, así como todos los resultados experimentales, están disponibles en [91].

Se crearon las instancias de los problemas empleando varias técnicas pseudoaleatorias para la generación de grafos. El orden (número de vértices) y el tamaño (número de aristas) de cada instancia se determinaron aleatoriamente utilizando la función *random()* dentro de los intervalos $[50, 25000]$ y $[180, 3 * 10^8]$, respectivamente. Para cada arista recién creada, se seleccionó aleatoriamente el par correspondiente de vértices actualmente no adyacentes hasta alcanzar la cantidad deseada de aristas. Para un orden dado, el tamaño se determinó según la densidad deseada. Se emplearon instancias con densidades desde 0.1 hasta 0.9, pero se concentró el estudio en grafos de densidad media. De hecho, los grafos dispersos tienden a tener una estructura muy simple, lo que hace que los problemas de optimización correspondientes sean más fáciles. Al mismo tiempo, los algoritmos simples funcionan bien para grafos casi completos debido a la uniformidad de las conexiones entre los vértices.

Dado que no existía anteriormente ningún algoritmo para el problema de dominación global estudiado, se utilizaron los valores óptimos obtenidos y las cotas superiores e inferiores disponibles para analizar el rendimiento de las heurísticas. En esta sección, ILP^{L1} e ILP^{L2} representan las cotas inferiores obtenidas por CPLEX e *intlinprog*, respectivamente, para el modelo ILP dentro del límite de tiempo de 2 minutos. Para todas las instancias en las Tablas 3.10-3.12, la mejor solución fue obtenida por la heurística H2. Por lo tanto, los valores en la columna etiquetada como U corresponden a esta heurística. En todas las tablas, los mejores resultados para cada instancia, ya sea en tiempo o en la cardinalidad de la mejor solución

factible, o en los casos donde el límite U resultó ser óptimo, están resaltados en negrita.

3.6.1. Rendimiento de los algoritmos heurísticos

En la Figura 3.10 se compara la calidad de la mejor solución generada por los algoritmos heurísticos frente a los valores óptimos obtenidos por los algoritmos exactos para instancias de hasta 8000 vértices. En esta figura, los nodos azules representan soluciones óptimas, y los cuadrados rojos representan la mejor solución obtenida por las heurísticas. Como se puede ver, las heurísticas encontraron una solución óptima para el 56,75 % de las instancias, mientras que, entre las instancias donde no se encontró el óptimo, el error absoluto promedio fue de aproximadamente 1 vértice.

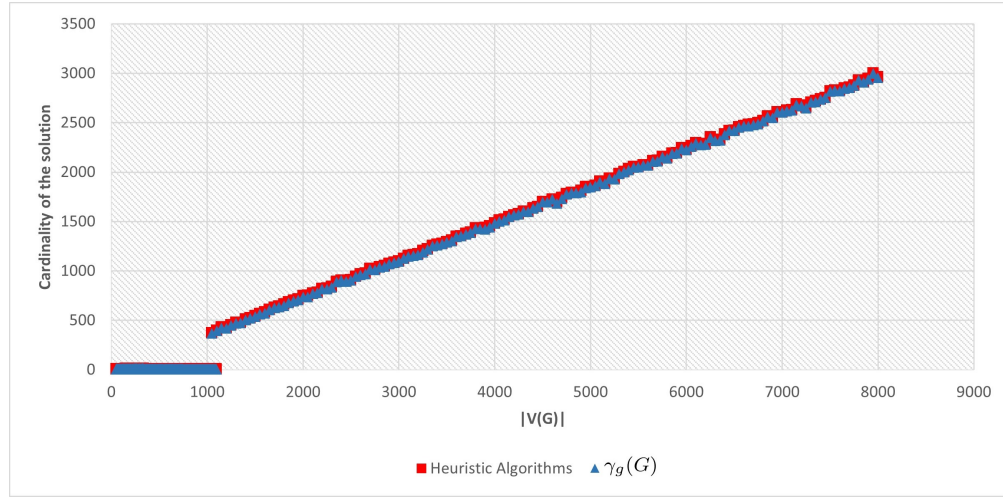
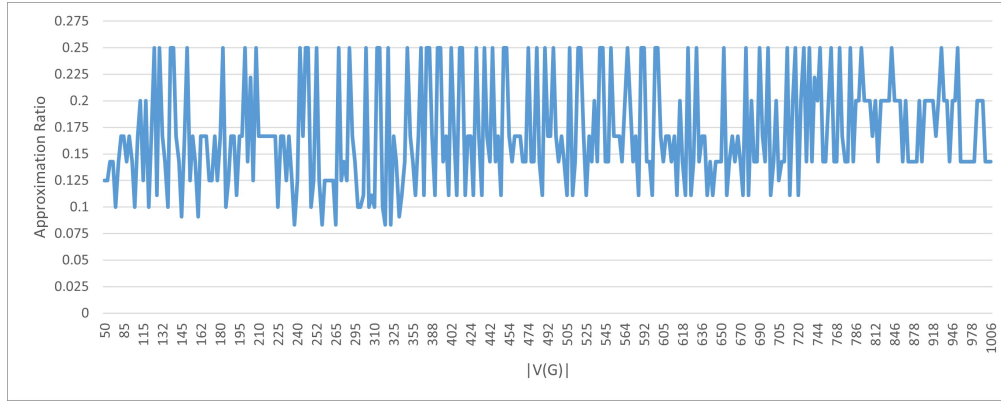
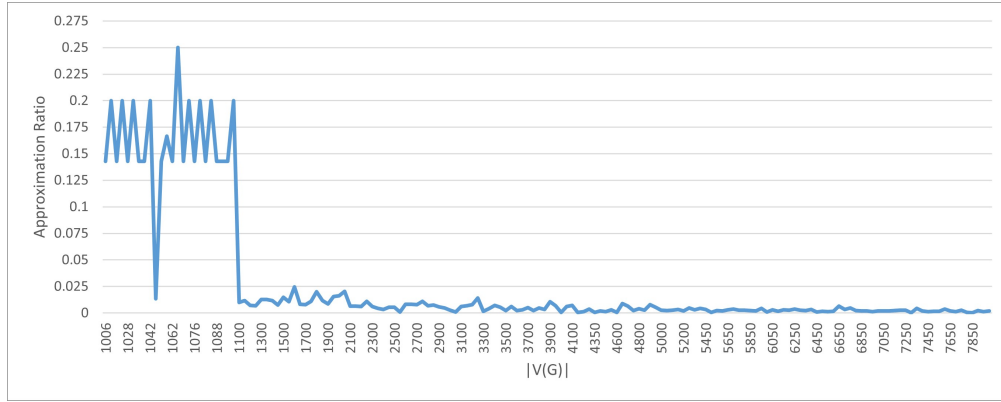


Figura 3.10: $\gamma_g(G)$ vs. mejor resultado de las heurísticas

Las Figuras 3.11 (a) y (b) muestran los radios de aproximación obtenidos para las instancias en las que las heurísticas no lograron encontrar una solución óptima. Se observa claramente un comportamiento asintótico favorable de las heurísticas para instancias de gran escala.



(a) Instancias con menos de 1006 vértices



(b) Instancias con más de 1006 vértices

Figura 3.11: Comportamiento asintótico de los radios de aproximación

Los resultados de las heurísticas reportados anteriormente incluyen la etapa de purificación. El proceso de purificación resultó ser eficiente en la práctica, como se ilustra en las Tablas 3.1 y 3.6, donde también se muestran los valores de las cotas inferiores y superiores correspondientes L y U . $|D|$ y $|D^*|$ representan, respectivamente, el número de vértices en el conjunto dominante global obtenido antes y después de la purificación.

V	E	L	ILP ^{L1}	H1			Purificación			H2			Purificación			U
				Tiempo(s)	D _{H1}	Tiempo(s)	D _{H1}	%	Tiempo(s)	D _{H2}	Time(s)	D _{H2}	%			
7300	7311	2663	2705	2794.67	3344	1.62	2706	19.08	567.73	3161	0.47	2706	14.39	6843		
7350	7474	2621	2710	2839.76	3365	1.68	2726	18.99	578.91	3177	0.50	2722	14.32	6890		
7400	7474	2677	2735	2949.72	3392	1.69	2743	19.13	591.42	3202	0.51	2740	14.43	6937		
7500	7535	2762	2817	3077.71	3427	1.67	2822	17.65	615.30	3270	0.53	2822	13.70	7031		
7600	7734	2729	2819	3223.23	3425	1.69	2831	17.34	642.02	3277	0.49	2830	13.64	7125		
7650	7696	2797	2845	3269.30	3490	1.73	2855	18.19	658.27	3336	0.53	2851	14.54	7171		
7700	7716	2803	2856	3325.65	3521	1.78	2862	18.72	703.98	3334	0.53	2860	14.22	7218		
7800	7804	2879	2930	3428.42	3571	1.81	2932	17.89	704.58	3397	0.54	2932	13.69	7312		
7850	7884	2851	2909	3499.23	3556	1.77	2913	18.08	705.05	3363	0.53	2910	13.47	7359		
7950	7982	2949	3000	6376.02	3641	2.03	3004	17.50	753.24	3481	0.55	3004	13.70	7453		
8000	8126	2882	2958	6828.32	3624	2.11	2966	18.16	758.84	3469	0.60	2964	14.56	7500		
9150	10443603	3	--	6762.49	21	0.47	20	4.76	17.71	22	0.43	19	13.64	6485		
9250	10673366	3	--	7020.20	20	0.46	19	5.00	17.71	21	0.44	19	9.52	6523		
9300	14357729	3	--	8222.71	20	0.54	18	10.00	21.33	21	0.53	17	19.05	5703		
9350	14540834	2	--	8380.73	16	0.57	15	6.25	18.87	18	0.59	15	16.67	5700		
9450	14825554	3	--	8755.81	19	0.53	17	10.53	20.95	20	0.68	16	20.00	5742		
9500	14983162	3	--	8857.65	20	0.57	18	10.00	25.09	20	0.57	16	20.00	5791		
9550	15141604	3	--	8974.30	20	0.61	18	10.00	26.14	24	0.66	18	25.00	5773		
9600	15329626	2	--	9114.64	16	0.61	15	6.25	19.76	18	0.64	16	11.11	5800		
9650	15460987	3	--	10186.50	20	0.59	18	10.00	22.17	20	0.64	17	15.00	5790		
9700	15621929	3	--	10488.80	19	0.57	17	10.53	22.48	20	0.67	16	20.00	5832		
9800	11959739	3	--	12332.80	23	0.53	21	8.70	26.78	28	0.58	21	25.00	6805		
12600	39630183	2	--	30335.40	12	1.29	11	8.33	37.34	12	1.45	11	8.33	5406		
12650	39945571	2	--	30143.80	13	1.34	12	7.69	44.06	14	1.64	11	21.43	5419		
12700	40262208	2	--	35540.20	13	1.73	12	7.69	43.48	14	1.50	11	21.43	5436		
12750	40580096	2	--	40648.00	13	1.38	12	7.69	44.04	14	1.51	11	21.43	5437		
12800	40841707	3	--	36502.20	15	1.49	13	13.33	53.28	16	1.61	13	18.75	5438		
12850	41219621	2	--	37844.20	13	1.39	12	7.69	41.89	13	1.46	12	7.69	5450		
12900	41541258	2	--	39178.50	12	1.37	11	8.33	38.03	12	1.39	11	8.33	5448		
12950	41864146	2	--	41767.80	12	1.68	11	8.33	37.53	12	1.34	11	8.33	5448		
13000	42188283	2	--	43198.60	12	1.40	11	8.33	41.69	13	1.45	10	23.08	5433		
13100	42840308	2	--	42710.00	12	1.44	11	8.33	42.70	13	1.45	10	23.08	5446		
13150	43168196	2	--	40698.00	12	1.46	11	8.33	41.15	12	1.35	11	8.33	5442		
25000	249810756	3	--	--	--	--	--	--	69.02	8	4.20	7	12.5	8353		

Tabla 3.1: Resultados de las heurísticas H1 y H2

En la Tabla 3.2 se compara la calidad de las soluciones obtenidas por las heurísticas H1 y H2 antes de la etapa de purificación, representadas por los parámetros $|D_{H1}|$ y $|D_{H2}|$, respectivamente.

	Instancias	%
$ D_{H1} = D_{H2} $	781	34.19
$ D_{H1} > D_{H2} $	365	15.98
$ D_{H1} < D_{H2} $	1138	49.83

Tabla 3.2: Análisis comparativo de las heurísticas H1 y H2 antes de la purificación

Las tablas 3.3, 3.5 y 3.8 a continuación, presentan un análisis comparativo de las heurísticas después de la purificación. Como se ve en la Tabla 3.1, en el 100 % de los casos, las soluciones entregadas por la heurística H2 fueron purificadas. La heurística H1 entregó conjuntos dominantes globales minimales para el 7.48 % de los casos (recordemos que un conjunto dominante minimal no puede ser purificado). Ambos algoritmos finalizaron en unos pocos segundos para las instancias con hasta 1000 vértices. La heurística H2 resultó ser más rápida, entregando soluciones en unos pocos segundos para la mayoría de las instancias, mientras que para la instancia más grande, con 14 100 vértices, se detuvo en 50 segundos. La heurística H1 tardó aproximadamente 11 horas para esta instancia. En promedio, entre las instancias resueltas por ambos algoritmos, la heurística H2 fue aproximadamente 88 veces más rápida.

	Instancias	%
$ D_{H1}^* = D_{H2}^* $	1286	56.31
$ D_{H1}^* > D_{H2}^* $	905	39.62
$ D_{H1}^* < D_{H2}^* $	93	4.07

Tabla 3.3: Análisis comparativo de las heurísticas H1 y H2 después de la purificación

La drástica diferencia observada en los tiempos de ejecución de las heurísticas H1 y H2 puede reducirse simplemente omitiendo la iteración inicial $h = 0$ de la heurística H1, que es la parte más costosa en tiempo de ese algoritmo. De hecho, en esa iteración, para encontrar dos vértices con la mayor cardinalidad posible del conjunto $(N[u] \cup N[v]) \setminus (N[u] \cap N[v])$, siempre se verificarán todos los pares de vértices en el grafo. El tiempo de ejecución de la heurística H1 modificada resultó ser, en promedio, casi 160 veces menor que el de la versión original. En cuanto a la calidad de las soluciones, la versión original entregó mejores soluciones en el

18.43 % de las instancias probadas. La Tabla 3.4 y la Tabla 3.5 presenta comparaciones detalladas antes y después de aplicar el procedimiento de purificación, respectivamente; $|D_{H1}|$, $|D_{H1_{mod}}|$, $|D_{H1}^*|$, $|D_{H1_{mod}}^*|$ representan la cardinalidad de los correspondientes conjuntos dominantes globales, antes y después de la purificación.

	Instancias	%
$ D_{H1} = D_{H1_{mod}} $	1558	68.21
$ D_{H1} > D_{H1_{mod}} $	190	8.32
$ D_{H1} < D_{H1_{mod}} $	536	23.47

Tabla 3.4: Análisis comparativo de las dos versiones de la heurística H1 antes de la purificación

	Instancias	%
$ D_{H1}^* = D_{H1_{mod}}^* $	1660	72.68
$ D_{H1}^* > D_{H1_{mod}}^* $	203	8.89
$ D_{H1}^* < D_{H1_{mod}}^* $	421	18.43

Tabla 3.5: Análisis comparativo de las dos versiones de la heurística H1 después de la purificación

Recordemos que la heurística H3 utiliza la heurística H1 como subrutina. Anteriormente se consideró una modificación de este último algoritmo, lo que resultó en un rendimiento mucho más rápido pero con una calidad de solución ligeramente peor. Modificaciones similares se realizaron en la heurística H3 y se obtuvieron resultados similares. La Tabla 3.6 muestra los resultados relacionados para unas 40 instancias seleccionadas. Entre las instancias resueltas por la heurística H3 y su modificación, en promedio, la heurística modificada fue aproximadamente 83 veces más rápida. Esta diferencia drástica se logró simplemente eliminando la iteración inicial de la heurística H3. En comparación con la heurística H2, en promedio, la heurística H3 modificada fue aproximadamente 5 veces más lenta. Los conjuntos dominantes globales entregados por la heurística H3 y su modificación fueron purificados en el 93.15 % y 93.22 % de las instancias, respectivamente (con un promedio de aproximadamente 30 vértices purificados). Las Tablas 3.7 y 3.8 presentan un análisis comparativo de las dos versiones de la heurística antes y después de la etapa de purificación, respectivamente.

V	E	L	H3			Purificación			H3 modificada			Purificación			U
			Tiempo(s)	$ D_{H3} $	Tiempo(s)	$ D_{H3}^* $	%	Tiempo(s)	$ D_{H3_{mod}} $	Tiempo(s)	$ D_{H3_{mod}}^* $	%	Tiempo(s)	$ D_{H3_{mod}}^* $	
7300	7311	2663	3934.11	3158	1.14	2706	14.31	1118.17	3158	1.10	2706	14.31	6843		
7350	7474	2621	4262.89	3174	1.17	2722	14.24	1114.77	3174	1.16	2722	14.24	6890		
7400	7474	2677	4306.95	3200	1.24	2740	14.38	1150.65	3200	1.19	2740	14.38	6937		
7500	7535	2762	4862.41	3270	1.24	2822	13.70	1209.28	3270	1.22	2822	13.70	7031		
7600	7734	2729	4510.99	3271	1.27	2829	13.51	1261.24	3271	1.21	2829	13.51	7125		
7650	7696	2797	3370.89	3331	1.20	2851	14.41	1287.06	3331	1.28	2851	14.41	7171		
7700	7716	2803	3418.88	3331	1.19	2860	14.14	1301.44	3331	1.28	2860	14.14	7218		
7800	7804	2879	3586.43	3395	1.32	2932	13.64	1359.95	3395	1.29	2932	13.64	7312		
7850	7884	2851	3645.01	3359	1.34	2910	13.37	1395.35	3359	1.25	2910	13.37	7359		
7950	7982	2949	6423.91	3479	1.49	3004	13.65	1710.06	3479	1.54	3004	13.65	7453		
8000	8126	2882	6742.18	3466	1.57	2964	14.48	1762.53	3466	1.65	2964	14.48	7500		
9150	10443603	3	8849.74	21	0.58	20	4.76	24.77	21	0.55	20	4.76	6485		
9250	10673366	3	8367.25	20	0.57	19	5.00	23.88	20	0.56	19	5.00	6523		
9300	14357729	3	9683.22	20	0.69	18	10.00	27.02	19	0.66	17	10.53	5703		
9350	14540834	2	9899.63	16	0.70	15	6.25	24.98	17	0.65	16	5.88	5700		
9400	11001589	3	8808.05	23	0.62	21	8.70	27.42	24	0.58	22	8.33	6579		
9450	14825554	3	10319.80	19	0.72	17	10.53	27.86	19	0.69	17	10.53	5742		
9500	14983162	3	10548.70	20	0.75	18	10.00	27.74	19	0.69	17	10.53	5791		
9550	15141604	3	10751.40	20	0.80	18	10.00	29.15	20	0.68	18	10.00	5773		
9600	15329626	2	10890.40	16	0.74	15	6.25	28.24	17	0.72	16	5.88	5800		
9650	15460987	3	10941.80	20	0.75	18	10.00	29.93	20	0.71	18	10.00	5790		
9700	15621929	3	15073.50	19	0.77	17	10.53	31.47	19	0.72	17	10.53	5832		
9800	11959739	3	13978.70	23	0.64	21	8.70	31.56	23	0.62	21	8.70	6805		
12600	39630183	2	25428.10	12	1.16	11	8.33	38.16	13	1.42	11	15.38	5406		
12650	39945571	2	25952.50	13	1.18	12	7.69	38.96	13	1.40	12	7.69	5419		
12700	40262208	2	26387.90	13	1.24	12	7.69	36.34	12	1.41	11	8.33	5436		
12750	40580096	2	26763.60	13	1.18	12	7.69	44.10	12	1.43	11	8.33	5437		
12800	40841707	3	31395.00	15	1.38	13	13.33	47.60	16	1.45	14	12.50	5438		
12850	41219621	2	31919.10	13	1.25	12	7.69	39.05	12	1.44	11	8.33	5450		
12900	41541258	2	32506.90	12	1.30	11	8.33	44.69	13	1.49	12	7.69	5448		
12950	41864146	2	34730.80	12	1.24	11	8.33	41.75	13	1.46	12	7.69	5448		
13000	42188283	2	33157.20	12	1.25	11	8.33	39.71	12	1.46	11	8.33	5433		
13100	42840308	2	33583.90	12	1.25	11	8.33	39.25	12	1.48	11	8.33	5446		
13150	43168196	2	34047.00	12	1.24	11	8.33	39.04	12	1.50	11	8.33	5442		

Tabla 3.6: Resultados de las dos versiones de la heurística H3

	Instancias	%
$ D_{H3} = D_{H3_{mod}} $	1630	71.36
$ D_{H3} > D_{H3_{mod}} $	179	7.84
$ D_{H3} < D_{H3_{mod}} $	475	20.80

Tabla 3.7: Análisis comparativo de las dos versiones de la heurística H3 antes de la purificación

	Instancias	%
$ D_{H3}^* = D_{H3_{mod}}^* $	1686	73.82
$ D_{H3}^* > D_{H3_{mod}}^* $	189	8.27
$ D_{H3}^* < D_{H3_{mod}}^* $	409	17.91

Tabla 3.8: Análisis comparativo de las dos versiones de la heurística H3 después de la purificación

En resumen, antes de aplicar el procedimiento de purificación, las tres heurísticas obtuvieron soluciones con igual cardinalidad en el 31 % de las instancias analizadas. Para las instancias restantes, las heurísticas H1, H2 y H3 obtuvieron las mejores soluciones en el 70.54 %, 8.69 % y 20.77 % de las instancias probadas, respectivamente. Al aplicar el procedimiento de purificación a las soluciones entregadas por las tres heurísticas, las soluciones purificadas correspondientes a H2 fueron mejores que las de H1 y H3. En promedio, el procedimiento de purificación redujo el tamaño de las soluciones entregadas por las heurísticas H1, H2 y H3 en un 16.03 %, 21.58 % y 15.34 %, respectivamente; ver Tablas 3.1 y 3.6. La Tabla 3.9 muestra el número de instancias donde las heurísticas encontraron las mejores soluciones.

	Instancias	%
All	1233	53.98
H1	121	5.30
H2	887	38.84
H3	43	1.88

Tabla 3.9: Análisis comparativo de las heurísticas H1, H2 y H3 tras la purificación

Comparando el tamaño de los mejores conjuntos dominantes globales obtenidos con la cota superior U , $|D^*| = \min\{|D_{H1}^*|, |D_{H2}^*|, |D_{H3}^*|\}$ representa aproximadamente el 10 % de U en promedio, es decir, una mejora del 90 % sobre esta cota superior. La cota inferior L representa,

en promedio, $1/3$ de $|D^*|$. Los datos experimentales completos se pueden encontrar en [91].

3.6.2. Rendimiento de IE vs. CPLEX e intlinprog

Para 25 instancias de tamaño medio y densidad media que IE encontró una solución óptima en un máximo de cuatro iteraciones, la Figura 3.12 ilustra cómo se actualizaron la cota inferior y superior durante la ejecución del algoritmo y cómo se redujo iterativamente la cardinalidad de los conjuntos dominantes formados. Los ejes verticales representan las cotas inferior y superior y las cardinalidades de las soluciones correspondientes, mientras que los ejes horizontales representan el tamaño de las instancias correspondientes. Los rangos para la cota inferior y superior están delimitados por regiones amarillas, las líneas grises muestran el valor actual de ν y los círculos azules representan a $\gamma_g(G)$.

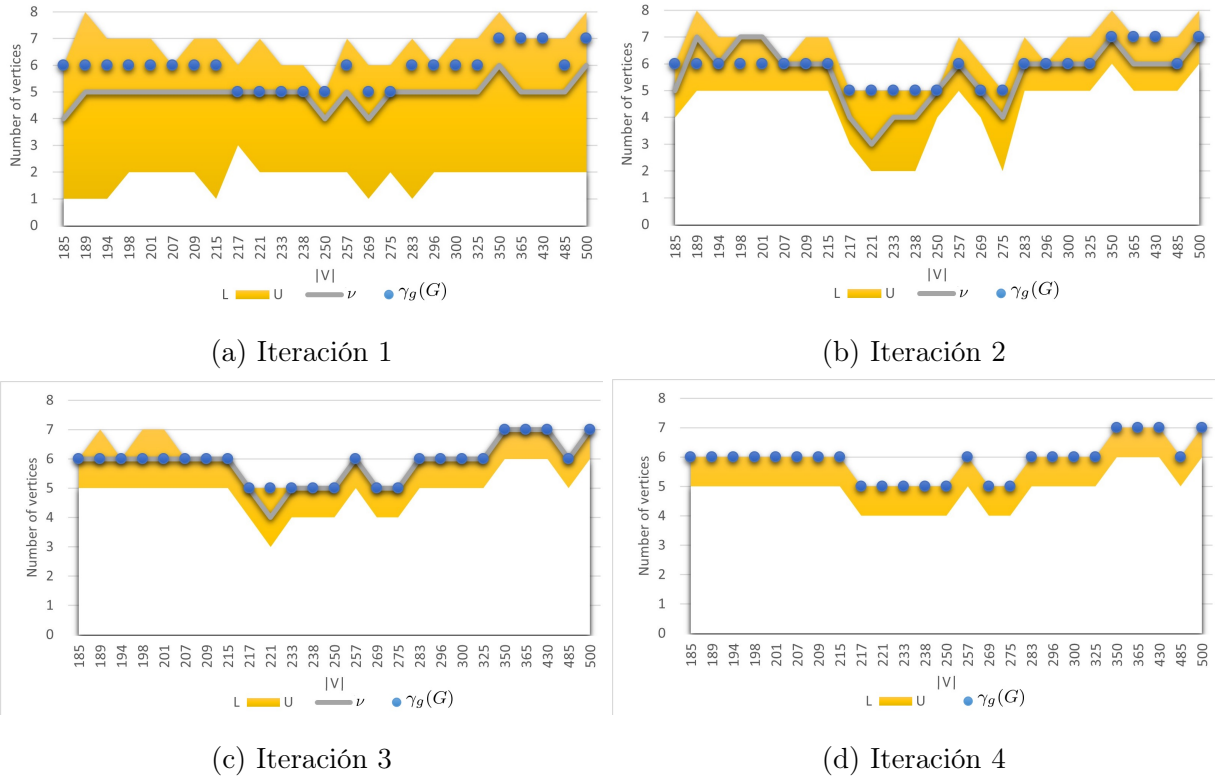


Figura 3.12: Cota inferior y superior del intervalo de búsqueda para el algoritmo IE en cuatro iteraciones consecutivas, y $\gamma_g(G)$

La Tabla 3.10 presenta los resultados para grafos con densidad inferior a 0.3 de ambos algoritmos exactos. Para estas instancias, los solvers *intlinprog* y CPLEX ofrecieron buenos resultados. Entre las instancias que fueron resueltas tanto por *intlinprog* como por CPLEX,

en promedio, CPLEX fue aproximadamente 16.26 veces más rápido. *Intlinprog* resolvió instancias con hasta 4000 vértices, y CPLEX resolvió instancias con hasta 8000 vértices. CPLEX superó la capacidad de memoria de la computadora para instancias con más de 8000 vértices.

$ V(G) $	$ E(G) $	Tiempo(s)				L	ILP^{L1}	$\gamma_g(G)$	U
		<i>intlinprog</i>	CPLEX	IE	H2				
50	286	0.18	0.09	0.20	0.0005	2	7	7	7
60	357	0.13	0.07	2.61	0.0013	2	6	6	7
70	524	0.16	0.08	3.14	0.0006	2	7	7	8
80	678	0.14	0.07	0.15	0.0011	2	6	6	6
90	842	0.18	0.11	64.32	0.0017	3	8	8	9
100	1031	0.23	0.09	0.23	0.0024	2	7	7	7
101	1051	1.31	0.13	1.32	0.0015	3	8	8	8
106	1154	1.57	0.10	97.52	0.0024	2	7	7	8
113	1306	1.34	0.18	1.36	0.0017	3	7	7	7
117	1398	2.73	0.17	104.51	0.0031	3	8	8	9
121	1493	3.11	0.11	3.14	0.0023	3	7	7	7
2550	2583	21.96	2.07	68.59	45.62	917	943	943	944
3050	3064	39.66	3.13	117.27	75.54	1106	1126	1126	1127
4000	4024	127.92	4.89	291.58	152.31	1456	1482	1482	1483
5000	5071	— — —	10.46	— — —	191.42	1795	1850	1850	1855
6000	6010	— — —	11.82	— — —	316.68	2186	2226	2226	2228
7000	7038	— — —	16.39	— — —	498.86	2548	2604	2604	2609
8000	8126	— — —	23.90	— — —	758.83	2882	2958	2958	2964

Tabla 3.10: Resultados para grafos con densidad inferior a 0,3

Se obtuvieron los siguientes resultados para los grafos de densidad media donde los algoritmos exactos lograron detenerse con una solución óptima dentro del límite de tiempo de 4 horas (ver Tabla 3.11 y Figura 3.13). Para los grafos de pequeño orden con hasta 260 vértices, CPLEX e *intlinprog* fueron más rápidos que el algoritmo IE. Ninguno de los algoritmos exactos terminó dentro de las 4 horas de tiempo de ejecución para grafos con más de 500 vértices. Para los grafos de tamaño intermedio, entre 260 y 500, IE fue 5.77 veces más rápido que *intlinprog*, mientras que IE fue más rápido que CPLEX en el 63.64 % de estas instancias. CPLEX fue 3.79 veces más rápido que *intlinprog*. Entre todos los grafos de densidad media (incluyendo instancias de pequeño tamaño con hasta 500 vértices), CPLEX fue más rápido que IE en el 72 % de ellos.

$ V(G) $	$ E(G) $	Tiempo(s)				L	ILP^{L1}	$\gamma_g(G)$	U
		<i>intlinprog</i>	CPLEX	IE	H2				
185	6849	3.25	1.15	23.63	0.0029	1	6	6	6
189	7147	3.87	1.22	22.55	0.0032	1	6	6	8
194	7529	4.09	1.77	28.29	0.0044	1	6	6	7
198	7842	3.71	1.30	31.53	0.0056	2	6	6	7
201	8081	4.82	1.65	31.03	0.0063	2	6	6	7
207	8569	5.28	1.89	33.91	0.0057	2	6	6	6
209	8735	7.41	1.80	35.65	0.0051	2	6	6	7
215	9243	12.25	4.43	41.79	0.0056	1	6	6	7
217	9415	19.16	11.59	32.18	0.0067	3	5	5	6

$ V(G) $	$ E(G) $	Tiempo(s)			H_2	L	ILP^{L1}	$\gamma_g(G)$	U
		<i>intlinprog</i>	CPLEX	IE					
221	9765	24.73	20.88	39.91	0.0079	2	5	5	7
233	10852	32.74	25.84	27.83	0.0066	2	5	5	6
238	11322	51.49	37.43	45.56	0.0092	2	5	5	6
250	12491	86.09	59.84	74.19	0.0096	2	5	5	5
257	13199	128.43	89.31	94.89	0.0079	2	4	6	7
269	14459	134.03	117.63	102.60	0.0109	1	3	5	6
275	15111	583.66	146.34	124.46	0.0152	2	4	5	6
283	16002	1166.80	376.79	436.23	0.0121	1	4	6	7
296	17505	5837.52	379.47	385.61	0.0141	2	5	6	6
300	17981	6418.32	418.20	405.03	0.0191	2	4	6	7
325	24896	7002.01	881.16	571.27	0.0131	2	5	6	7
350	27495	7854.36	4615.84	3057.62	0.0193	2	5	7	8
365	31606	8153.24	1734.93	6686.34	0.0160	2	4	7	7
430	44216	10121.83	5931.42	7001.24	0.0256	2	5	7	7
485	41536	11936.22	2211.16	1484.51	0.0308	2	4	6	7
500	42420	12351.49	5633.91	4525.39	0.0315	2	5	7	8

Tabla 3.11: Resultados para grafos con densidad $\approx 0,5$

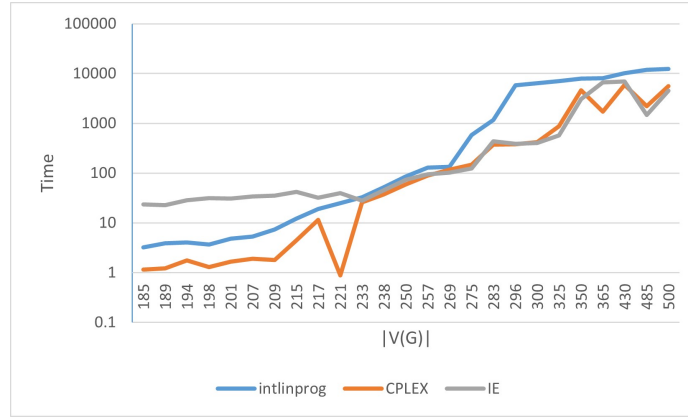


Figura 3.13: IE vs. ILP para grafos con densidad $\approx 0,5$ (se aplicó escala logarítmica)

La Tabla 3.12 presenta resultados para instancias más grandes de hasta 1098 vértices con alta densidad ($\approx 0,8$). Los algoritmos exactos se completaron en 120 minutos para las instancias más grandes. Para estos grafos cuasi-completos, $\gamma_g(G)$ resultó ser muy pequeño (oscila entre 4 y 7), donde *intlinprog* y CPLEX resultaron en tiempos de ejecución más pequeños (ver Figura 3.14). En promedio, CPLEX fue aproximadamente 6.56 veces más rápido que *intlinprog*.

$ V(G) $	$ E(G) $	Tiempo(s)			H_2	L	ILP^{L1}	$\gamma_g(G)$	U
		<i>intlinprog</i>	CPLEX	IE					
474	97121	20.11	0.24	113.59	0.0093	2	4	4	5
518	116381	32.56	0.50	160.37	0.0113	2	4	4	5
580	130202	37.67	0.26	746.51	0.0211	2	5	5	6
634	175537	33.93	1.06	410.16	0.0291	2	4	4	5
712	222137	39.85	0.96	646.24	0.0260	2	4	4	5
746	244124	44.08	1.27	753.398	0.0267	2	4	4	5
816	260190	47.24	3.87	2604.72	0.0498	2	5	5	6
892	350495	41.63	3.34	3017.06	0.0505	2	5	5	5
932	382996	47.28	1.21	1805.78	0.0502	2	4	4	5

$ V(G) $	$ E(G) $	Tiempo(s)			$H2$	L	ILP^{L1}	$\gamma_g(G)$	U
		<i>intlinprog</i>	CPLEX	IE					
970	415180	43.83	1.54	4186.27	0.0534	2	5	5	5
972	416905	37.55	6.96	4360.8	0.0553	2	5	5	5
1014	454116	40.84	5.66	5136.9	0.0594	2	5	5	5
1036	474219	48.64	7.03	5341.2	0.0612	2	5	5	5
1042	426451	44.52	12.44	6742.94	0.0895	2	5	5	6
1064	500420	40.21	1.79	6224.74	0.0789	2	5	5	5
1068	504205	41.84	1.24	3173.52	0.1228	2	4	4	5
1080	515722	44.24	0.89	6698.43	0.0982	2	5	5	5
1096	531243	52.96	0.86	7010.88	0.0786	2	5	5	5
1098	533220	51.51	0.90	7246.31	0.0825	2	5	5	5

Tabla 3.12: Resultados de grafos con densidad $\approx 0,8$

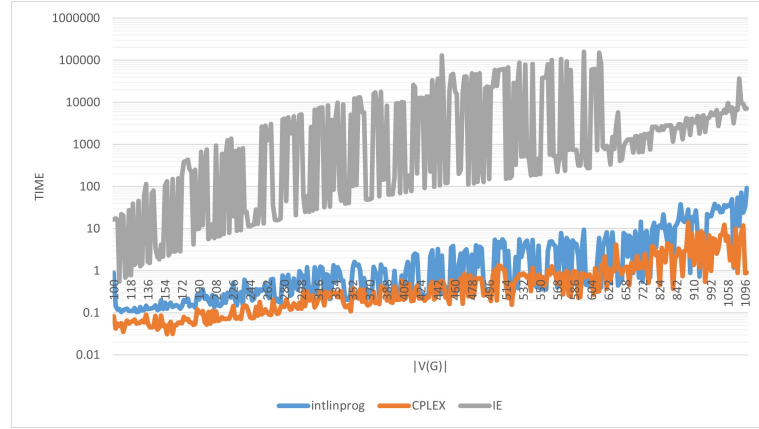


Figura 3.14: IE vs. ILP para grafos densos (se aplicó escala logarítmica)

Note que, en general, la cota inferior L del Corolario 3.1 es muy pobre en comparación con ILP^L y $\gamma_g(G)$. Los solvers *intlinprog* y CPLEX alcanzaron una solución óptima en menos de 2 minutos para todas las instancias con densidad aproximada 0,2 y 0,8, como se muestra en la Tabla 3.10 y la Tabla 3.12, respectivamente, es decir, $ILP^L = \gamma_g(G)$ para todas las instancias en estas tablas. Para las instancias con densidad aproximadamente 0,5, *intlinprog* y CPLEX alcanzaron una solución óptima en menos de 2 minutos solo para instancias con menos de 250 vértices. Para las restantes instancias de referencia, la diferencia entre $\gamma_g(G)$ y las cotas inferiores ILP^{L1} e ILP^{L2} fue, en promedio, de 3.02 y 4.18 vértices, respectivamente.

3.7. Conclusiones

En este capítulo se describieron algoritmos exactos y de aproximación para el problema de dominación global y se demostró que sigue siendo NP -hard para familias particulares de grafos. También se presentaron algunas familias de grafos donde las heurísticas propuestas

garantizan una solución óptima. Dado que no existía un algoritmo de solución para este problema, se utilizaron los valores óptimos obtenidos por IE, *intlinprog* y CPLEX para analizar el comportamiento práctico de las heurísticas, que resultaron ser muy favorables, tanto en la calidad de las soluciones obtenidas como en el tiempo requerido para ello. También se emplearon cotas superiores e inferiores conocidas para medir el rendimiento práctico de las heurísticas. En el algoritmo IE, el uso de cotas inferiores obtenidas a través de la relajación de la formulación ILP del problema resultó ser más efectivo que el uso de las cotas inferiores existentes en la literatura. Aún así, CPLEX resultó ser más rápido que el algoritmo IE para la mayoría de las instancias de referencia de tamaño pequeño y moderado.

Conjunto Dominante Global Total

4.1. Introducción

Para definir lo que es un conjunto dominante global total, es necesario introducir las siguientes definiciones.

Definición 4.1. *Un subconjunto no vacío $D \subseteq V$ es un conjunto dominante total de un grafo $G(V, E)$ si cada vértice en V es adyacente a al menos un vértice en D .*

Definición 4.2. *El número de dominación total de un grafo G , $\gamma_t(G)$, es el número de vértices en el conjunto dominante total más pequeño.*

Un conjunto dominante total con cardinalidad $\gamma_t(G)$ se denota como $\gamma_t(G)$ -conjunto (ver [10, 11, 13, 46, 65, 76]). Nótese que, por las definiciones anteriores, si queremos construir un conjunto dominante total, el grafo G no puede tener vértices aislados (con grado 0) y $\gamma_t(G) \geq 2$.

Definición 4.3. *Un subconjunto D de los vértices de un grafo G es un conjunto dominante global total (GTDS, por sus siglas en inglés) si D es un conjunto dominante total tanto en G como en su complemento $\overline{G}$.*

Definición 4.4. *El número de dominación global total de un grafo G , $\gamma_t^g(G)$, es el número de vértices en su conjunto dominante global total más pequeño.*

Un $\gamma_t^g(G)$ -conjunto es un conjunto dominante global total de cardinalidad $\gamma_t^g(G)$ (ver [1, 64, 67, 75]). El objetivo en este capítulo es encontrar y estudiar tal conjunto.

Como es fácil de ver, si queremos construir un GTDS $D \subseteq V(G)$ de un grafo G , todos los vértices en $V(G)$ deben tener al menos un vecino y un no vecino en D ; además, el grafo G no puede tener vértices aislados ni vértices con grado $n - 1$, donde n es el número de vértices en el grafo G . A partir de estas observaciones, se tiene que $4 \leq \gamma_t^g(G) \leq n$ y solo se pueden analizar grafos de orden $n \geq 4$.

Los GTDS son un concepto fundamental en la teoría de grafos con posibles aplicaciones en el diseño de redes, localización de instalaciones y problemas de optimización. Uno de los primeros trabajos significativos sobre este tema es el de [75], que introduce y estudia algunas propiedades básicas de los GTDS en grafos. Dos años más tarde, [64] complementó este estudio proporcionando nuevas cotas inferiores y superiores para el problema.

En [12], se demuestra que, si $G_1, G_2, \dots, G_r$ ($r \geq 2$) son las componentes conexas de un grafo G , entonces

$$\gamma_t^g(G) = \sum_{i=1}^r \gamma_t^g(G_i).$$

Por esta razón, la Sección 4.2 se centra en el estudio de grafos conexos.

En los últimos años, ha habido un interés significativo en el estudio de los GTDS debido a su relevancia en aplicaciones prácticas y a sus ricas propiedades teóricas. El objetivo es encontrar un GTDS de cardinalidad mínima, un *conjunto dominante global total óptimo*. Sin embargo, como se demostró en [1], este problema ha sido probado como *NP*-completo. Además, como se señala en [87] y [88], el problema sigue siendo *NP*-completo incluso para grafos bipartitos y cordales. En [87], se propone un algoritmo de tiempo polinomial para encontrar un conjunto *k*-dominante global total mínimo (GTkDS) específicamente en grafos bipartitos y cordales, junto con un algoritmo de aproximación con un radio de aproximación de $2(1 + \ln |V|)$ para el problema del GTkDS en cualquier grafo. Notablemente, los autores se abstienen de presentar resultados experimentales, enfatizando el carácter teórico de sus hallazgos.

Por otro lado, [1] y [12] sugirieron algunas cotas inferiores y superiores para el problema. Hasta donde sabemos, no se conocen otros resultados. En particular, no se han propuesto anteriormente algoritmos exactos ni aproximados para el problema del GTDS en ningún tipo de grafos.

En este capítulo, se proponen nuevas cotas inferiores y superiores para $\gamma_t^g(G)$. Además, se presenta un método de solución exacto basado en una formulación ILP del problema, junto con tres heurísticas y un procedimiento especialmente diseñado para reducir el tamaño de los GTDS generados por las heurísticas. Asimismo, se presentan resultados experimentales y estudios computacionales para evaluar el rendimiento de la formulación ILP y las heurísticas en 1130 instancias. La formulación ILP proporcionó soluciones óptimas para instancias con hasta 8000 vértices. Se determinó el valor de $\gamma_t^g(G)$ para 896 instancias con $50 \leq |V(G)| \leq 8000$, que sirven como instancias de referencia para futuras investigaciones. Los valores de $\gamma_t^g(G)$ y las instancias están disponibles en [92]. Es notable que, para aproximadamente el 41 % de las instancias en las que la formulación ILP logró una solución óptima, al menos una de las heurísticas también generó una solución óptima. Además, el error absoluto promedio para las instancias restantes fue de aproximadamente 1.58 vértices.

En resumen, este capítulo contribuye al entendimiento y avance de los GTDS, ofreciendo ideas sobre sus propiedades estructurales y proporcionando algoritmos prácticos para su cálculo. En la Sección 4.2, se presentan nuevas propiedades combinatorias sobre el $\gamma_t^g(G)$ y se describe una formulación ILP del problema. En la Sección 4.3, se describen las tres heurísticas, se establecen familias de grafos para las cuales las heurísticas generan una solución óptima, y se detalla el procedimiento de purificación. La Sección 4.4 contiene la descripción de los resultados experimentales.

4.2. Cotas y resultados exactos para $\gamma_t^g(G)$

Se comienza esta sección con algunos resultados conocidos que se utilizarán más adelante. Akhbari y otros en [1] obtuvieron el siguiente resultado.

Lema 4.1. *Sea G un grafo con grado máximo $\Delta(G)$.*

- (i) Si $\gamma_t(G) > \Delta(G) + 1$, entonces $\gamma_t^g(G) = \gamma_t(G)$.
- (ii) Si $n > \Delta(G)(\Delta(G) + 1)$, entonces $\gamma_t^g(G) = \gamma_t(G)$.

Claramente, la definición de un GTDS proporciona de manera directa una cota inferior para el parámetro $\gamma_t^g(G)$.

Observación 4.1. Sea G un grafo, entonces

- (i) $\gamma_t^g(G) \geq \max\{\gamma_t(G), \gamma_t(\overline{G})\}$.
- (ii) Si G no es un grafo conexo, entonces $\gamma_t^g(G) = \gamma_t(G)$.
- (iii) $\gamma_t^g(G) \geq 4$.

Kulli y otros en [75] obtuvieron la siguiente proposición.

Proposición 4.1. Sea G un grafo.

- (i) Si $\text{diam}(G) = 3$, entonces $\gamma_t^g(G) \leq \gamma_t(G) + 2$.
- (ii) Si $\text{diam}(G) = 4$, entonces $\gamma_t^g(G) \leq \gamma_t(G) + 1$.
- (iii) Si $\text{diam}(G) \geq 5$, entonces $\gamma_t^g(G) = \gamma_t(G)$.

Akhbari y otros en [1] obtuvieron el siguiente resultado.

Proposición 4.2. Para cualquier $k \geq 4$ hay un grafo G tal que $\text{diam}(G) = \text{diam}(\overline{G}) = 2$ y $\gamma_t^g(G) = k$.

La Figura 4.1 muestra otro grafo G , diferente del ciclo C_5 , tal que $\text{diam}(G) = 2$, $\text{diam}(\overline{G}) = 2$, $\gamma_t^g(G) = 4$, $\gamma_t(G) = 3$ y $\delta(G) = 2$.

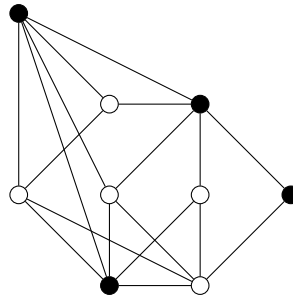


Figura 4.1: Grafo G con $\text{diam}(G) = \text{diam}(\overline{G}) = 2$

Lema 4.2. Para cualquier grafo G , $\gamma_t(\overline{G}) = 2$ si y solo si $\text{diam}(G) \geq 3$.

Demostración. Por un lado, si $\gamma_t(\overline{G}) = 2$, entonces $D = \{u, v\}$ es un conjunto dominante total (TDS) de $\overline{G}$, por lo que u y v no son adyacentes en G , así que $d(u, v) \geq 2$. Además, si $d(u, v) = 2$, entonces tienen un vecino en común, por lo que D no es un TDS en $\overline{G}$, lo cual es una contradicción. Por lo tanto, $\text{diam}(G) \geq 3$. Por otro lado, si existen $u, v \in V$ tales que $d(u, v) \geq 3$, entonces no son vértices adyacentes y cualquier vértice $z \in V \setminus \{u, v\}$ no es adyacente a ambos u y v , así que $\{u, v\}$ es un TDS de $\overline{G}$. ■

Proposición 4.3. Sea G un grafo con grado mínimo $\delta(G) = 2$, tal que $\text{diam}(G) = 2$ y $\text{diam}(\overline{G}) = 2$, entonces $\gamma_t^g(G) = 4$.

Demostración. Dado que $\text{diam}(\overline{G}) = 2$, entonces $\gamma_t(G) \geq 3$ (ver Lema 4.2). Si $v \in V$ satisface $N(v) = \{u_1, u_2\}$, dado que $\{u_1, u_2\}$ es un conjunto dominante, tenemos que u_1 y u_2 no son vértices adyacentes ya que $\gamma_t(G) \geq 3$. Entonces, al tomar cualquier vecino $w \neq v$ de u_1 , tenemos que $\{v, u_1, u_2, w\}$ es un GTDS. ■

Note que el grafo de Petersen y su complemento tienen diámetro 2, pero el grado mínimo es 3 y $\gamma_t^g(G) = 5$ (ver Figura 4.2).

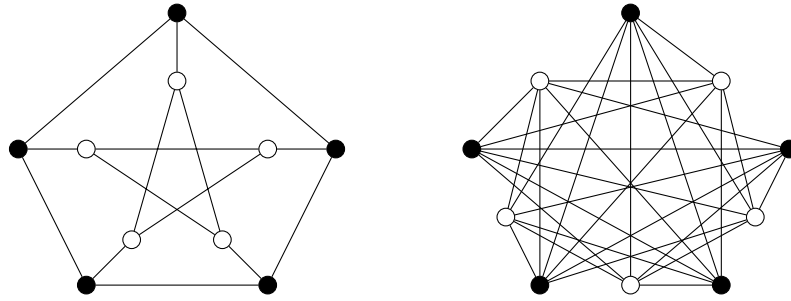


Figura 4.2: Grafo de Petersen y su complemento

Proposición 4.4. Si G es un grafo libre de triángulos y $\gamma_t(G) = 2$, entonces $\gamma_t^g(G) = 4$.

Demostración. Si $D = \{v_1, v_2\}$ es un TDS mínimo, dado que $g(G) \geq 4$, cada vértice en $V \setminus D$ tiene exactamente un vecino en D . Por lo tanto, al tomar cualquier vértice u_1 no adyacente a v_1 y cualquier vértice u_2 no adyacente a v_2 , tenemos que $D \cup \{u_1, u_2\}$ es un GTDS mínimo. ■

Teorema 4.1. *Sea G un grafo libre de triángulos, entonces $\gamma_t(G) \leq \gamma_t^g(G) \leq \gamma_t(G) + 1$.*

Demostración. La cota inferior es muy conocida y se obtiene fácilmente. Por otro lado, sea D un $\gamma_t(G)$ -conjunto, supongamos que para cada $v \in D$ tenemos que $1 \leq \delta_D(v) < |D| - 1$. Ahora, es fácil ver que para cada $v \in D$ existe al menos un vértice no adyacente a v . También tenemos que para cada $v \in V \setminus D$ se cumple que $1 \leq \delta_D(v) < |D|$. Note que, de lo contrario, $g(G) = 3$, lo cual es una contradicción. Por lo tanto, D es un GTDS y $\gamma_t^g(G) = |D| \leq \gamma_t(G) + 1$. Además, supongamos que existe $v \in D$ tal que $\delta_D(v) = |D| - 1$ (note que, si existe, es único). Note que, para todo $w \in D - \{v\}$, se tiene que existe $u \in V \setminus D$ tal que $u \sim w$ y $u \not\sim v$. Así, $D \cup \{u\}$ es un GTDS. Por lo tanto, $\gamma_t^g(G) \leq |D| + 1 = \gamma_t(G) + 1$. Entonces, $\gamma_t(G) \leq \gamma_t^g(G) \leq \gamma_t(G) + 1$. ■

Si G es el grafo de Petersen, entonces $g(G) = 5$, $\gamma_t(G) = 4$ y $\gamma_t^g(G) = 5$. Si G es el producto cartesiano $P_2 \square P_4$, tenemos que $g(G) = 4$, $\gamma_t(G) = 4 = \gamma_t^g(G)$. Además, sabemos que el diámetro del grafo de Petersen es 2.

El siguiente resultado se prueba en [38].

Lema 4.3. *Si G es un grafo con $\text{diam}(G) = 2$ y grado mínimo $\delta(G)$, entonces $\gamma_t(G) \leq \delta(G) + 1$.*

Por el Lemma 4.3 y el Teorema 4.1 se obtiene el siguiente resultado.

Proposición 4.5. *Sea G un grafo libre de triángulos con $\text{diam}(G) = 2$ y grado mínimo $\delta(G)$, entonces $\gamma_t^g(G) \leq \delta(G) + 2$.*

Lema 4.4. *Para cualquier grafo G , $\gamma_t^g(G) = \gamma_t(G)$ si y solo si existe un TDS mínimo D tal que, para cada vértice $v \in V$, se satisface que $D \not\subseteq N(v)$.*

Demostración. Si $\gamma_t^g(G) = \gamma_t(G)$, entonces existe un TDS mínimo D que también es un GTDS, por lo que cada vértice $v \in V$ satisface $\bar{\delta}_D(v) \geq 1$, es decir, $D \not\subseteq N(v)$. Si existe un TDS mínimo D tal que, para cada vértice $v \in V$, se satisface que $D \not\subseteq N(v)$, entonces cada vértice $v \in V$ tiene un no vecino en D , es decir, D es un GTDS. ■

Por el Teorema 4.1 y el Lema 4.4 se obtiene el siguiente resultado.

Proposición 4.6. *Sea G un grafo libre de triángulos y D un TDS mínimo. Si existe un vértice $v \in V$ tal que $D \subseteq N(v)$, entonces $\gamma_t^g(G) = \gamma_t(G) + 1$.*

Si consideramos un grafo G con $\text{diam}(G) = 2$, es claro que $g(G) \in 3, 4, 5$. Podríamos pensar que para cualquier grafo G con $\text{diam}(G) = 2$ y $g(G) \in 3, 4$, tenemos que $\gamma_t^g(G) = 4$. Sin embargo, eso no es cierto para $g(G) = 4$. Por ejemplo, en los grafos G y G' mostrados en la Figura 4.3, vemos que $\gamma_t^g(G) = 5$ y $\gamma_t^g(G') = 4$.

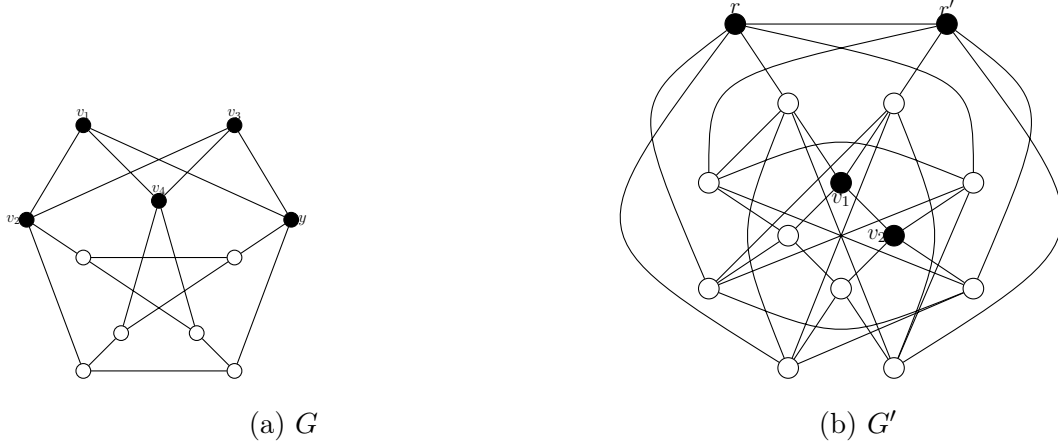


Figura 4.3: Grafo G y G' tales que $\text{diam}(G) = 2$ y $g(G) = 4$

En [38] se prueba el siguiente lema:

Lema 4.5. *Si G es un grafo con diámetro 2, orden n y cuello 5, entonces $\gamma_t(G) = 1 + \sqrt{n-1}$.*

Proposición 4.7. *Sea G un grafo con $\text{diam}(G) = 2$ y $g(G) = 5$. Entonces, $\gamma_t^g(G) = 4$ si y solo si G es un ciclo de 5 vértices.*

Demostración. Se sabe que $\gamma_t^g(C_5) = 4$. Si $\text{diam}(G) = 2$, $g(G) = 5$ y n es el orden de G , por el Lema 4.5 se tiene que $\gamma_t(G) = 1 + \sqrt{n-1}$. Si suponemos que $\gamma_t^g(G) = 4$, dado que $\gamma_t(G) \leq \gamma_t^g(G) \leq \gamma_t(G) + 1$, tenemos que $\gamma_t(G) \in \{3, 4\}$, por lo tanto $n \in \{5, 10\}$, en consecuencia, G es un ciclo de 5 vértices o el grafo de Petersen. Pero, si G es el grafo de Petersen, se tiene que $\gamma_t^g(G) = 5$. ■

Recordemos que, dado un conjunto de vértices S y $v \in S$, u es un vecino privado de v con respecto a S si $u \in N(v)$ y $u \notin N(v')$ para cualquier $v' \in S \setminus \{v\}$. El conjunto de todos los vecinos privados de v con respecto a S se denota por $pn[v, S]$.

Lema 4.6. Sea $G = (V, E)$ un grafo con orden n y grado máximo $\Delta(G) \leq n - 2$. Si existe un vértice $v \in V$ tal que $N(v)$ es un conjunto dominante, entonces existe un GTDS S de cardinalidad $\delta(v) + 3$.

Demostración. Denotamos $A = N(v) = \{u_1, \dots, u_t\}$. Si existe $u_i \in N(v)$ y $w \in pn[u_i, A]$, entonces, si tomamos cualquier vértice $z \notin N[u_i]$, se puede verificar que $N[v] \cup \{w, z\}$ es un GTDS. Por lo tanto, podemos suponer que ningún vértice en A tiene un vecino privado con respecto a A . Si $\delta_A(u_i) < t - 1$ para cada $i \in \{3, 4, \dots, t\}$, tomando $w_1 \notin N[u_1]$ y $w_2 \notin N[u_2]$, podemos ver que $A \cup \{v, w_1, w_2\}$ es un GTDS. Entonces podemos suponer que $\delta_A(u_i) = t - 1$ para $i \in \{1, \dots, s\}$ y $s \geq 3$. Para cualquier $i \in \{1, \dots, s\}$, existe un vértice $w_i \in V \setminus N[v]$ tal que $w_i \notin N(u_i)$, por lo tanto, $S = A \cup \{v, w_1, \dots, w_s\}$ es un GTDS. Pero, dado que ningún vértice en $N(v)$ tiene vecinos privados, entonces $S_1 = S \setminus \{u_1\}$ también es un GTDS. Si tomamos $A_1 = A \setminus \{u_1\}$ y existe un vértice $u_i \in A_1$ tal que $|pn[u_i, A_1]| \neq 0$, podemos tomar $z \in pn[u_i, A_1]$ y $w_i \notin N[u_i]$ y $A_1 \cup \{v, w_1, z, w_i\}$ es un GTDS con cardinalidad $\delta(v) + 3$. Si para cualquier $u_i \in A_1$ tenemos $|pn[u_i, A_1]| = 0$, tomamos $A_2 = A \setminus \{u_1, u_2\}$. Si existe un vértice $u_i \in A_2$ tal que $|pn[u_i, A_2]| \neq 0$, podemos tomar $z \in pn[u_i, A_2]$ y $w_i \notin N[u_i]$, y $A_2 \cup \{v, w_1, w_2, z, w_i\}$ es un GTDS con cardinalidad $\delta(v) + 3$. Podemos continuar este método para obtener el resultado. ■

Proposición 4.8. Si G es un grafo con $\text{diam}(G) = 2$ y grado mínimo $\delta(G)$, entonces $\gamma_t^g(G) \leq \delta(G) + 3$.

Demostración. Note que si G es un grafo con $\text{diam}(G) = 2$ y v un vértice de grado mínimo, entonces $N(v)$ es un conjunto dominante de G . Por el Lema 4.6, tenemos que $\gamma_t^g(G) \leq \delta(G) + 3$. ■

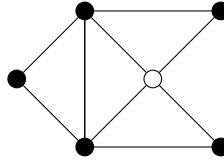


Figura 4.4: Grafo G con $\delta(G) = 2$, $\text{diam}(G) = 2$ y $\gamma_t^g(G) = 5$

Proposición 4.9. Para cualquier $\delta(G)$ existe un grafo $G = (V, E)$ con grado mínimo $\delta(G)$, $\text{diam}(G) = 2$, $g(G) = 3$ y $\gamma_t^g(G) = \delta(G) + 3$.

Demostración. Considere un grafo completo $K_{\delta(G)+1}$, cuyos vértices son $\{v, u_1, u_2, \dots, u_{\delta(G)}\}$, y otros $2\delta(G)$ vértices $\{w_1, w_2, \dots, w_{\delta(G)}, z_1, z_2, \dots, z_{\delta(G)}\}$, con las siguientes aristas:

- (1) $\{u_i w_j\} \in E$ para todo $i \neq j$,
- (2) $\{u_i z_j\} \in E$ para todo i, j ,
- (3) $\{z_i w_j\} \in E$ para todo i, j .

Dado que $N[u_i] = V \setminus \{w_i\}$ para cada $i \in \{1, \dots, \delta(G)\}$, entonces w_i (los vértices negros en la Figura 4.5) debe pertenecer a cualquier GTDS para cada $i \in \{1, \dots, \delta(G)\}$. Finalmente, cualquier vértice en $\{u_1, u_2, \dots, u_{\delta(G)}\}$ junto con cualquier vértice en $\{z_1, z_2, \dots, z_{\delta(G)}\}$ formaría una estrella con $\{w_1, w_2, \dots, w_{\delta(G)}\}$, por lo que se necesitaría otro vértice para obtener un GTDS. ■

Por ejemplo, $\{v, u_1, z_1, w_1, w_2, \dots, w_{\delta(G)}\}$ (vértices negros y grises en la Figura 4.5) es un GTDS mínimo de G .

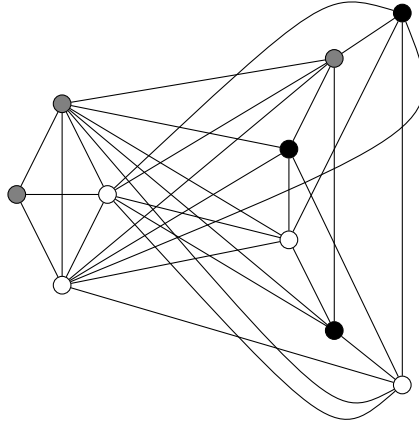


Figura 4.5: Grafo G con $\delta(G) = 3$ y $\gamma_t^g(G) = 6$

El siguiente resultado está demostrado en [12].

Lema 4.7 ([12]). *Sea G un grafo. Si $\gamma_t(G) \leq \Delta(G) + 1$, entonces $\gamma_t^g(G) \leq \Delta(G) + 2$.*

Por los Lemas 4.3 y 4.7, se obtiene el siguiente resultado, que mejora la Proposición 4.8 para grafos regulares. Para mayor claridad, se proporciona una nueva demostración.

Proposición 4.10. *Si G es un grafo con $\text{diam}(G) = 2$ y grado máximo $\Delta(G)$. Entonces, $\gamma_t^g(G) \leq \Delta(G) + 2$.*

Demostración. Sea v un vértice de G con grado $\Delta(G)$. Denotemos $A = N_G(v)$, de manera que $|A| = \Delta(G)$. Sea w un vértice arbitrario en $V \setminus N_G[v]$. Ahora, definamos $D = A \cup \{v, w\}$. Es fácil ver que v y w tienen al menos un vértice adyacente en A y que no son vecinos entre sí. Observamos que $|D| = \Delta(G) + 2$, entonces cada vértice en A tiene al menos un vértice adyacente y uno no adyacente en D . Por otro lado, haciendo el mismo análisis anterior, cada vértice en $V \setminus (N_G[v] \cup \{w\})$ tiene al menos un vértice adyacente y uno no adyacente en D . Por lo tanto, D es un GTDS, y así $\gamma_t^g(G) \leq \Delta(G) + 2$. ■

Proposición 4.11. *Sea G un grafo con grado mínimo $\delta(G) = 1$, $\{u_1, u_2, \dots, u_r\}$ las hojas de G y $\{v_1, v_2, \dots, v_t\}$ los correspondientes soportes. Si $G' = G \setminus \{u_1, u_2, \dots, u_r\}$ entonces, $\gamma_t^g(G) \leq \gamma_t^g(G') + t$.*

Demostración. Si consideramos $G' = G \setminus \{u_1, u_2, \dots, u_r\}$ y D' un GTDS mínimo de G' , entonces $D' \cup \{v_1, v_2, \dots, v_t\}$ es un GTDS de G . Por lo tanto, $\gamma_t^g(G) \leq \gamma_t^g(G') + t$. ■

En la siguiente imagen se puede ver que la cota superior anterior es ajustada.

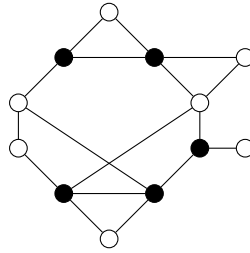


Figura 4.6: Grafo G con $\text{diam}(G) = 4$ y $\gamma_t^g(G) = \gamma_t^g(G') + 1$

Proposición 4.12. *Si G es un grafo planar y libre de triángulos con $\text{diam}(G) = 2$, entonces $\gamma_t^g(G) = 4$.*

Demostración. Por el Teorema 4.1, obtenemos directamente que $\gamma_t^g(G) \leq \gamma_t(G) + 1$. En [56] se prueba el siguiente resultado: Si G es un grafo plano con $\text{diam}(G) = 2$, entonces $\gamma_t(G) \leq 3$. Se tiene directamente que $\gamma_t^g(G) \leq \gamma_t(G) + 1 \leq 4$. Por lo tanto, $\gamma_t^g(G) = 4$. ■

La implicación opuesta en la proposición anterior no es verdadera. Por ejemplo, si $G = C_5$, el ciclo con cinco vértices, entonces $\text{diam}(G) = 2 = \text{diam}(\overline{G})$ y $\gamma_t(G) = \gamma_t(\overline{G}) = 3$, pero $\gamma_t^g(G) = 4$. Si $G = C_6$, entonces $\text{diam}(\overline{G}) = 2$ y $\gamma_t^g(G) = 4 < 4 + 2 = \gamma_t(G) + \gamma_t(\overline{G})$.

Bermudo y otros, en [12] obtuvieron las siguientes proposiciones.

Proposición 4.13. *Sea P_n y C_n los grafos camino y ciclo de orden n , respectivamente, con $n \geq 4$. Entonces, se tienen los siguientes $\gamma_t^g(G)$:*

$$\gamma_t^g(P_n) = \gamma_t^g(C_n) = \begin{cases} 4 & \text{if } n = 4, 5 \\ \frac{n}{2} + 1 & \text{if } n \equiv 2 \pmod{4} \\ \lceil \frac{n}{2} \rceil & \text{e.c.c.} \end{cases}$$

Proposición 4.14. *Sea G un grafo cúbico de orden n , entonces*

$$\gamma_t^g(G) = \begin{cases} \gamma_t(G) & \text{if } \gamma_t(G) > 4 \\ 5 & \text{if } \gamma_t(G) = 4 \text{ and for any } \gamma_t(G)\text{-set } D \text{ of } G \text{ is } G[D] = S_3 \\ 4 & \text{e.c.c.} \end{cases}$$

Sean $G_1(V_1, E_1)$ y $G_2(V_2, E_2)$ grafos. La unión de los grafos $G_1 + G_2$ resulta en $G'(V', E')$ donde $V' = V_1 \cup V_2$ y $E' = E_1 \cup E_2 \cup \{\{x, y\} | x \in V_1 \wedge y \in V_2\}$.

Proposición 4.15. *Sea $G' = G_1 + G_2$ el grafo resultante de la unión de G_1 y G_2 , donde $|V(G_1)|, |V(G_2)| \geq 3$, $\Delta(G_1) \leq |V(G_1)| - 2$ y $\Delta(G_2) \leq |V(G_2)| - 2$, entonces $\gamma_t^g(G') = \gamma_t(\overline{G_1}) + \gamma_t(\overline{G_2})$.*

Demostración. Dado que $\Delta(G_1) \leq |V(G_1)| - 2$ y $\Delta(G_2) \leq |V(G_2)| - 2$, entonces $\overline{G_1}$ y $\overline{G_2}$ son las componentes conexas del grafo disconexo $\overline{G'}$. Por la Observación 4.1 (ii), tenemos que $\gamma_t^g(\overline{G'}) = \gamma_t(\overline{G_1}) + \gamma_t(\overline{G_2})$. Note que $\gamma_t^g(G') = \gamma_t^g(\overline{G'})$, entonces $\gamma_t^g(G') = \gamma_t(\overline{G_1}) + \gamma_t(\overline{G_2})$. ■

Como consecuencia directa, se obtiene el siguiente corolario.

Corolario 4.1. *Sea $K_{n,m}$ un grafo bipartito completo de orden $n + m$ y $n, m \geq 2$, entonces $\gamma_t^g(K_{n,m}) = 4$.*

Proposición 4.16. *Si G es un grafo tal que $\gamma_t^g(G) = \gamma_t(G) + \gamma_t(\overline{G})$, entonces $\text{diam}(G), \text{diam}(\overline{G}) \leq 3$.*

Demostración. Si $\text{diam}(G) = 4$, por la Proposición 4.1, $\gamma_t^g(G) \leq \gamma_t(G) + 1 < \gamma_t(G) + 2$. Por el Lema 4.2, $\gamma_t(\overline{G}) = 2$, entonces $\gamma_t^g(G) < \gamma_t(G) + \gamma_t(\overline{G})$, es una contradicción. Si $\text{diam}(G) \geq 5$, entonces $\gamma_t^g(G) = \gamma_t(G) < \gamma_t(G) + 2 \leq \gamma_t(G) + \gamma_t(\overline{G})$, también una contradicción. Dado que $\gamma_t^g(G) = \gamma_t^g(\overline{G})$, lo mismo ocurre con $\text{diam}(\overline{G})$. ■

Sea G' y H dos grafos de orden n' y m , respectivamente. El producto corona $G' \odot H$ se define como el grafo obtenido a partir de G' y H tomando una copia de G' y n' copias de H y uniendo por una arista cada vértice de la i -ésima copia de H con el i -ésimo vértice de G .

Proposición 4.17. *Sea G' un grafo de orden n' , sea H cualquier grafo y consideremos el producto corona $G = G' \odot H$. Entonces,*

- (a) *Si G' es un grafo completo, entonces $\gamma_t^g(G) = n' + 2$.*
- (b) *Si G' no es un grafo completo y $\Delta(G') = n' - 1$, entonces $\gamma_t^g(G) = n' + 1$.*
- (c) *Si $\Delta(G') < n' - 1$, entonces $\gamma_t^g(G) = n'$.*

Demostración. Si D es un GTDS y denotamos por H_v la copia de H correspondiente al vértice $v \in V(G)$, entonces $|D \cap (\{v\} \cup V(H_v))| \geq 1$, por lo que $|D| \geq n'$. Si $v \in V(G) \setminus D$, entonces $|D \cap V(H_v)| \geq 2$, por lo que podemos suponer que $V(G') \subseteq D$. (a) Dado que cualquier vértice en D necesita un no vecino en D , D debe tener dos vértices en diferentes copias de H . (b) Dado que cualquier vértice en D con grado igual a n' necesita un no vecino en D , D debe contener un vértice en una copia de H que no esté conectado a un vértice de grado n' . (c) Si $\Delta(G') < n' - 1$, entonces $V(G')$ es un GTDS. ■

4.2.1. Formulación ILP

En [69] y [95], proponen una formulación ILP para el problema del conjunto dominante y el conjunto dominante global, respectivamente. Siguiendo estas ideas, se propone la siguiente formulación ILP para el problema que se está estudiando (GTDS) en este capítulo.

Dado un grafo $G = (V, E)$ de orden n , su matriz de adyacencia $A = (a_{ij})_{n \times n}$ se define de la siguiente manera: $a_{ij} = a_{ji} = 1$ si $\{i, j\} \in E(G)$, y $a_{ij} = a_{ji} = 0$ en caso contrario.

La matriz de adyacencia de $\overline{G}$, $B = (b_{ij})_{n \times n}$, se define de la siguiente manera: $b_{ij} = b_{ji} = 1$ si $a_{ij} = 0$, y $b_{ij} = b_{ji} = 0$ si $a_{ij} = 1$ o $i = j$.

Dado un GTDS D de cardinalidad $\gamma_t^g(G)$ e $i \in V(G)$, se definen las variables de decisión de la siguiente manera:

$$x_i = \begin{cases} 1, & \text{if } i \in D \\ 0, & \text{otherwise,} \end{cases} \quad (4.1)$$

El *problema del conjunto dominante global total* se puede formular de la siguiente manera:

$$\text{mín } \sum_{i=1}^n x_i, \quad (4.2)$$

$$\text{subject to : } \sum_{j=1}^n a_{ij}x_j \geq 1, \quad \forall i \in V(G), \quad (4.3)$$

$$\sum_{j=1}^n b_{ij}x_j \geq 1, \quad \forall i \in V(\overline{G}), \quad (4.4)$$

$$\sum_{i=1}^n x_i \geq 4. \quad (4.5)$$

$$x_i \in \{0, 1\}, \quad (4.6)$$

Aquí se desea minimizar la función objetivo (4.2). Se tienen $2n$ restricciones de tipo 4.3 y 4.4, garantizando que para cada $i \in V(G)$ y cada $j \in V(\overline{G})$, al menos uno de los vértices en D sea adyacente o no adyacente a j . Note que si se quiere encontrar un *conjunto dominante total mínimo*, basta con eliminar el conjunto de restricciones (4.4).

Si se conocen las propiedades estructurales del grafo, se pueden utilizar las cotas inferiores y superiores propuestas en la Sección 4.2 para agregar nuevas restricciones a la formulación ILP; acotando así, la región factible y el número de soluciones factibles que deben ser enumeradas por el método de solución empleado.

Si G es un grafo con $\text{diam}(G) = 2$, por la Proposición 4.8,

$$\sum_{i=1}^n x_i \leq \delta(G) + 3. \quad (4.7)$$

Si G es un grafo libre de triángulos o regular con $\text{diam}(G) = 2$, por las Proposiciones 4.5 y 4.10,

$$\sum_{i=1}^n x_i \leq \delta(G) + 2. \quad (4.8)$$

4.3. Algoritmos Heurísticos

4.3.1. Heurística GTAD

La primera heurística GTAD es un algoritmo voraz que trabaja en a lo sumo $n-4$ iteraciones. Se necesitan algunas definiciones para describir esta heurística. Denotemos por S_h al GTDS formado por la heurística en la iteración h . El conjunto de vértices dominados de forma total y global en la iteración h se denota por D_h , este conjunto de vértices está formado por los vértices que tienen al menos un vecino y un no vecino en S_h y no están en S_h . Por otro lado, A_h denota el conjunto de vértices que no tienen vecinos en S_h en la iteración h , es decir, $N(A_h) \cap S_h = \emptyset$. Finalmente, B_h denota el conjunto de vértices tales que para cada $v \in B_h$ y $u \in S_h$, existe la arista $\{u, v\} \in E$ en la iteración h .

En la iteración h , se define el *grado activo global total* del vértice $v \in D_h$ en los conjuntos A_h y B_h , tal que $N(v) \cap S_h \neq \emptyset$ y $|S_h \setminus N(v)| \geq 1$, como el número de vértices vecinos en A_h , no vecinos en B_h y el propio vértice. Denotemos el *grado activo global total* por $GTAD_{A_h, B_h}(v)$. Note que

$$GTAD_{A_h, B_h}(v) = |N_{A_{h-1}}[v] \cup \overline{N}_{B_{h-1}}[v]|.$$

Ahora, se describe la heurística en detalle. Una cota bien conocida en la dominación global total es que $\gamma_t^g(G) \geq 4$ para todo grafo G . Esto lleva a la idea en la primera parte de la heurística, que se basa en elegir cuatro vértices u, v, q y w de tal manera que la cardinalidad del conjunto $(N[u] \cup N[v] \cup N[q] \cup N[w]) \setminus (N[u] \cap N[v] \cap N[q] \cap N[w])$ sea lo más grande posible, donde los vértices u, v, q y w tienen un vecino y un no vecino en el conjunto $\{u, v, q, w\}$. Así, inicialmente $S_0 = \{u, v, q, w\}$, $D_0 = (N[u] \cup N[v] \cup N[q] \cup N[w]) \setminus (N[u] \cap N[v] \cap N[q] \cap N[w])$, $A_0 = V \setminus (N[u] \cup N[v] \cup N[q] \cup N[w])$ y $B_0 = V \setminus (S_0 \cup D_0 \cup A_0)$. Nótese que estos son conjuntos disjuntos y $A_0 \cup B_0 \cup C_0 \cup D_0 = V$.

Observación 4.2. *Los conjuntos de vértices A_h , B_h , D_h y S_h son disjuntos y $A_h \cup B_h \cup D_h \cup S_h = V$.*

Lema 4.8. *Si $A_0 = B_0 = \emptyset$ entonces*

$$(N[u] \cup N[v] \cup N[q] \cup N[w]) \setminus (N[u] \cap N[v] \cap N[q] \cap N[w]) = V \setminus \{u, v, q, w\} \quad (4.9)$$

y $S_0 = \{u, v, q, w\}$ es una solución óptima.

Demostración. La Ecuación 4.9 se sigue inmediatamente de la condición $A_0 = B_0 = \emptyset$ y del hecho de que $|S_0| = 4$. Por la Ecuación 4.9, el conjunto de vértices $N[u] \cap N[v] \cap N[q] \cap N[w]$ es vacío. Esto implica que S_0 es un GTDS. Alternativamente, $A_0 = B_0 = \emptyset$ implica que $D_0 = V \setminus S_0$ según la Observación 4.2. Entonces, S_0 es mínimo ya que un GTDS no puede contener menos de 4 vértices. ■

Los conjuntos iniciales se actualizan de manera iterativa en cada iteración $h > 0$, resultando en los conjuntos actuales de vértices A_h , B_h , D_h y S_h , donde S_h es un GTDS parcial. Cada iteración $h > 0$ comienza si $A_h \neq \emptyset$ o $B_h \neq \emptyset$. En cada iteración se selecciona un vértice $v_h \in D_{h-1}$, tal que $N_{D_{h-1}}(v_h) \neq S_{h-1}$ y que tenga el $GTAD_{A_h, B_h}(v)$ máximo en A_{h-1} y B_{h-1} .

Entonces, los conjuntos se modifican de la siguiente manera: el vértice v_h se mueve al conjunto S_{h-1} , resultando en el conjunto S_h . Así, los conjuntos A_h , B_h y D_h se actualizan correspondientemente. Cuando el algoritmo se detiene, el conjunto S_h es un GTDS para el valor actual de h . Un pseudo-código del algoritmo se muestra a continuación (ver Algoritmo 18).

Teorema 4.2. *La heurística GTAD se ejecuta en un máximo de $n - |D_0| - 4$ iteraciones y su complejidad temporal es $O(n^5)$.*

Demostración. Primero, recordemos que el número total de iteraciones en el bucle externo de la heurística GTAD está acotado por $|A_0| + |B_0| = n - |D_0| - 4$. En la iteración $h = 0$, para cada conjunto de cuatro vértices $\{u, v, q, w\}$ en $V(G)$, el conjunto $(N[u] \cup N[v] \cup N[q] \cup N[w]) \setminus (N[u] \cap N[v] \cap N[q] \cap N[w])$ se calcula sumando todas las entradas 1 en las filas correspondientes a los vértices $\{u, v, q, w\}$ si las cuatro entradas no son simultáneamente 1. Dado que cada fila contiene n entradas y hay $\frac{n(n-1)(n-2)(n-3)}{24}$ combinaciones de vértices a

verificar, el cálculo de $(N[u] \cup N[v] \cup N[q] \cup N[w]) \setminus (N[u] \cap N[v] \cap N[q] \cap N[w])$ para todos los conjuntos de vértices $\{u, v, q, w\}$ en $V(G)$ y la selección del máximo toma un tiempo $O(n^5)$. Por la Observación 4.2, $|A_0| + |B_0| < n$. Por lo tanto, el número total de iteraciones en el bucle externo está acotado por n . En cada iteración $h \geq 1$, el $GTAD_{A_h, B_h}(v)$ se calcula para cada vértice en D_{h-1} contando todas las entradas 1 y 0 en la fila correspondiente a ese vértice, tal que las entradas 1 pertenecen a A_h , y las entradas 0 pertenecen a B_h . Dado que cada fila contiene n entradas y hay menos de n filas por verificar, el cálculo de $GTAD_{A_h, B_h}(v)$ para todos los elementos en D_{h-1} y la selección del máximo toma un tiempo $O(n^2)$. Por lo tanto, la heurística GTAD encuentra un GTDS S en un tiempo $O(n^5)$. ■

Algorithm 18 Heuristic GTAD

Input: A graph $G(V, E)$.
Output: A global total dominating set S_h .
 $h := 0$;
 $\{u, v, q, w\} :=$ four vertex with the largest possible cardinality of the set
 $(N[u] \cup N[v] \cup N[q] \cup N[w]) \setminus (N[u] \cap N[v] \cap N[q] \cap N[w])$,
where the vertices u, v, q and w have a neighbor and a non-
neighbor in the set $\{u, v, q, w\}$;
 $S_0 := \{u, v, q, w\}$;
 $D_0 := (N[u] \cup N[v] \cup N[q] \cup N[w]) \setminus (N[u] \cap N[v] \cap N[q] \cap N[w])$;
 $A_0 := V \setminus (N[u] \cup N[v] \cup N[q] \cup N[w])$;
 $B_0 := V \setminus (S_0 \cup D_0 \cup A_0)$;
{ iterative step }
while $A_h \neq \emptyset \vee B_h \neq \emptyset$ **do**
 $h := h + 1$;
 $v_h :=$ a vertex $v \in D_{h-1}$ with the maximum $GTAD_{A_{h-1}, B_{h-1}}(v)$ and $N_{D_{h-1}}(v) \neq S_{h-1}$;
 $S_h := S_{h-1} \cup \{v_h\}$;
 $D_h := C_{h-1} \cup N_{A_{h-1}}(v_h) \cup \overline{N}_{B_{h-1}}(v_h)$;
 $A_h := A_{h-1} \setminus N_{A_{h-1}}(v_h)$;
 $B_h := B_{h-1} \setminus \overline{N}_{B_{h-1}}(v_h)$;
end while

Teorema 4.3. *La heurística GTAD encuentra una solución óptima si se detiene en la iteración h con $|S_h| \leq 5$, para $h = 0, 1$.*

Demostración. Supongamos que $|S_h| = 4$, lo que significa que $h = 0$ y el conjunto $S_0 = \{u, v, q, w\}$. Dado que el algoritmo terminó con el conjunto S_0 , debe cumplirse que $A_0 = B_0 = \emptyset$, lo que lleva a la ecuación 4.9, y que S_0 es un GTDS (Lema 4.8). Si ahora $|S_h| = 5$ ($h = 1$), entonces no existen cuatro vértices $\{u, v, q, w\}$ que satisfagan la ecuación 4.9 (por construcción). Por lo tanto, $\gamma_t^g(G) \geq 5$. Al mismo tiempo, dado que el algoritmo

se detuvo en la iteración 1, $A_1 = B_1 = \emptyset$, y por el Lema 4.2, $D_1 = V \setminus S_1$. Así, S_1 es una solución factible, y la proposición queda demostrada. ■

Por las Proposiciones 4.5, 4.8, 4.10, 4.12 y el Teorema 4.3 se obtiene el siguiente resultado.

Proposición 4.18. *Si G es un grafo con $\text{diam}(G) = 2$, y cumple una de las siguientes condiciones:*

- *es libre de triángulos y $\delta(G) \leq 3$;*
- *$\delta(G) \leq 2$;*
- *$\Delta(G) \leq 3$;*
- *es planar y libre de triángulos.*

Entonces $\gamma_t^g(G) \leq 5$ y la heurística GTAD encuentra una solución óptima.

En este punto, se puede decir que un subconjunto $S \subseteq V$ es un GTDS del grafo G , cuando para cada vértice $v \in V(G)$ hay al menos un vecino y un no vecino de v en S . Siguiendo esta idea, en la siguiente sección se propone otra heurística para obtener un GTDS.

4.3.2. Heurística GTD

Ahora, se presenta una segunda heurística para aproximar un GTDS. Esta heurística sigue el enfoque de aproximarse iterativamente a un conjunto dominante total tanto en el grafo G como en su complemento. Los conjuntos de vértices S_h , D_h , A_h y B_h se definen como en la heurística GTAD (ver Algoritmo 18). A continuación, se proporciona una descripción detallada de esta segunda heurística.

Primero, se seleccionan dos vértices $\{u, v\}$ donde u es el vértice con el mayor grado en $V(G)$ y v es su vecino con el grado más bajo. Luego, seleccionan otros dos vértices $\{q, w\}$, de manera que q es el vértice con el mayor número de vecinos no dominados por $\{u, v\}$, w es su vecino con el grado más bajo, y $\{u, v\} \not\subseteq (N(q) \cap N(w))$. Así, inicialmente $S_0 = \{u, v, q, w\}$, $D_0 = (N[u] \cup N[v] \cup N[q] \cup N[w]) \setminus (N[u] \cap N[v] \cap N[q] \cap N[w])$, $A_0 = V \setminus (N[u] \cup N[v] \cup N[q] \cup N[w])$ y $B_0 = V \setminus (S_0 \cup D_0 \cup A_0)$. Nótese que la Observación 4.2 y el Lema 4.8 también son válidos

en este caso.

Los conjuntos iniciales se actualizan iterativamente en cada iteración $h > 0$, lo que resulta en los conjuntos actuales de vértices S_h , D_h , A_h y B_h , donde S_h es el GTDS parcial. Cada iteración $h > 0$ comienza si $A_h \neq \emptyset$ o $B_h \neq \emptyset$. Ahora, en cada iteración, hay dos partes. Primero, se elige un vértice $v \in D_{h-1}$ tal que $N_{V \setminus N[S_{h-1}]}[v]$ sea lo más grande posible y $N_{S_{h-1}}(v) \neq \emptyset$. Luego, se añade v a S_h , es decir, $S_h = S_{h-1} \cup \{v\}$. Nótese que v se elige para maximizar el número de vértices adyacentes entre aquellos que aún no tienen vecinos en S_{h-1} . En este punto, los conjuntos A_h , B_h y D_h se actualizan correspondientemente.

La segunda parte de cada iteración se ejecuta si el conjunto S_h aún no es un GTDS, es decir, si $A_h \neq \emptyset$ o $B_h \neq \emptyset$. Por lo tanto, se elige un vértice $u \in D_h$ tal que $\overline{N}_{V \setminus \overline{N}[S_h]}[u]$ sea lo más grande posible y $|S_h \setminus N(u)| \geq 1$. Luego, se asigna u a S_h , es decir, $S_h = S_h \cup \{u\}$. Nótese que el vértice u es aquel que tiene más vértices no adyacentes entre los que no tienen no-vecinos en S_h . En este punto, los conjuntos A_h , B_h y D_h se actualizan correspondientemente.

Finalmente, en S_h , se obtiene un GTDS del grafo G . Un pseudo-código de la heurística se muestra a continuación (ver Algoritmo 19).

Teorema 4.4. *La heurística GTD se ejecuta como máximo en $n - |D_0| - 4$ iteraciones, y su complejidad temporal es $O(n^3)$.*

Demostración. Primero, recordemos que el número total de iteraciones en el bucle externo de la Heurística GTD está acotado por $|A_0| + |B_0| = n - |D_0| - 4$. Siguiendo la misma idea que en la demostración del Teorema 4.2, es fácil ver que en la iteración $h = 0$, cada vértice u , v , q y w se determina en un tiempo $O(n^2)$.

Por la Observación 4.2, $|A_0| + |B_0| < n$. Por lo tanto, el número total de iteraciones en el bucle externo está acotado por n . En cada iteración $h \geq 1$, para cada vértice $v \in D_{h-1}$, se calculan $|N_{V \setminus N[S_{h-1}]}[v]|$ y $|N_{S_{h-1}}(v)|$ contando todas las entradas 1 en la fila correspondiente a ese vértice, donde las entradas 1 pertenecen a A_h y S_{h-1} , respectivamente. Para cada vértice $u \in D_h$, se calculan $|\overline{N}_{V \setminus \overline{N}[S_h]}[u]|$ y $|S_h \setminus N(u)|$ contando todas las entradas 0 en la fila correspondiente a ese vértice, donde las entradas 0 pertenecen a B_h y S_h , respectivamente. Dado que cada fila contiene n entradas y hay menos de n filas por verificar, el cálculo de

$|NV \setminus N[S_{h-1}][v]|$, $|N_{S_{h-1}}(v)|$, $|\overline{N}_{V \setminus \overline{N}[S_h]}[u]|$ y $|S_h \setminus N(u)|$ para todos los elementos en D_h y la selección del máximo toma un tiempo $O(n^2)$.

Por lo tanto, la heurística GTD encuentra un GTDS S en un tiempo $O(n^3)$. ■

Algorithm 19 Heuristic H2

Input: A graph $G(V, E)$.
Output: A global total dominating set S_h .
 $h := 0$;
 $\{u, v\} := u$ is the vertex with the highest degree in $V(G)$ and v is its neighbor
with the lowest degree;
 $\{q, w\} := q$ is the vertex with the highest number of neighbors not dominated
by $\{u, v\}$, w is its neighbor with the lowest degree, and
 $\{u, v\} \not\subseteq (N(q) \cap N(w))$.
 $S_0 := \{u, v, q, w\}$;
 $D_0 := (N[u] \cup N[v] \cup N[q] \cup N[w]) \setminus (N[u] \cap N[v] \cap N[q] \cap N[w])$;
 $A_0 := V \setminus (N[u] \cup N[v] \cup N[q] \cup N[w])$;
 $B_0 := V \setminus (S_0 \cup D_0 \cup A_0)$; { iterative step }
while $A_h \neq \emptyset \vee B_h \neq \emptyset$ **do**
 $h := h + 1$;
 $v_h :=$ a vertex $v \in D_{h-1}$ such that $N_{V \setminus N[S_{h-1}]}[v]$ be maximum
 and $N_{S_{h-1}}(v) \neq \emptyset$;
 $S_h := S_{h-1} \cup \{v\}$;
 $D_h = N(S_h) \cap \overline{N}(S_h)$;
 $A_h := V(G) \setminus N(S_h)$;
 $B_h := B_{h-1} \setminus \overline{N}(S_h)$;
 if $A_h \neq \emptyset \vee B_h \neq \emptyset$ **then**
 $u_h :=$ a vertex $u \in D_h$ such that $\overline{N}_{V \setminus \overline{N}[S_h]}[u]$ be maximum
 and $|S_h \setminus N(u)| \geq 1$;
 $S_h := S_h \cup \{u\}$;
 $D_h = N(S_h) \cap \overline{N}(S_h)$;
 $A_h := V(G) \setminus N(S_h)$;
 $B_h := B_{h-1} \setminus \overline{N}(S_h)$;
 end if
end while

Proposición 4.19. *La heurística GTD encuentra una solución óptima si se detiene en la iteración $h = 0$ con $|S_h| = 4$.*

Demostración. Primero observe que $\gamma_t^g(G) \geq 4$. Supongamos que $|S_h| = 4$, lo que implica que $h = 0$ y el conjunto $S_0 = \{u, v, q, w\}$. Dado que el algoritmo se detuvo con el conjunto S_0 , debe cumplirse que $A_0 = B_0 = \emptyset$, lo que lleva a la ecuación (4.9) y establece que S_0 es un GTDS (Lema 4.8). Por lo tanto, S_0 es una solución óptima. ■

Cabe destacar que, a diferencia de la heurística GTAD, en este caso no se puede garantizar que si $|S_h| = 5$, entonces S_h sea una solución factible óptima. En la Figura 4.7, se muestra un grafo

con $\gamma_t^g(G) = 4$ y $|S_h| = 5$. En la iteración $h = 0$, la heurística GTD elige $S_0 = \{u, v, q, w\}$. Luego, $D_0 = \{v_1, 1, 2, 3, 6, 7, 8\}$, $A_0 = \{4, 5\}$ y $B_0 = \emptyset$. En la iteración $h = 1$, se elige el vértice v_1 . Entonces, $S_1 = \{u, v, q, w, v_1\}$, y S_1 es un GTDS de cardinalidad 5.

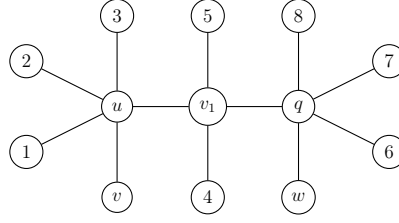


Figura 4.7: Grafo con $\gamma_t^g(G) = 4$ y $|S_h| = 5$

Sea G un grafo conexo de orden $n > 2$. Además, sea Q_i , $i = \overline{1, n}$, un grafo conexo de orden $k_i > 2$ que contiene el grafo estrella S_{k_i-1} como un subgrafo. Sea, además, q_i el centro del grafo S_{k_i-1} (un vértice con grado máximo $k_i - 1$ en ese grafo). $G' = G \circ_{q_i} Q_i$ es el grafo compuesto por estos $n + 1$ grafos, en el cual cada vértice $v_i \in V(G)$ está asociado con el vértice q_i del grafo Q_i . La Figura 4.8 ilustra la estructura de tal grafo compuesto.

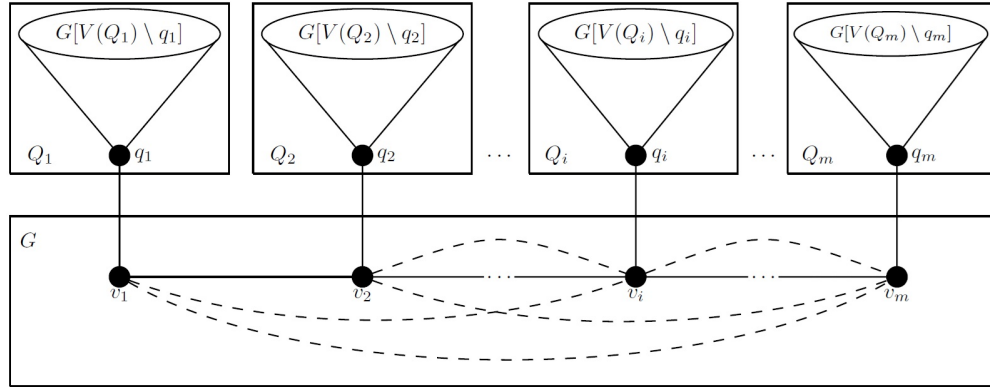


Figura 4.8: Grafo $G' = G \circ_{q_i} Q_i$

Proposición 4.20. Sea $G' = G \circ_{v_i} Q_i$ un grafo conexo, donde G tiene orden $n \geq 2$. Entonces, $\gamma_t^g(G') = 2n$.

Demostración. Como se puede ver fácilmente en la Figura 4.8, $n \geq 2$ implica que cualquier par de vértices de tipo $\{q_i, v_i\}$ forma un conjunto dominante total del grafo $\overline{G'}$, mientras que $\{q_i, v_i | i = \overline{1, n}\}$ forma un conjunto dominante global total del grafo G' , dado que $\{q_i, v_i\}$ es un $\gamma_t(G[V(Q_i) \cup v_i])$ -conjunto y cada vértice $v \in V(G)$ tiene un vecino y un no vecino de

tipo q_i . Cabe destacar que $\gamma_t(Q_i) = 2$, para $i = \overline{1, n}$. Por lo tanto, $\{q_i, v_i | i = \overline{1, n}\}$ es un $\gamma_t^g(G')$ -conjunto, y $\gamma_t^g(G') = 2n$. ■

Proposición 4.21. *Sea $G' = G \circ_{v_i} Q_i$ un grafo conexo, donde G tiene orden $n \geq 2$ y Q_i tiene orden $k_i \geq 2$, para $i = \overline{1, n}$, de modo que $k_1 \geq k_2 \geq \dots \geq k_i \geq \dots \geq k_{n-1} \geq k_n > \Delta(G)$, $\delta(Q_1) > N_{V(G)}(v_1)$, y $\delta(Q_2) > N_{V(G)}(v_2)$, donde $v_1, v_2 \in V(G)$. Entonces, la heurística GTD encuentra una solución óptima para el grafo G' .*

Demostración. Claramente, $\{q_i, v_i\}$ es un conjunto dominante total del grafo $G[V(Q_i) \cup v_i]$, $\gamma_t(Q_i) = 2$, y el $\gamma_t^g(G')$ -conjunto es $\cup_{i=1}^n \{q_i, v_i\}$ (ver Proposición 4.20 y Figura 4.8).

Dado que $k_1 \geq k_2 \geq \dots \geq k_i \geq \dots \geq k_{n-1} \geq k_n > \Delta(G)$, $\delta(Q_1) > N_{V(G)}(v_1)$, y $\delta(Q_2) > N_{V(G)}(v_2)$, la heurística GTD en la iteración $h = 0$ seleccionará los vértices $\{q_1, v_1, q_2, v_2\}$, formando el conjunto S_0 , que es un conjunto dominante total del grafo $\overline{G'}$.

Note que en cada iteración impar $h \geq 1$, los únicos vértices en el conjunto D_{h-1} que satisfacen $N_{V \setminus N[S_{h-1}]}[v] \neq \emptyset$ y $N_{S_{h-1}}(v) \neq \emptyset$ son de tipo v_i . Cuando h es par, existe un vértice de tipo q_i en el conjunto D_{h-1} que tiene el mayor número de vecinos no dominados en G' .

En cada iteración siguiente $h = 1, \dots, 2n - 4$, la heurística GTD añade un vértice de tipo v_i o q_i a la solución actual S_h y devuelve un GTDS $S_{2n-4} = \{q_i, v_i | i = \overline{1, n}\}$, que es óptimo según la Proposición 4.20. ■

4.3.3. Heurística GTD-AD

La tercera heurística combina la iteración 0 de la heurística GTD y la parte iterativa de la heurística GTAD. Los conjuntos de vértices S_h , D_h , A_h y B_h , así como la iteración inicial 0 con los vértices u , v , q y w , se definen como en la heurística GTD. Así, inicialmente $S_0 = \{u, v, q, w\}$, $D_0 = (N[u] \cup N[v] \cup N[q] \cup N[w]) \setminus (N[u] \cap N[v] \cap N[q] \cap N[w])$, $A_0 = V \setminus (N[u] \cup N[v] \cup N[q] \cup N[w])$ y $B_0 = V \setminus (S_0 \cup D_0 \cup A_0)$. Note que la Observación 4.2 y el Lema 4.8 también son válidos en este caso.

Los conjuntos iniciales se actualizan iterativamente en cada iteración $h > 0$, como en la heurística GTAD, resultando en los conjuntos actuales de vértices A_h , B_h , D_h y S_h , donde S_h es el GTDS parcial. Si se satisface la condición $A_h = B_h = \emptyset$ (ver Lema 4.8), entonces el

algoritmo se detiene en la iteración $h \geq 1$.

Para cada iteración $h > 0$, se selecciona un vértice v_h de D_{h-1} tal que $N_{D_{h-1}}(v_h) \neq S_{h-1}$ y tenga el máximo *grado activo global total* en A_{h-1} y B_{h-1} . Por lo tanto, los conjuntos se modifican de la siguiente manera: el vértice v_h se mueve al conjunto S_{h-1} , resultando en el conjunto S_h . Así, los conjuntos A_h , B_h y D_h se actualizan correspondientemente. Cuando el algoritmo se detiene, el conjunto S_h es un GTDS para el h actual. A continuación se muestra un pseudo-código del algoritmo (ver Algoritmo 20).

Algorithm 20 Heuristic GTD-AD

Input: A graph $G(V, E)$.

Output: A global total dominating set S_h .

$h := 0$;

$\{u, v\} := u$ is the vertex with the highest degree in $V(G)$ and v is its neighbor
with the lowest degree;

$\{q, w\} := q$ is the vertex with the highest number of neighbors not dominated
by $\{u, v\}$, w is its neighbor with the lowest degree, and
 $\{u, v\} \not\subseteq (N(q) \cap N(w))$.

$S_0 := \{u, v, q, w\}$;

$D_0 := (N[u] \cup N[v] \cup N[q] \cup N[w]) \setminus (N[u] \cap N[v] \cap N[q] \cap N[w])$;

$A_0 := V \setminus (N[u] \cup N[v] \cup N[q] \cup N[w])$;

$B_0 := V \setminus (S_0 \cup D_0 \cup A_0)$; { iterative step }

while $A_h \neq \emptyset \vee B_h \neq \emptyset$ **do**

$h := h + 1$;

$v_h :=$ a vertex $v \in D_{h-1}$ with the maximum $GTAD_{A_{h-1}, B_{h-1}}(v)$ and $N_{D_{h-1}}(v) \neq S_{h-1}$;

$S_h := S_{h-1} \cup \{v_h\}$;

$D_h := C_{h-1} \cup N_{A_{h-1}}(v_h) \cup \overline{N}_{B_{h-1}}(v_h)$;

$A_h := A_{h-1} \setminus N_{A_{h-1}}(v_h)$;

$B_h := B_{h-1} \setminus \overline{N}_{B_{h-1}}(v_h)$;

end while

Teorema 4.5. *La heurística GTD-AD se ejecuta como máximo en $n - |D_0| - 4$ iteraciones, y su complejidad temporal es $O(n^3)$.*

Demostración. Primero, recordemos que el número total de iteraciones en el bucle externo está acotado por $|A_0| + |B_0| = n - |D_0| - 4$. Siguiendo la misma idea que en la demostración de los Teoremas 4.2 y 4.4, es fácil ver que en la iteración $h = 0$, se determinan cada uno de los vértices u , v , q y w en un tiempo $O(n^2)$.

Por la Observación 4.2, $|A_0| + |B_0| < n$. En cada iteración $h \geq 1$, se calcula el grado activo global total para cada vértice en D_{h-1} como en la demostración del Teorema 4.2. Dado que cada fila contiene n entradas y hay menos de n filas por verificar, el cálculo del grado activo global total para todos los elementos en D_{h-1} y la selección del máximo toma un tiempo $O(n^2)$.

Por lo tanto, la heurística GTD-AD encuentra un GTDS S en tiempo $O(n^3)$. ■

Note que, por la construcción del algoritmo, la heurística GTD-AD también satisface la condición de optimalidad propuesta en la Proposición 4.19.

4.3.4. Procedimiento de purificación

En [83], [95] y [94], se demostró que los procedimientos de purificación son útiles para reducir el número de vértices en una solución factible no óptima del problema de dominación clásico y de dominación global. A continuación se presentan dos definiciones que se utilizan en el procedimiento de purificación propuesto. La *vecindad privada total* $tpr(v, S, G)$ del vértice $v \in S \subseteq V(G)$ se define como

$$tpr(v, S, G) = \{u \in V \mid N(u) \cap S = \{v\}\}.$$

Un vértice en $tpr(v, S, G)$ se llama *vecino privado total* de v .

Note que el conjunto S^* formado de esta manera es un GTDS minimal del grafo G . De hecho, por construcción, la eliminación de cualquier vértice daría como resultado una solución que no domina total y globalmente a los vecinos privados de ese vértice. Siguiendo la misma idea que la prueba de los Teoremas 4.2 y 4.4, es fácil ver que el procedimiento *Purify* se puede implementar en tiempo $O(pn^2) = O(n^3)$.

Algorithm 21 Purify GTDS

Input: A graph $G(V, E)$ and a global total dominating set S_h .

Output: A global total dominating set S^* .

$S^* = \{v_1, \dots, v_p\}$; {vertices of S_h in the order they were added by the GTAD, GTD or GTD-AD heuristic.}

$h := p$;

{ iterative step }

while $h > 0$ **do**

if $tpn(v_h, S^*, G) = \emptyset \wedge tpn(v_h, S^*, \overline{G}) = \emptyset$ **then**

$S^* := S^* \setminus \{v_h\}$; {we purify vertex v_h }

end if

$h := h - 1$;

end while

4.4. Resultados experimentales

En esta sección, se describen los experimentos computacionales para GTDS. Se implementó la formulación ILP en C++ utilizando la biblioteca CPLEX, y las heurísticas en C++ usando el sistema operativo Windows 10 para una arquitectura de 64 bits. Los algoritmos se probaron en 1130 instancias de referencia tomadas de [92].

Dado que no existían previamente instancias de referencia para el problema de dominación estudiado, se utilizaron los valores óptimos obtenidos por CPLEX, así como los límites superiores e inferiores conocidos, para analizar el rendimiento de las heurísticas propuestas. Se denotó como ILP^L la cota inferior obtenida por la formulación ILP dentro del límite de tiempo de 20 minutos. De manera similar, U representa el valor de la mejor solución proporcionada por una de las heurísticas.

La Tabla 4.1 muestra los valores de $\gamma_t(G)$ y $\gamma_t^g(G)$ para 30 de las 1050 instancias analizadas con CPLEX, así como el valor de U y algunas propiedades estructurales de estas instancias. Los valores de $\gamma_t(G)$, $\gamma_t^g(G)$ y U mostrados en la Tabla 4.1 pueden compararse con las cotas superiores e inferiores propuestas en la Sección 4.2 y con las mejores soluciones proporcionadas por las heurísticas. Esta comparación permite evaluar la precisión y efectividad de las cotas teóricas y las heurísticas al estimar estos índices para las instancias analizadas.

La Figura 4.9 presenta una comparación entre las mejores soluciones producidas por los algoritmos heurísticos y los valores óptimos obtenidos por CPLEX para instancias con hasta

8000 vértices. En esta figura, los nodos azules denotan soluciones óptimas, mientras que los cuadrados rojos representan las mejores soluciones encontradas por las heurísticas. Las heurísticas lograron encontrar una solución óptima en aproximadamente el 41 % de las instancias. Para las instancias en las que no se alcanzó la solución óptima, el error absoluto promedio fue de alrededor de 1.58 vértices.

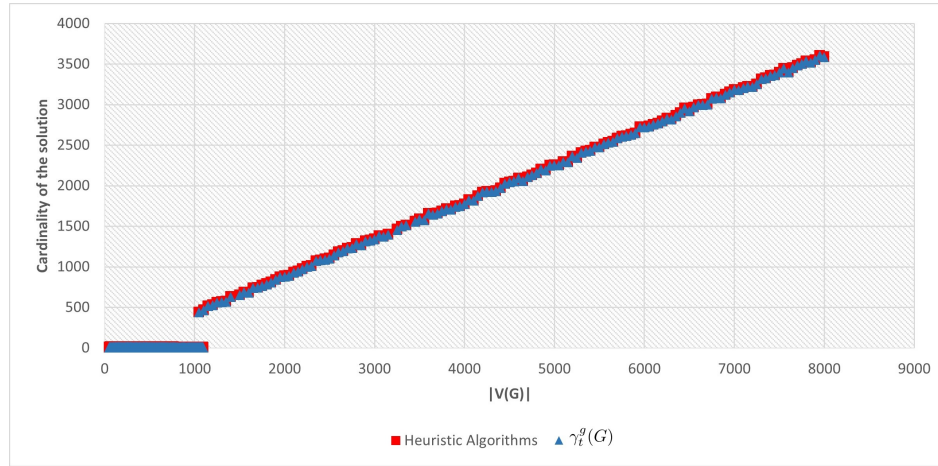
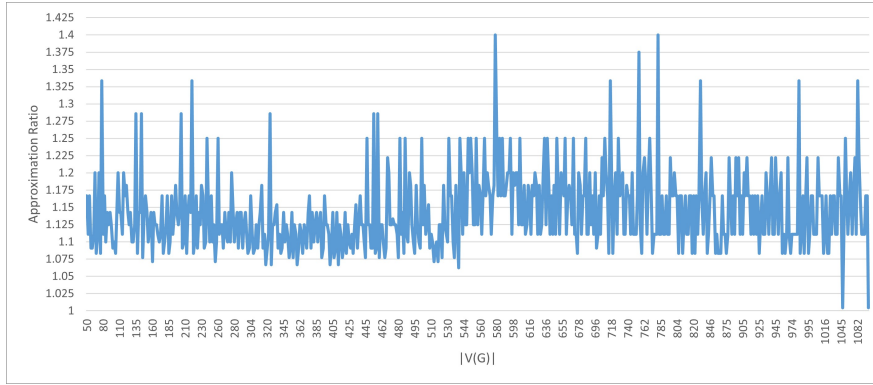


Figura 4.9: Resultados de la aproximación vs. $\gamma_t^g(G)$.

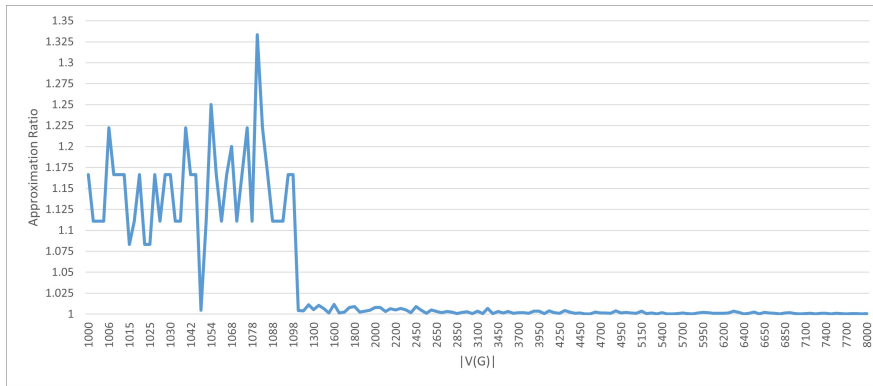
Grafo	$ V(G) $	$ E(G) $	$diam(G)$	$g(G)$	ILP $\gamma_t(G)$			ILP $\gamma_t^g(G)$			ILP^L			U	
					$ S $	Tiempo(s)	gap	$ S $	Tiempo(s)	gap	$ S $	Tiempo(s)	$ S $	Tiempo(s)	
BD0_21	150	3725	3	3	6	0.433868	0	6	0.565838	0	6	0.001727	7	0.001727	
BD3_26	175	6858	6	3	7	0.171801	0	7	0.185212	0	7	0.0034096	7	0.0034096	
BD1_31	200	4547	6	3	11	14.9226	0	11	15.0656	0	11	0.00301	11	0.00301	
BD0_41	250	10375	3	3	7	66.5806	0	7	97.6669	0	7	0.003799	7	0.003799	
BD3_46	275	16470	8	3	10	0.759432	0	10	0.446515	0	10	0.0054548	12	0.0054548	
BD3_56	325	24896	7	3	7	1.19912	0	7	0.773005	0	7	0.0077495	9	0.0077495	
BD1_61	350	18337	8	3	13	349.769	0	13	276.863	0	13	0.0089748	14	0.0089748	
BD0_66	375	23375	3	3	7	567.937	0	7	690.26	0	7	0.0062221	8	0.0062221	
BD3_76	425	41332	8	3	11	18.7068	0	11	17.6541	0	11	0.0113868	12	0.0113868	
BD6_176	450	87361	6	3	5	0.034316	0	5	0.0968375	0	5	0.0040491	5	0.0040491	
BD3_86	475	54183	6	3	8	30.3236	0	8	26.7823	0	8	0.0126022	9	0.0126022	
BD3_101	550	73046	6	3	8	48.4127	0	8	41.2481	0	8	0.0176894	10	0.0176894	
BD6_251	600	156934	6	3	5	0.0696031	0	5	0.151407	0	5	0.0074838	6	0.0074838	
BD3_121	650	102571	7	3	8	126.061	0	8	217.006	0	8	0.0276011	10	0.0276011	
BD6_288	674	171941	8	3	9	0.239355	0	9	0.253122	0	9	0.0163373	10	0.0163373	
BD3_136	725	127996	7	3	8	158.403	0	8	171.538	0	8	0.0273989	10	0.0273989	
BD6_326	750	214085	8	3	9	0.31592	0	9	0.343116	0	9	0.0344682	10	0.0344682	
BD6_351	800	281227	6	3	5	0.0980406	0	5	0.236529	0	5	0.0209552	6	0.0209552	
BD6_378	854	285299	6	3	6	0.316186	0	6	0.426589	0	6	0.035279	7	0.035279	
BD6_401	900	356878	6	3	6	0.564847	0	6	0.552009	0	6	0.0298821	6	0.0298821	
BD6_431	960	398942	8	3	9	0.366656	0	9	0.550262	0	9	0.0512926	9	0.0512926	
BD6_455	1008	398824	7	3	6	0.537497	0	6	0.682789	0	6	0.0604479	7	0.0604479	
BD6_483	1064	500420	6	3	6	0.562107	0	6	0.54335	0	6	0.0397546	6	0.0397546	
BD5_4	1200	1305	22	3	525	0.0395533	0	525	0.285345	0	525	0.700585	531	0.700585	
BD5_26	2300	2407	26	3	1009	0.0792524	0	1009	0.785574	0	1009	5.6417	1014	5.6417	
BD5_43	3150	3221	29	3	1398	0.117124	0	1398	1.56636	0	1398	13.1076	1399	13.1076	
BD5_65	4250	4299	28	3	1927	0.205809	0	1927	2.96783	0	1927	37.4256	1929	37.4256	
BD5_89	5450	5489	34	3	2472	0.334689	0	2472	4.86376	0	2472	77.5741	2472	77.5741	
BD5_107	6350	6491	29	3	2857	0.453513	0	2857	7.15439	0	2857	124.236	2863	124.236	
BD5_120	7000	7038	33	3	3186	0.554199	0	3186	8.40799	0	3186	178.857	3186	178.857	
BD5_130	7500	7535	31	3	3398	0.627284	0	3398	10.7075	0	3398	219.461	3398	219.461	
BD5_140	8000	8126	31	3	3586	0.757664	0	3586	14.912	0	3586	261.609	3589	261.609	
BD4_13	10450	27251021	—	—	12	1220.34	0.51802	11	1264.61	0.818182	1.999998	12	6.98681		
BD4_18	10700	28523632	—	—	14	1248.79	0.372537	14	1450.51	0.372533	8.784538	15	8.49546		

Tabla 4.1: Comparación de los valores de $\gamma_t(G)$ y $\gamma_t^g(G)$ con las soluciones heurísticas para 30 instancias analizadas con CPLEX

La Figura 4.10 muestra el radio de aproximación de las heurísticas en aquellas instancias donde no se encontró la solución óptima. Para instancias con menos de 1100 vértices, el radio de aproximación mínimo fue 1.0042, el máximo fue 1.4, y la media fue 1.1436. Para instancias con más de 1100 vértices, el radio de aproximación mínimo fue 1.0002, el máximo fue 1.0909, y la media fue 1.0073. Este resultado indica el rendimiento superior de las heurísticas en instancias con un mayor número de vértices, donde los algoritmos exactos no son prácticos. Los bajos radios de aproximación demuestran la alta calidad y fiabilidad de las heurísticas propuestas para proporcionar soluciones casi óptimas de manera eficiente. Estos hallazgos subrayan el papel crucial de las heurísticas en la resolución de problemas a gran escala que son computacionalmente inviables para los algoritmos exactos.



(a) Instancias con menos de 1100 vértices



(b) Instancias con más de 1000 vértices

Figura 4.10: Radio de aproximación

La Figura 4.11 muestra una comparación de los tiempos en los que CPLEX encontró $\gamma_t^g(G)$ y $\gamma_t(G)$, y el tiempo en el que la heurística encontró la mejor solución. En la literatura y en la Sección 4.2, se proponen condiciones bajo las cuales se puede determinar $\gamma_t^g(G)$ utilizando

$\gamma_t(G)$. Estas condiciones no son aplicables a todos los tipos de grafos, y al mismo tiempo, los resultados experimentales mostrados en la Figura 4.11 reflejan que, en la práctica, no siempre es eficiente encontrar primero un conjunto $\gamma_t(G)$ -conjunto para luego encontrar un $\gamma_t^g(G)$ -conjunto. En el 24.8 % de las instancias, CPLEX encontró $\gamma_t^g(G)$ en menos tiempo que $\gamma_t(G)$. Por otro lado, en las instancias donde la heurística encontró la solución óptima, es decir, un $\gamma_t^g(G)$ -conjunto, la heurística fue, en promedio, 12374 veces más rápida que CPLEX.

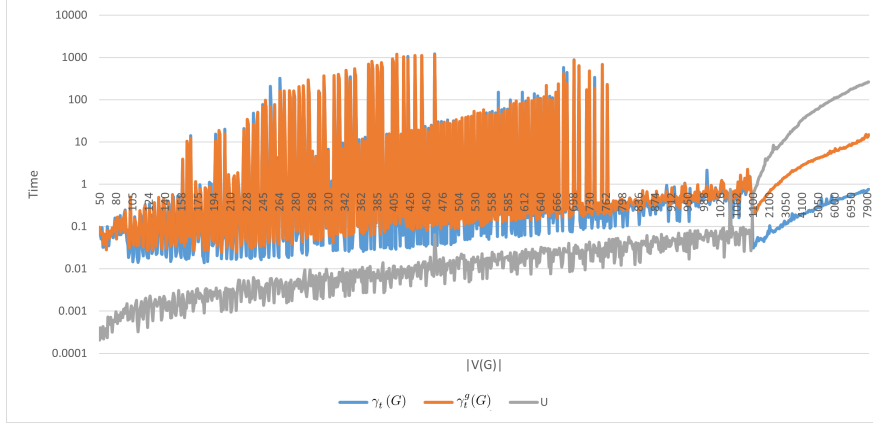


Figura 4.11: Tiempo Formulación ILP vs. Heurística (se aplicó escala logarítmica)

Los resultados anteriores reportados para las heurísticas incluyen la etapa de purificación. El proceso de purificación demostró ser efectivo en la práctica, como se ilustra en la Tabla 4.3. En esta tabla, $|S|$ y $|S^*|$ denotan el número de vértices en el GTDS antes y después de la purificación, respectivamente.

Los resultados obtenidos para las heurísticas se pueden resumir como sigue. Antes de la aplicación del procedimiento de purificación, las tres heurísticas obtuvieron soluciones con igual cardinalidad en el 2.59 % de las instancias analizadas. Para las instancias restantes, las heurísticas GTAD, GTD y GTD-AD obtuvieron las mejores soluciones en el 41.21 %, 50.49 % y 5.71 % de las instancias probadas, respectivamente (ver tabla 4.2).

	Instancias	%
GTAD	462	41.21
GTD	566	50.49
GTD-AD	64	5.71
Todas	29	2.59

Tabla 4.2: Análisis comparativo de las heurísticas antes de la purificación

Grafo	V	E	Heurística GTAD			Heurística GTD			Heurística GTD-AD			Purificada			ILP ^L		
			Tiempo(s)	S	%	Tiempo(s)	S	%	Tiempo(s)	S	%	Tiempo(s)	S	%	Tiempo(s)	S	%
BD0_91	500	31187	6088.13	11	0.0053	11	0.0054	11	0.0030	11	0.00	0.0128	11	0.0028	11	0.00	6.01
BD1_96	525	33228	6548.14	15	0.0056	14	0.0271	16	0.0039	15	6.25	0.0225	16	0.0073	15	6.25	9.03
BD0_100	545	49413	12458.8	9	0.0056	9	0.0240	11	0.0034	11	0.00	0.0150	10	0.0035	10	0.00	4.91
BD3_101	550	73046	—	—	—	—	0.0317	12	0.0046	10	16.67	0.0177	12	0.0051	11	8.33	8.00
BD3_111	600	84557	—	—	—	—	0.0516	15	0.0082	13	13.33	0.0236	15	0.0060	13	13.33	11.00
BD6_276	650	164146	—	—	—	—	0.0258	10	0.0058	7	30.00	0.0338	19	0.0143	7	63.16	6.00
BD3_131	700	116132	—	—	—	—	0.0611	15	0.0062	12	20.00	0.0316	15	0.0079	14	6.67	11.00
BD6_327	752	248140	—	—	—	—	0.0182	7	0.0072	6	14.29	0.0151	7	0.0053	6	14.29	6.00
BD3_151	800	152707	—	—	—	—	0.0838	15	0.0102	13	13.33	0.0413	15	0.0100	12	20.00	9.87
BD6_376	850	317920	—	—	—	—	0.0432	10	0.0071	6	40.00	0.0774	30	0.0151	6	80.00	6.00
BD3_171	900	198258	—	—	—	—	0.0876	12	0.0109	11	8.33	0.0455	12	0.0093	11	8.33	6.86
BD3_182	955	219280	—	—	—	—	0.1363	15	0.0160	13	13.33	0.0586	15	0.0095	14	6.67	9.18
BD3_191	1000	240857	—	—	—	—	0.1239	15	0.0134	14	6.67	0.0708	16	0.0131	14	12.50	9.08
BD5_10	1500	1598	—	—	—	—	1.3666	720	0.1242	653	9.31	3.2679	720	0.1354	653	9.31	652.00
BD5_20	2000	2128	—	—	—	—	3.7344	975	0.2305	887	9.03	7.9509	975	0.1947	887	9.03	880.00
BD5_41	3050	3064	—	—	—	—	11.6821	1492	0.3959	1381	7.44	50.5312	1492	0.3819	1381	7.44	1381.00
BD5_63	4150	4160	—	—	—	—	34.3193	2041	0.8292	1876	8.08	81.4496	2041	0.6999	1876	8.08	1876.00
BD5_82	5100	5139	—	—	—	—	63.0205	2501	1.1791	2294	8.28	152.9580	2501	1.0145	2294	8.28	2292.00
BD5_100	6000	6010	—	—	—	—	111.5620	2971	1.7376	2722	8.38	259.4440	2971	1.7651	2722	8.38	2722.00
BD5_110	6500	6586	—	—	—	—	135.6990	3168	1.8169	2925	7.67	327.7410	3168	1.5391	2925	7.67	2918.00
BD5_121	7050	7142	—	—	—	—	180.1830	3446	2.4496	3181	7.69	422.7810	3446	1.9864	3181	7.69	3180.00
BD5_131	7550	7557	—	—	—	—	221.3130	3738	2.5921	3446	7.81	527.7400	3738	2.2750	3446	7.81	3446.00
BD5_140	8000	8126	—	—	—	—	261.6090	3866	3.2614	3589	7.17	682.2990	3866	2.7646	3589	7.17	3586.00
BD4_3	9950	24703396	—	—	—	—	13.4991	15	0.9191	13	13.33	5.9653	15	0.9167	14	6.67	5.84
BD4_14	10500	27465482	—	—	—	—	15.1761	17	0.9967	15	11.76	8.1691	18	1.0769	16	11.11	8.78
BD4_24	11000	30148357	—	—	—	—	18.8644	18	1.2541	16	11.11	9.1008	18	1.1713	17	5.56	—
BD4_35	11550	33295796	—	—	—	—	18.0031	16	1.3649	14	12.50	8.5208	15	1.2841	14	6.67	—
BD4_44	12000	35943033	—	—	—	—	18.0301	15	1.2757	13	13.33	9.7955	16	1.3034	15	6.25	—
BD4_54	12500	38946982	—	—	—	—	22.9198	19	1.4388	17	10.53	13.1701	18	1.5308	17	5.56	—
BD4_65	13050	42455020	—	—	—	—	23.7640	18	1.4864	16	11.11	13.0467	19	1.5886	17	10.53	—
BD4_75	13550	45836296	—	—	—	—	24.6814	15	2.0193	13	13.33	12.4248	16	1.6505	15	6.25	—
BD4_86	14100	49635558	—	—	—	—	21.9406	14	1.8046	13	7.14	13.5461	16	1.9129	14	12.50	—
BD7_1	25000	249810756	—	—	—	—	38.8526	11	4.4324	8	27.27	115.8030	42	10.7639	9	78.57	—
BD7_2	25000	2499475	—	—	—	—	278.2380	328	2.9041	327	0.30	1006.1800	328	7.6087	327	0.30	—

Tabla 4.3: Resultados de las heurísticas GTAD, GTD y GTD-AD

Cuando se aplicó el procedimiento de purificación a las soluciones generadas por la heurística GTD, las soluciones purificadas fueron superiores a las construidas por las heurísticas GTAD y GTD-AD. En promedio, el procedimiento de purificación redujo el tamaño de las soluciones de las heurísticas GTAD, GTD y GTD-AD en un 9.40 %, 18.22 % y 27.64 %, respectivamente, ver Tabla 4.3. La Tabla 4.4 muestra el número de instancias en las que las heurísticas encontraron las mejores soluciones después de la purificación.

	Instancias	%
GTAD	294	26.23
GTD	617	55.04
GTD-AD	33	2.94
Todas	177	15.79

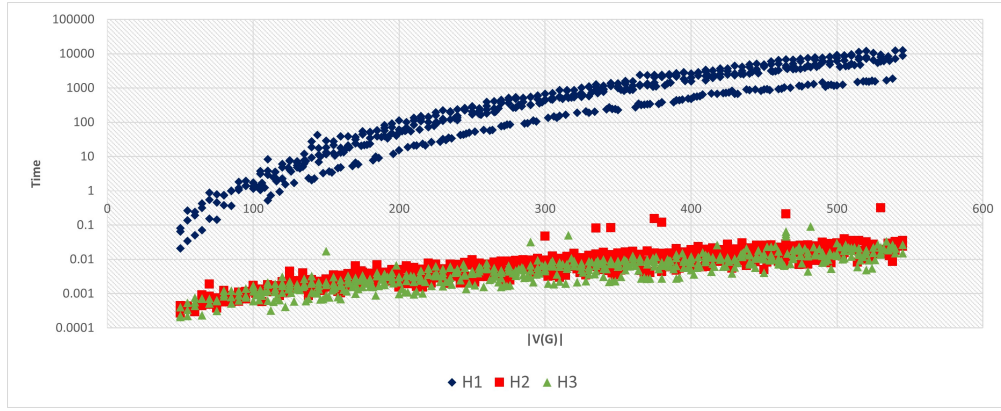
Tabla 4.4: Análisis comparativo de las heurísticas después de la purificación

La Tabla 4.5 muestra el número de soluciones factibles entregadas por las heurísticas GTAD, GTD y GTD-AD que fueron purificadas. A partir de esto, se puede entender que las heurísticas GTAD, GTD y GTD-AD encontraron soluciones factibles minimales en el 69.5 %, 9.72 % y 11.78 % de las instancias, respectivamente.

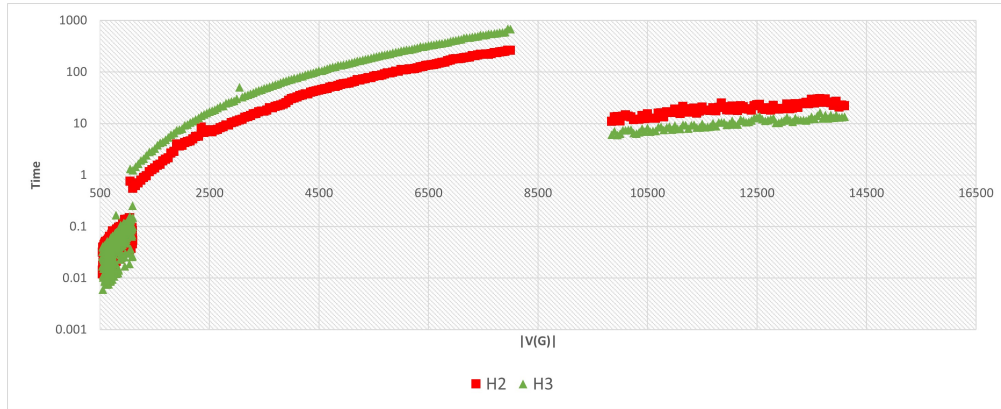
GTAD	GTD	GTD-AD
157	1012	989

Tabla 4.5: Instancias purificadas por heurística

En cuanto a los tiempos de ejecución, para las instancias de referencia más grandes con 25000 vértices, las heurísticas GTD y GTD-AD terminaron en menos de dos minutos. La Figura 4.12 muestra los tiempos de ejecución para todas las instancias probadas. En promedio, la heurística GTD entregó un GTDS en menos tiempo que las heurísticas GTAD y GTD-AD. La heurística GTD fue 95160 veces más rápida que GTAD y 1.18 veces más rápida que GTD-AD. Las Heurísticas GTD y GTD-AD son competitivas en términos de tiempo; GTD-AD fue más rápida que GTD en el 57.62 % de las instancias.



(a) Instancias con fewer than 550 vertices



(b) Instancias con más de 500 vértices

Figura 4.12: Tiempos de ejecución de las heurísticas

4.5. Conclusiones

En este capítulo se propusieron nuevas cotas inferiores y superiores para $\gamma_t^g(G)$. Además, se describió una formulación ILP y algoritmos de aproximación para el problema de dominación global total. Dado que anteriormente no existía ningún algoritmo de solución para el problema, se utilizaron valores óptimos obtenidos por el solver CPLEX para la formulación ILP a fin de analizar los resultados de las heurísticas. En las pruebas, se emplearon las instancias de referencia existentes anteriormente para el problema de dominación clásica. CPLEX fue capaz de resolver las 1,050 instancias de referencia existentes con hasta 10,700 vértices, mientras que las heurísticas encontraron soluciones de alta calidad para las instancias de referencia más grandes con hasta 25,000 vértices. Se proporcionaron algunas familias de grafos donde las heurísticas garantizaban una solución óptima.

Conjunto k -Dominante Global Total

5.1. Introducción

Para definir lo que es un conjunto k -dominante global total, es necesario introducir las siguientes definiciones.

Definición 5.1. Sea $D \subseteq V(G)$ un subconjunto no vacío, D es un conjunto k -dominante total del grafo G si hay al menos k vértices en el conjunto D adyacentes a cada vértice $v \in V(G)$, también diremos que el vértice v está k -dominado totalmente por el conjunto D .

Definición 5.2. El número de k -dominación total del grafo G , denotado por $\gamma_{kt}(G)$, es la cardinalidad del conjunto k -dominante total con el menor número de vértices.

En lo que sigue, nos referiremos a un conjunto k -dominante total de cardinalidad $\gamma_{kt}(G)$ como un $\gamma_{kt}(G)$ -conjunto. Un conjunto 1-dominante total se denomina normalmente *conjunto dominante total*, y el número de 1-dominación total se denomina *número de dominación total*, denotado por $\gamma_t(G)$. El lector se puede remitir a [10, 11, 13, 46, 65, 76] para más detalles sobre estas definiciones.

Definición 5.3. Dado un conjunto no vacío $D \subseteq V$, D es un conjunto k -dominante global total del grafo G (GTKDS, por sus siglas en inglés) si D es un conjunto k -dominante total tanto para el grafo G como para su complemento $\overline{G}$.

Definición 5.4. *El número de k -dominación global total de G , denotado por $\gamma_{kt}^g(G)$, es la cardinalidad de un conjunto k -dominante global total con el menor número de vértices.*

Un conjunto k -dominante global total con cardinalidad $\gamma_{kt}^g(G)$ será referido como un $\gamma_{kt}^g(G)$ -conjunto. Note que, si $k = 1$, un conjunto 1-dominante global total es un conjunto dominante global total, estudiado en el capítulo anterior (ver [1, 75]).

Como es bien conocido y también fácilmente observable,

$$2k + 1 \leq \gamma_{kt}^g(G) \leq n,$$

para cualquier grafo G de orden n . Aquí se estudiarán exclusivamente los grafos conexos, debido a un hecho conocido en [12], si $G_1, G_2, \dots, G_r$ ($r \geq 2$) son las componentes conexas del grafo G , entonces

$$\gamma_{kt}^g(G) = \sum_{i=1}^r \gamma_{kt}^g(G_i).$$

El objetivo principal de este capítulo es complementar el estudio actual del número de k -dominación global total en grafos. Primero, se dan cotas superiores e inferiores para $\gamma_{kt}^g(G)$, y luego se desarrolla un método que genera un $GTkDS$ a partir de un $GT(k-1)DS$ para valores sucesivos de k . Basados en este método, se establece una relación entre $\gamma_{(k-1)t}^g(G)$ y $\gamma_{kt}^g(G)$, lo que, a su vez, proporciona otra cota superior para $\gamma_{kt}^g(G)$.

El resto del capítulo está organizado de la siguiente manera. En la próxima sección se presentan resultados conocidos y algunas observaciones. En las Secciones 5.3 y 5.4 se derivan cotas superiores e inferiores, respectivamente, para $\gamma_{kt}^g(G)$. En la Sección 5.5 se propone un método que obtiene un conjunto $(k+1)$ -dominante global total a partir de un conjunto k -dominante global total.

5.2. Relaciones entre $\gamma_{kt}^g(G)$ y $\gamma_{kt}(G)$

Claramente, la definición de un $GTkDS$ proporciona una cota inferior para el parámetro $\gamma_{kt}^g(G)$:

Observación 5.1. *Sea G un grafo, entonces $\gamma_{kt}^g(G) \geq \max\{\gamma_{kt}(G), \gamma_{kt}(\overline{G})\}$.*

La cota inferior mencionada anteriormente no es necesariamente alcanzable, como se ilustra en la Figura 5.1 que muestra el grafo G y su complemento $\overline{G}$, junto con el conjunto 2-dominante total mínimo en ambos grafos (vértices negros).

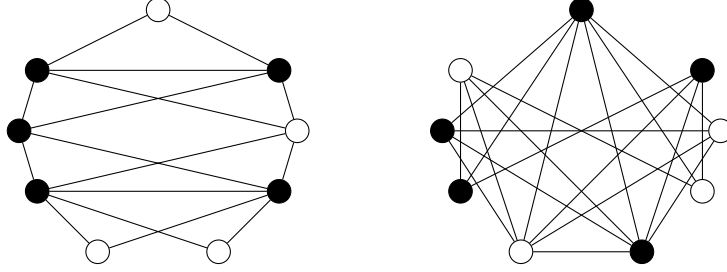


Figura 5.1: Grafo G y $\overline{G}$, que satisfacen $\gamma_{2t}(G) = 5$, $\gamma_{2t}(\overline{G}) = 5$ y $\gamma_{2t}^g(G) = 6$

La siguiente proposición fue demostrada en [12].

Proposición 5.1. *Sea G un grafo,*

- (i) *Si $\gamma_{kt}(G) > \Delta(G) + k$, entonces $\gamma_{kt}^g(G) = \gamma_{kt}(G)$.*
- (ii) *Si $\gamma_{kt}(G) \leq \Delta(G) + k$, entonces $\gamma_{kt}^g(G) \leq \Delta(G) + k + 1$.*

Corolario 5.1. *Sea G un grafo con grado máximo Δ . Entonces, $\gamma_{kt}^g(G) \leq \max\{\gamma_{kt}(G), \Delta + k + 1\}$.*

Proposición 5.2. *Sea G un grafo de orden n y grado máximo Δ . Si $n > \frac{\Delta(\Delta+k)}{k}$, entonces $\gamma_{kt}^g(G) = \gamma_{kt}(G)$.*

Demostración. Si $n > \frac{\Delta(\Delta+k)}{k}$, entonces $\Delta + k < \frac{kn}{\Delta} \leq \gamma_{kt}(G)$, en consecuencia, $\Delta + k + 1 \leq \gamma_{kt}(G)$. Por el Corolario 5.1 se obtiene que $\gamma_{kt}^g(G) = \gamma_{kt}(G)$. ■

Teorema 5.1. *Para cualquier grafo G , se tiene que $\gamma_{kt}^g(G) = \gamma_{kt}(G)$ si y solo si existe un conjunto k -dominación total mínimo D tal que cualquier subconjunto D' de D con $|D| - k + 1$ vértices no está incluido en ninguna estrella del grafo, es decir, no existe un vértice $v \in V$ tal que $D' \subseteq N[v]$.*

Demostración. Sea D un conjunto k -dominante total mínimo que también es un GT k DS, y sea D' un subconjunto de D con cardinalidad $|D| - k + 1$. Si existe un vértice $v \in V$ tal que $D' \subseteq N[v]$, entonces $v \in D'$ y es adyacente a $|D| - k$ vértices en D' , por lo que v tiene

menos de k vértices no adyacentes en D , o $v \notin D'$ y es adyacente a $|D| - k + 1$ vértices en D' , por lo que v tiene menos de k vértices no adyacentes en D . En ambos casos se tiene una contradicción con el hecho de que D es un $GTkDS$.

Por otro lado, tomemos un conjunto k -dominación total D mínimo, tal que cualquier subconjunto D' de D con $|D| - k + 1$ vértices y cada vértice $v \in V$, cumple que $D' \not\subseteq N[v]$. Entonces, para cualquier vértice $v \in D$, $N(v) < |D| - k$, por lo que v tiene, al menos, k no vecinos en D . Si $v \in V \setminus D$, $N(v) < |D| - k + 1$, entonces v tiene, al menos, k no vecinos en D . Por lo tanto, D es un $GTkDS$. ■

5.3. Cotas superiores para $\gamma_{kt}^g(G)$

En esta sección, se obtienen algunas cotas superiores para el número de k -dominación global total de un grafo G . Bermudo y otros, en [12], mostraron una caracterización cuando el número de k -dominación global total es igual al orden del grafo, pero aquí se presenta esa caracterización de una manera más específica. Para ello, en la siguiente proposición se da una condición que garantiza que el número de k -dominación global total es menor que n .

Proposición 5.3. *Sea G un grafo de orden n , con grado mínimo δ y grado máximo Δ . Si $k < \min\{\delta, n - \Delta - 1\}$, entonces $\gamma_{kt}^g(G) \leq n - 1$.*

Demostración. Veamos que, para cualquier $v \in V$, el conjunto $D = V \setminus v$ es un $GTkDS$ de G . Se tiene que $\delta_D(v) = \delta(v) \geq \delta > k$ y $\bar{\delta}_D(v) = n - 1 - \delta(v) \geq n - 1 - \Delta > k$. Para cada $u \in D$, $\delta_D(u) \geq \delta(u) - 1 \geq \delta - 1 \geq k$ y $\bar{\delta}_D(u) \geq n - 1 - \delta(u) - 1 \geq n - 2 - \Delta \geq k$. Por lo tanto, D es un $GTkDS$ de G . ■

La Proposición 5.3 no es una correspondencia. Si se considera un triángulo y se le añade una hoja a cada vértice del triángulo, en tal caso, $\gamma_{1t}^g(G) \leq n - 1 = 5$ y $k = 1 = \min\{\delta, n - \Delta - 1\}$.

Ahora, para presentar la caracterización de todos los grafos que tienen $\gamma_{kt}^g(G) = n$, se necesita definir el siguiente conjunto. Dado un grafo G y un entero i , sea $T_i(G) = \{v \in V(G) | \delta(v) = i\}$ (es decir, el conjunto de vértices en el grafo G con grado i).

Teorema 5.2. *Dado un grafo G de orden n y los grados mínimo y máximo, δ y Δ , se tiene que $\gamma_{kt}^g(G) = n$ si y solo si se cumple una de las condiciones (a)-(c) que se dan a continuación*

$$(a) \ k = \delta < n - \Delta - 1 \text{ y } V = \bigcup_{v \in T_\delta(G)} N(v).$$

$$(b) \ k = n - \Delta - 1 < \delta \text{ y } V = \bigcup_{w \in T_\Delta(G)} (V \setminus N[w]).$$

$$(c) \ k = \delta = n - \Delta - 1 \text{ y } V = \left(\bigcup_{v \in T_\delta(G)} N(v) \right) \cup \left(\bigcup_{w \in T_\Delta(G)} (V \setminus N[w]) \right).$$

Demostración. (a) Si $k = \delta < n - \Delta - 1$ y $V = \bigcup_{v \in T_\delta(G)} N(v)$, consideramos $D = V \setminus \{u\}$ para cualquier $u \in V$. Notemos que existe un $v \in N(u)$ tal que $\delta(v) = k$, lo que implica que $\delta_D(v) < k$. Por lo tanto, D no es un $GTkDS$ de G . Así, $\gamma_{kt}^g(G) = n$.

(b) Si $k = n - \Delta - 1 < \delta$ y $V = \bigcup_{w \in T_\Delta(G)} (V \setminus N[w])$, para cualquier $u \in V$ existe $w \in V$ tal que $\delta(w) = \Delta$ y $u \notin N[w]$. Si consideramos $D = V \setminus \{u\}$, entonces $\bar{\delta}_D(w) \leq n - \Delta - 2 < k$, por lo que D no es un $GTkDS$ de G . Por lo tanto, $\gamma_{kt}^g(G) = n$.

(c) Si $k = \delta = n - \Delta - 1$ y $V = \left(\bigcup_{v \in T_\delta(G)} N(v) \right) \cup \left(\bigcup_{w \in T_\Delta(G)} (V \setminus N[w]) \right)$, al utilizar (a) o (b) obtenemos que $V \setminus \{u\}$ no es un $GTkDS$ de G , para cualquier $u \in V$. En consecuencia, $\gamma_{kt}^g(G) = n$.

Finalmente, si asumimos que $\gamma_{kt}^g(G) = n$, por la Proposición 5.3 tenemos que $k \in \{\delta, n - \Delta - 1\}$. Para cada vértice $v \in V$, notemos que $D = V \setminus v$ no es un $GTkDS$ de G , por lo que existe $u \in D$ tal que $\delta_D(u) < k$ o $\bar{\delta}_D(u) < k$. Si $k = \delta < n - \Delta - 1$, dado que $\bar{\delta}_D(u) \geq n - 2 - \delta(u) \geq n - 2 - \Delta \geq k$, entonces $\delta_D(u) < k = \delta$, lo que implica que $u \in T_\delta(G)$ y $v \in N(u)$. Si $k = n - \Delta - 1 < \delta$, como $\delta_D(u) \geq \delta(u) - 1 \geq \delta - 1 \geq k$, entonces $n - 2 - \delta(u) \leq \bar{\delta}_D(u) < k = n - \Delta - 1$, es decir, $n - 2 - \delta(u) = \bar{\delta}_D(u) = n - \Delta - 2$, así que $u \in T_\Delta(G)$ y $v \in V \setminus N[u]$. Si $k = \delta = n - \Delta - 1$, como $\delta_D(u) < k$ o $\bar{\delta}_D(u) < k$, entonces $u \in T_\delta(G)$ y $v \in N(u)$, o $u \in T_\Delta(G)$ y $v \in V \setminus N[u]$. ■

El siguiente corolario se obtiene directamente del Teorema 5.2.

Corolario 5.2. *Sea G un grafo con grado mínimo δ , grado máximo Δ y orden $n \neq \Delta + \delta + 1$. Entonces, $\gamma_{kt}^g(G) = n$ si y solo si se cumple una de las siguientes condiciones*

(a) $k = \delta < n - \Delta - 1$ y $\gamma_{kt}(G) = n$.

(b) $k = n - \Delta - 1 < \delta$ y $\gamma_{kt}(\overline{G}) = n$.

Corolario 5.3. *Sea G un grafo de orden n , con grado mínimo δ y grado máximo Δ . Si se cumple una de las siguientes condiciones:*

(a) $k = \delta < n - \Delta - 1$ y $|T_\delta(G)| \geq n - \delta$,

(b) $k = n - \Delta - 1 < \delta$ y $|T_\Delta(G)| \geq \Delta + 1$,

(c) $k = \delta = n - \Delta - 1$ y $|T_\delta(G)| \geq n - \delta$ o $|T_\Delta(G)| \geq \Delta + 1$,

entonces $\gamma_{kt}^g(G) = n$.

Demostración. Dado que $\gamma_{kt}^g(G) = \gamma_{kt}^g(\overline{G})$, $\overline{\Delta} = n - \delta - 1$, $T_{\overline{\Delta}}(\overline{G}) = T_\delta(G)$ y $V \setminus N_{\overline{G}}[w] = N(w)$, basta con comprobar que $|T_\Delta(G)| \geq \Delta + 1$ implica que $V = \bigcup_{w \in T_\Delta(G)} (V \setminus N[w])$. Pero, para cualquier vértice $v \in V$, si $|T_\Delta(G)| \geq \Delta + 1$, entonces existe un vértice $w \in T_\Delta(G)$ que no es vecino de v , por lo que $v \in \bigcup_{w \in T_\Delta(G)} (V \setminus N[w])$. ■

Se demostró en [12] que $\gamma_{kt}^g(G) \leq \min\{\gamma_{kt}(G) + \Delta, \gamma_{kt}(G) + \gamma_{kt}(\overline{G})\}$. Sería conveniente caracterizar los grafos G tales que $\gamma_{kt}^g(G) = \gamma_{kt}(G) + \Delta$, y los grafos G tales que $\gamma_{kt}^g(G) = \gamma_{kt}(G) + \gamma_{kt}(\overline{G})$. Por otro lado, los invariantes de un grafo son importantes al caracterizarlos; a continuación, se utilizan algunos de ellos, como el diámetro y la longitud del ciclo más corto.

Teorema 5.3. *Si G es un grafo tal que $\text{diam}(G) \geq 5$, entonces todo conjunto k -dominante total es un $GTkDS$ de G .*

Demostración. Sea D un conjunto k -dominante total y $u, v \in V$ tales que $d(u, v) \geq 5$. Dado que $\delta_D(u) \geq k$ y $\delta_D(v) \geq k$, existen $\{u_1, \dots, u_k\} \subseteq D \cap N(u)$ y $\{v_1, \dots, v_k\} \subseteq D \cap N(v)$. Para cualquier vértice $w \in V$, se sabe que $\delta_D(w) \geq k$. Si $u_i \in N(w)$ para algún $i \in 1, \dots, k$, entonces $w \notin \bigcup_{i=1}^k N[v_i]$, lo que significa que $\overline{\delta}_D(w) \geq k$. Por lo tanto, D es un $GTkDS$ de G . ■

Corolario 5.4. *Si G es un grafo tal que $\text{diam}(G) \geq 5$, entonces $\gamma_{kt}^g(G) = \gamma_{kt}(G)$.*

Proposición 5.4. Si G es un grafo tal que $\text{diam}(G) = 4$ y existen $\{u, v_1, \dots, v_k\} \subseteq V$ tal que $\text{dist}(u, v_j) = 4$ para cada $j \in \{1, \dots, k\}$, entonces $\gamma_{kt}^g(G) \leq \gamma_{kt}(G) + k$.

Demostración. Sea D un conjunto k -dominante total mínimo de G . Entonces existe el conjunto de vértices $\{u_1, \dots, u_k\} \subseteq D$ tal que $\{u_1, \dots, u_k\} \subseteq N(u)$. Para cualquier vértice $w \in V$ e $i \in \{1, \dots, k\}$, w no puede ser adyacente ni a u_i ni a v_i , por lo que $D \cup \{v_1, \dots, v_k\}$ es un GT k DS. ■

En la Figura 5.2 se puede ver un ejemplo donde se alcanza la igualdad en la Proposición 5.4 para $k = 2$. Teniendo en cuenta que cualquier vecino de un vértice de grado 2 debe pertenecer a cualquier conjunto 2-dominante total (vértices grises), se muestra en esa figura el conjunto 2-dominante total mínimo (b) y el GT2DS mínimo (c).

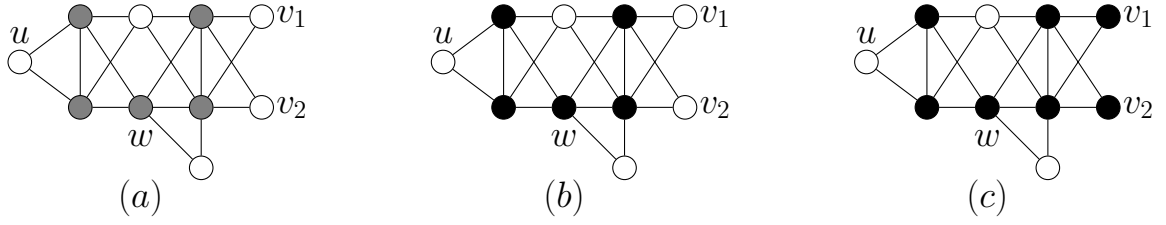


Figura 5.2: (a) G , (b) $\gamma_{2t}(G)$ -conjunto y (c) $\gamma_{2t}^g(G)$ -conjunto

Para un grafo G , se define $\delta^*(G) = \min\{\delta(G), \delta(\overline{G})\}$.

Proposición 5.5. Sea G un grafo de orden n y grado mínimo δ . Entonces, $\gamma_{kt}^g(G) \leq n - \delta^*(G) + k$.

Demostración. Veamos que cada conjunto $D \subseteq V$ tal que $|D| \geq n - \delta^*(G) + k$ es un GT k DS. Dado que $|D| \geq n - \delta + k$, cada vértice v satisface que $\delta_{V \setminus D}(v) \leq \delta - k$, por lo tanto, $\delta_D(v) \geq k$. Como $|D| \geq n - \bar{\delta} + k$, cada vértice v satisface que $\bar{\delta}_{V \setminus D}(v) \leq \bar{\delta} - k$, por lo que $\bar{\delta}_D(v) \geq k$. ■

5.4. Cotas inferiores para $\gamma_{kt}^g(G)$

Se sabe que cualquier grafo G satisface que $\gamma_{kt}^g(G) \geq 2k + 1$, y se proporcionó en [12], una caracterización para los grafos que satisfacen la igualdad. También en ese trabajo, los autores demostraron la siguiente desigualdad.

Observación 5.2. Sea G un grafo de orden n , con grado mínimo δ y grado máximo Δ . Entonces,

$$\gamma_{kt}^g(G) \geq \max \left\{ \frac{kn}{\Delta}, \frac{kn}{n - \delta - 1} \right\}$$

La cota inferior dada anteriormente se alcanza, por ejemplo, en el grafo que se muestra en la Figura 5.3, donde $\gamma_{2t}^g(G) = \frac{2n}{n - \delta - 1}$.

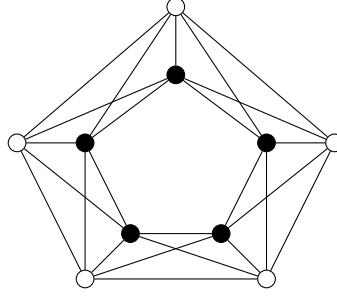


Figura 5.3: Grafo G de orden $n = 10$ y $\delta = 5$

Teorema 5.4. Sea G un grafo de orden n , con grado máximo Δ y tamaño m . Entonces,

$$\gamma_{kt}^g(G) \geq \frac{2m + n(2k - \Delta) + (2k + 1)^2}{n + 2k}.$$

Demostración. Sea D un $\gamma_{kt}(G)$ -conjunto. Dado que cada vértice en $V \setminus D$ no puede tener más de $|D| - k$ vecinos en D , tenemos que $E(D, V \setminus D) \leq (n - |D|)(|D| - k)$ entonces

$$\begin{aligned} m &= E(D, D) + E(D, V \setminus D) + E(V \setminus D, V \setminus D) \\ &\leq \frac{|D|\Delta(G) - E(D, V \setminus D)}{2} + E(D, V \setminus D) + \frac{(\Delta - k)(n - |D|)}{2} \\ &\leq \frac{|D|\Delta + (n - |D|)(|D| - k) + (\Delta - k)(n - |D|)}{2} \\ &= \frac{|D|\Delta + (n - |D|)(|D| - 2k + \Delta)}{2} \\ &= \frac{-|D|^2 + (n + 2k)|D| + n\Delta - 2kn}{2}, \end{aligned}$$

lo que implica que

$$(2k + 1)^2 + 2m \leq |D|^2 + 2m \leq (n + 2k)|D| + n\Delta - 2kn,$$

entonces

$$|D| \geq \frac{2m + n(2k - \Delta) + (2k + 1)^2}{n + 2k}.$$

■

Teorema 5.5. Sea G un grafo de orden n , con grado máximo Δ y tamaño m . Entonces,

$$\gamma_{kt}^g(G) \geq \frac{2m + n(\Delta - 2k)}{n + k - \Delta}.$$

Demostración. Suponga que D es un $\gamma_{kt}(G)$ -conjunto y que $|D| \geq 2r + 1$ para algún $r \geq 2$, y $|D| \geq 2k + 2$. Dado que D es minimal, para cualquier vértice $v_1 \in D$ existe un vértice w_{v_1} tal que se cumple una de las siguientes condiciones.

- (1) $w_{v_1} \in D$, $v_1 \in N(w_{v_1})$ y $\delta_D(w_{v_1}) = k$,
- (2) $w_{v_1} \in D$, $v_1 \notin N(w_{v_1})$ y $\delta_D(w_{v_1}) = |D| - k - 1$,
- (3) $w_{v_1} \in V \setminus D$, $v_1 \in N(w_{v_1})$ y $\delta_D(w_{v_1}) = k$,
- (4) $w_{v_1} \in V \setminus D$, $v_1 \notin N(w_{v_1})$ y $\delta_D(w_{v_1}) = |D| - k$.

Ahora, en los casos (1) y (3), tomamos $v_2 \in D \setminus N(w_{v_1})$, y, en los casos (2) y (4), tomamos $v_2 \in D \cap N(w_{v_1})$, y sabemos que existe un vértice $w_{v_2} \neq w_{v_1}$ tal que se cumple una de las condiciones anteriores. Dado que $|D| \geq 2r + 1$, podemos obtener los vértices $w_{v_1}, \dots, w_{v_r}$ que satisfacen una de las condiciones anteriores. Suponemos que existen $i, j - i, s$ y $r - j - s$ vértices que satisfacen las condiciones (1), (2), (3) y (4), respectivamente. Entonces,

$$\begin{aligned} E(D, D) &\leq \frac{ik + (j - i)(|D| - k - 1) + (|D| - j)(|D| - k - 1)}{2} \\ &= \frac{ik - i(|D| - k - 1) + |D|(|D| - k - 1)}{2} \\ &= \frac{i(2k - |D| + 1) + |D|(|D| - k - 1)}{2}, \end{aligned}$$

$$\begin{aligned} E(D, V \setminus D) &\leq \frac{sk + (r - j - s)(|D| - k) + (n - |D| - r + j)(|D| - k)}{2} \\ &= \frac{sk - s(|D| - k) + (n - |D|)(|D| - k)}{2} \\ &= \frac{(n - |D|)(|D| - k) + s(2k - |D|)}{2}, \end{aligned}$$

y

$$\begin{aligned}
E(V \setminus D, V \setminus D) &\leq \frac{s(\Delta - k) + (r - j - s)(\Delta - |D| + k)}{2} \\
&+ \frac{(n - |D| - r + j)(\Delta - k)}{2} \\
&= \frac{s(\Delta - k) + (r - j - s)(\Delta - k - |D| + 2k)}{2} \\
&+ \frac{(n - |D| - r + j)(\Delta - k)}{2} \\
&= \frac{(\Delta - k)(n - |D|) + (r - j - s)(2k - |D|)}{2},
\end{aligned}$$

por lo tanto,

$$\begin{aligned}
m &\leq E(D, D) + E(D, V \setminus D) + E(V \setminus D, V \setminus D) \\
&\leq \frac{i(2k - |D| + 1) + |D|(|D| - k - 1)}{2} + \frac{(n - |D|)(|D| - k) + s(2k - |D|)}{2} \\
&\quad + \frac{(\Delta - k)(n - |D|) + (r - j - s)(2k - |D|)}{2} \\
&= \frac{i(2k - |D| + 1) + |D|(|D| - k - 1)}{2} \\
&+ \frac{(n - |D|)(|D| - 2k + \Delta) + (r - j)(2k - |D|)}{2} \\
&= \frac{|D|(n + k - \Delta) + n(-2k + \Delta) + (i + r - j)(2k - |D|) + i}{2} \\
&\leq \frac{|D|(n + k - \Delta) + n(-2k + \Delta)}{2},
\end{aligned}$$

entonces

$$|D| \geq \frac{2m + n(\Delta - 2k)}{n + k - \Delta}.$$

■

Teorema 5.6. Sea G un grafo de orden n y grado máximo Δ . Si $k \geq \min\{\frac{\Delta}{2}, \frac{n-\delta-1}{2}\}$, entonces

$$\gamma_{kt}^g(G) \geq \frac{\sqrt{4kn+1}+1}{2}.$$

Demostración. Sea D un $\gamma_{kt}(G)$ -conjunto. Para cada $v \in D$, si suponemos que $k \geq \frac{\Delta}{2}$, se tiene que $\delta_D(v) \geq \delta_{\overline{D}}(v)$, entonces

$$|D|(|D| - k - 1) \geq \sum_{v \in D} \delta_D(v) \geq \sum_{v \in D} \delta_{\overline{D}}(v) \geq (n - |D|)k,$$

lo que implica que $|D|^2 - |D| \geq kn$ o, equivalentemente, $(|D| - \frac{1}{2})^2 \geq kn + \frac{1}{4}$, es decir, $|D| \geq \frac{\sqrt{4kn+1}+1}{2}$.

Si $\frac{n-\delta-1}{2} \leq k < \frac{\Delta}{2}$, como $\gamma_{kt}^g(G) = \gamma_{kt}^g(\overline{G})$ y $\overline{\Delta} = n - \delta - 1$, podemos obtener lo mismo. ■

La cota inferior dada en el Teorema 5.6 se alcanza, por ejemplo, en el grafo mostrado en la Figura 5.4.

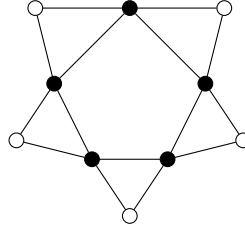


Figura 5.4: Grafo G con $\gamma_{2t}^g(G) \geq \frac{\sqrt{8n+1}+1}{2}$.

En la teoría de grafos es común analizar grafos obtenidos mediante alguna transformación a partir de un grafo dado originalmente. Un ejemplo de tal transformación es la eliminación de una o más aristas del grafo. Dado un grafo G , es natural preguntarse qué sucede si se añade una arista a ese grafo o si se elimina alguna de sus aristas. Note que, eliminar una arista en G es equivalente a añadir una arista al grafo $\overline{G}$. Por lo tanto, basta con estudiar solo uno de estos casos.

Proposición 5.6. *Sea G un grafo de orden n , con grado mínimo δ , grado máximo Δ y sea $k < \min\{\delta, n - \Delta - 1\}$. Entonces, se satisfacen las siguientes desigualdades para una arista e :*

$$\gamma_{kt}^g(G - e) \leq \gamma_{kt}^g(G) + 2,$$

$$\gamma_{kt}^g(G + e) \leq \gamma_{kt}^g(G) + 2.$$

Demostración. Sea G un grafo, D un $\gamma_{kt}^g(G)$ -conjunto, y consideremos $e \in E$. Note que $e \in E(V \setminus D, V \setminus D)$, $e \in E(D, V \setminus D)$ o $e \in E(D, D)$. Entonces, se puede dividir la prueba en tres casos. Denotaremos por G' al grafo $G - e$.

Caso 1: Si $e \in E(V \setminus D, V \setminus D)$. Note que cada vértice en $V(G')$ tiene al menos k vecinos y k no vecinos en D . Por lo tanto, $\gamma_{kt}^g(G') \leq |D| = \gamma_{kt}^g(G) < \gamma_{kt}^g(G) + 2$.

Caso 2: Si $e \in E(D, V \setminus D)$. Sea $e = \{u, v\}$, donde $u \in D$ y $v \in V \setminus D$. Note que, para cada $w \in V(G) \setminus \{v\}$, $\delta_D(w) \geq k$ y $\overline{\delta}_D(w) \geq k$. Por otro lado, note que $\overline{\delta}_D(v) > k$ en G' , y si

$\delta_D(v) \geq k$ en G' , entonces $\gamma^{gkt}(G') \leq |D| = \gamma_{kt}^g(G) < \gamma_{kt}^g(G) + 2$. Ahora, si $\delta_D(v) = k - 1$ en G' , entonces existe $w \in V(G') \setminus D$ tal que $w \in N_{G'}(v)$. Por lo tanto, $D \cup \{w\}$ es un GTkDS de G' , entonces $\gamma_{kt}^g(G') \leq |D \cup \{w\}| = \gamma_{kt}^g(G) + 1 < \gamma_{kt}^g(G) + 2$.

Caso 3: Si $e \in E(D, D)$. Sea $e = \{u, v\}$, donde $u, v \in D$. Note que, para todo $w \in V(G) \setminus \{u, v\}$, $\delta_D(w) \geq k$ y $\bar{\delta}_D(w) \geq k$. En el peor de los casos, $\delta_D(u) < k$ y $\delta_D(v) < k$. Los demás casos se resuelven como los anteriores, ya que existen $w, p \in V(G') \setminus D$ tales que $w \in N_{G'}(u)$ y $p \in N_{G'}(v)$. Ahora, si $w = p$, entonces $D \cup \{w\}$ es un GTkDS de G' y $\gamma_{kt}^g(G') \leq |D \cup \{w\}| = \gamma_{kt}^g(G) + 1 < \gamma_{kt}^g(G) + 2$; si $w \neq p$, entonces $D \cup \{w, p\}$ es un GTkDS de G' . Por lo tanto $\gamma_{kt}^g(G') \leq |D \cup \{w, p\}| = \gamma_{kt}^g(G) + 2$.

Por lo tanto, se satisface la primera desigualdad, $\gamma_{kt}^g(G - e) \leq \gamma_{kt}^g(G) + 2$. Ahora bien, como mencionamos anteriormente, para este problema eliminar una arista en G es análogo a agregar una arista en $\bar{G}$. Dado que $G - e$ y $\bar{G} + e$ son grafos complementarios, y es sabido que $\gamma_{kt}^g(\bar{G}) = \gamma_{kt}^g(G)$, entonces $\gamma_{kt}^g(G - e) = \gamma_{kt}^g(\bar{G} + e)$. Por lo tanto, por la primera desigualdad $\gamma_{kt}^g(\bar{G} + e) = \gamma_{kt}^g(G - e) \leq \gamma_{kt}^g(G) + 2 = \gamma_{kt}^g(\bar{G}) + 2$. Entonces, $\gamma_{kt}^g(G + e) \leq \gamma_{kt}^g(G) + 2$. ■

Definición 5.5. Sea S un subconjunto del conjunto $V(G)$ tal que el grado máximo del subgrafo inducido por el conjunto S no es mayor que $k - 1$. Entonces, el conjunto S es un conjunto k -independiente de G .

Definición 5.6. El número de k -independencia del grafo G , denotado por $\beta_k(G)$, es la cardinalidad de un conjunto k -independiente de máxima cardinalidad.

Definición 5.7. El número de k -independencia inferior, $i_k(G)$, es la cardinalidad mínima de un conjunto k -independiente maximal en el grafo G .

Proposición 5.7. Sea D un GTkDS de G y sea $V \setminus D$ un conjunto $(\Delta - k)$ -independiente maximal. Entonces,

$$n - \beta_{\Delta-k}(G) \leq |D| \leq \min\{n - \gamma(G), n - i_{\Delta-k}(G)\}.$$

Demostración. Dado que $V \setminus D$ es un conjunto $(\Delta - k)$ -independiente maximal, entonces $V \setminus D$ es un conjunto dominante, por lo que $n - |D| \geq \gamma(G)$. Además, $i_{\Delta-k}(G) \leq n - |D| \leq \beta_{\Delta-k}(G)$. ■

5.5. Derivación de cotas superiores para $\gamma_{(k+1)t}^g(G)$ a partir de $\gamma_{kt}^g(G)$

Es intuitivamente claro que cuanto mayor es k , más difícil es encontrar un $\text{GT}k\text{DS}$ de cardinalidad mínima en un grafo $G = (V, E)$. En particular, se observa fácilmente la siguiente relación: $\gamma_{1t}^g(G) \leq \gamma_{2t}^g(G) \leq \gamma_{3t}^g(G) \leq \dots \leq \gamma_{kt}^g(G)$, para cada $k \leq \min\{\delta, n - \Delta - 1\}$. Idealmente, uno desearía tener un método que obtenga un $\text{GT}(k+1)\text{DS}$ de cardinalidad mínima a partir de un $\text{GT}k\text{DS}$ de cardinalidad mínima. Está claro que esta no es una tarea fácil. En esta sección se desarrolla un método que genera un $\text{GT}(k+1)\text{DS}$ a partir de un $\text{GT}k\text{DS}$. Basado en éste, se establece una relación entre $\text{GT}k\text{DS}$ y $\text{GT}(k+1)\text{DS}$ de cardinalidad mínima, más precisamente, entre $\gamma_{kt}^g(G)$ y $\gamma_{(k+1)t}^g(G)$, lo que, a su vez, proporciona cotas superiores para $\gamma_{(k+1)t}^g(G)$.

Primero es necesario introducir algunas definiciones. Dado un subconjunto $D \subseteq V$ del conjunto de vértices V , sea $N(D)$ el conjunto de vértices de $V \setminus D$ que tienen al menos un vecino en D ; es decir, $N(D) = \{x \in V \setminus D \mid \exists y \in D \text{ tal que } x \in N_V(y)\}$. De manera similar, se denota por $\overline{N}(D)$ al conjunto de vértices de $V \setminus D$ que tienen al menos un no vecino en D .

Definición 5.8. Sean A y B subconjuntos del conjunto V . Un subconjunto $D \subseteq A$ es un conjunto de dominación relativa de B desde el conjunto A si para cada $x \in B$ existe al menos un vértice $v \in D$ tal que $v \in N(x)$ o $v \in B$.

Definición 5.9. La cardinalidad mínima de dicho conjunto de dominación relativa es el número de dominación relativa del conjunto B desde el conjunto A y se denota por $\gamma'(A, B)$.

Se abrevia como $\gamma'(A, B)$ -conjunto a un conjunto de dominación relativa de B desde A con cardinalidad $\gamma'(A, B)$.

Finalmente, $\gamma'(\overline{A}, \overline{B})$ es el número de dominación relativa de B desde el conjunto A en el grafo $\overline{G}$ y un $\gamma'(\overline{A}, \overline{B})$ -conjunto es un conjunto de dominación relativa de B desde el conjunto A en el grafo $\overline{G}$ con cardinalidad $\gamma'(\overline{A}, \overline{B})$.

Lema 5.1. Sea G un grafo con $\text{diam}(G) = 2$, $g(G) = 4$ y sea S un subgrafo inducido isomorfo a C_4 . Además, sea $B = V \setminus (N(S) \cup S)$ y $A = N(B)$. Entonces, $\gamma_{1t}^g(G) \leq \gamma'(A, B) + 4$.

Demostración. Sea D' un $\gamma'(A, B)$ -conjunto, $D = S \cup D'$ y $v \in V$. Dado que $\text{diam}(G) = 2$, $D' \subseteq A \subseteq N(S)$. Así, podemos ver que $v \in N(S)$, $v \in B$ o $v \in S$. Si $v \in N(S)$, entonces tiene al menos un vecino en S y, por lo tanto, también en D . Por otro lado, si $v \in B$, entonces v debe tener al menos un vecino en D' y, por ende, también en D . Si $v \in S$, entonces v tiene al menos un vecino en S y, por lo tanto, también en D . Entonces, D es un conjunto 1-dominante total de G .

Si $v \in S$, entonces existe al menos un vértice no vecino de v en S , y por ende también en D . Si $v \in B$, entonces los cuatro vértices en S son no vecinos de v , y por lo tanto el vértice v tiene al menos un no vecino en el conjunto D . Si $v \in N(S)$, dado que $g(G) = 4$, v tiene como máximo dos vecinos en S , por lo que tiene al menos dos no vecinos en S y, por lo tanto, también en D . Entonces, D es un conjunto 1-dominante global de G . Finalmente, D es un conjunto 1-dominante global total de G , así que $\gamma_{1t}^g(G) \leq \gamma'(A, B) + |S| = \gamma'(A, B) + 4$. ■

En la Figura 5.5 se representa el grafo G , donde el conjunto S está formado por los vértices blancos, el conjunto A por los vértices negros y el conjunto B por los vértices grises. Observe que $\gamma'(A, B) = 2$, el conjunto $\{u, v\}$ es un $\gamma'(A, B)$ -conjunto y $\gamma_{1t}^g(G) = 6$.

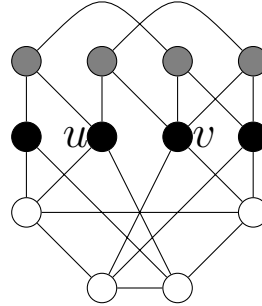


Figura 5.5: Grafo G con $\gamma'(A, B) = 2$ y $\gamma_{1t}^g(G) = 6$

Corolario 5.5. Sea G un grafo con $\text{diam}(G) = 2$, $g(G) = 4$, y sea S un subgrafo inducido isomorfo a C_4 , $B = V \setminus (N(S) \cup S)$ y $A = N(B)$. Entonces se cumplen las siguientes condiciones.

- Si $B = \emptyset$, entonces $\gamma_{1t}^g(G) = 4$.

- Como $\gamma'(A, B) \leq |B|$, entonces $\gamma_{1t}^g(G) \leq |B| + 4$.
- Si $|N(x) \cap S| = 2$, $\forall x \in A$, entonces $\gamma_{2t}^g(G) \leq 2|B| + 4$.

Sea k un entero positivo con $1 \leq k < \min\{\delta, n - \Delta - 1\}$, y D un $\gamma_{kt}^g(G)$ -conjunto del grafo G . A continuación, se definen conjuntos especiales de vértices que se utilizarán más adelante en las derivaciones propuestas.

- $H = V(G) \setminus D$.
- $Z = \{x \in H \mid \delta_D(x) \geq k + 1 \text{ y } \bar{\delta}_D(x) \geq k + 1\}$ son todos los vértices en H que están global total $(k + 1)$ -dominados.
- $X = T_k(G[D])$ son todos los vértices en D con sólo k vecinos.
- $Y = T_{|D|-k-1}(G[D])$ son todos los vértices en D con sólo k no vecinos.
- $X' = N(X) \cap H$ son todos los vértices en H que tienen al menos un vecino en el conjunto X .
- $N = \gamma'(X', X)$ -conjunto, un conjunto dominante relativo de X desde el conjunto X' .
- $Y' = \bar{N}(Y) \cap H$ son todos los vértices en el conjunto H que tienen al menos un no vecino en el conjunto Y .
- $R = \gamma'(\overline{Y'}, \overline{Y})$ -conjunto, un conjunto dominante relativo de X desde el conjunto X' en $\overline{G}$.
- $P = H \setminus Z$ son todos los vértices en el conjunto H que aún no son global total $(k + 1)$ -dominados.
- $M = \gamma'(H, P)$ -conjunto $\cup \gamma'(\overline{H}, \overline{P})$ -conjunto;
- $S = D \cup N \cup R \cup M$;

Ahora se muestra que el conjunto S obtenido como se indicó anteriormente es un $GT(k+1)DS$ dado un $\gamma_{kt}^g(G)$ -conjunto D .

Teorema 5.7. *Sea G un grafo y D un $\gamma_{kt}^g(G)$ -conjunto arbitrario. Entonces, el conjunto S obtenido como se indicó anteriormente es un $GT(k + 1)DS$ del grafo G .*

Demostración. Sea D un $\gamma_{kt}^g(G)$ -conjunto arbitrario, $H = V \setminus D$, $Z = \{x \in H : \delta_D(x) \geq k+1 \text{ y } \bar{\delta}_D(x) \geq k+1\}$, $X = T_k(G[D])$ y $Y = T_{|D|-k-1}(G[D])$. Además, sea $P = H \setminus Z$, E un $\gamma'(H, P)$ -conjunto, F un $\gamma'(\overline{H}, \overline{P})$ -conjunto y $M = E \cup F$ (todos estos conjuntos contruidos como se especificó anteriormente). Si $X = \emptyset$ y $Y = \emptyset$, entonces cada vértice de $D \cup Z$ tiene al menos $k+1$ vértices adyacentes y $k+1$ vértices no adyacentes en el conjunto D . Además, note que cada vértice $v \in P$ tiene al menos $k+1$ vértices adyacentes y $k+1$ vértices no adyacentes en el conjunto $D \cup M$. Y dado que $V = D \cup Z \cup P$, entonces $D \cup M$ es un $\text{GT}(k+1)\text{DS}$ del grafo G .

Asumamos ahora que $X \neq \emptyset$, $Y = \emptyset$, $X' = N(X) \cap H$ y N es un $\gamma'(X', X)$ -conjunto (observe que, por la construcción del conjunto X' , siempre existe el conjunto N). Note que cada vértice del conjunto $D \cup Z$ tiene al menos $k+1$ vértices adyacentes y $k+1$ vértices no adyacentes en el conjunto $D \cup N$. Además, cada vértice $v \in P$ tiene al menos $k+1$ vértices adyacentes y $k+1$ vértices no adyacentes en el conjunto $D \cup M$. Dado que $V = D \cup Z \cup P$, entonces $D \cup N \cup M$ es un $\text{GT}(k+1)\text{DS}$ de G .

El caso $X = \emptyset$ y $Y \neq \emptyset$ es análogo al caso anterior. Obtenemos que $D \cup R \cup M$ es un $\text{GT}(k+1)\text{DS}$ de G , donde $Y' = \overline{N}(Y) \cap H$ y R es un $\gamma'(\overline{Y'}, \overline{Y})$ -conjunto.

Finalmente, asumamos que $X \neq \emptyset$ y $Y \neq \emptyset$. Sea $X' = N(X) \cap H$, $Y' = \overline{N}(Y) \cap H$, N es un $\gamma'(X', X)$ -conjunto y R un $\gamma'(\overline{Y'}, \overline{Y})$ -conjunto. Usando argumentos similares a los anteriores, nuevamente obtenemos que S es un $\text{GT}(k+1)\text{DS}$ del grafo G . ■

En la siguiente proposición se deriva una cota superior para $\gamma_{(k+1)t}^g(G)$. En la misma proposición, se da una condición necesaria para que $\gamma_{(k+1)t}^g(G)$ sea igual a $\gamma_{(k+1)t}(G)$.

Proposición 5.8. *Sea G un grafo con $\delta \geq k$ y D un $\gamma_{kt}^g(G)$ -conjunto. Entonces se cumplen las siguientes condiciones:*

- a) $\gamma_{(k+1)t}^g(G) \leq \gamma_{kt}^g(G) + |N \cup R \cup M|$.
- b) Si $|N \cup M| > \Delta + k - \gamma_{kt}^g(G)$, entonces $\gamma_{(k+1)t}^g(G) = \gamma_{(k+1)t}(G)$.

Demostración. a) Por el Teorema 5.7, S es un $\text{GT}(k+1)\text{DS}$ de G , por lo tanto, la cota se cumple de manera trivial.

b) Recordemos que $|S| = \gamma_{kt}^g(G) + |N \cup R \cup M|$. Además, es fácil ver que $S \setminus R$ es un $\text{GT}(k+1)\text{DS}$ de G . En [12] se demuestra que si $\gamma_{kt}(G) > \Delta + k$, entonces $\gamma_{kt}^g(G) = \gamma_{kt}(G)$

(ver Proposición 5.1). Por lo tanto, si $|S| \geq \gamma_{kt}^g(G) + |N \cup M| \geq \gamma_{(k+1)t}(G) > \Delta + k + 1$, entonces $\gamma_{(k+1)t}^g(G) = \gamma_{(k+1)t}(G)$. Si $|N \cup M| > \Delta + k + 1 - \gamma_{kt}^g(G)$ entonces $\gamma_{(k+1)t}^g(G) = \gamma_{(k+1)t}(G)$. ■

Usando la definición de los conjuntos introducidos anteriormente, el Teorema 5.7 y la Proposición 5.8, se puede obtener un $GTkDS$ para cualquier $k = 2, \dots, \min\{\delta, n - \Delta - 1\}$. Como resultados adicionales, también se obtienen las correspondientes cotas superiores para $\gamma_{kt}^g(G)$. Finalmente, note que este procedimiento proporciona un $GTkDS$ de cardinalidad mínima, $2 \leq k \leq \min\{\delta, n - \Delta - 1\}$, para algunos grafos (ver Figura 5.6).

En la Figura 5.6 se muestra un grafo G con $\gamma_{1t}^g(G) = 4$, $\gamma_{2t}^g(G) = 6$ y $\gamma_{3t}^g(G) = 8$. Note que si $D = \{v_1, v_2, v_3, v_4\}$ es un $\gamma_{1t}^g(G)$ -conjunto, entonces $S = \{v_1, v_2, v_3, v_4, v_5, v_7\}$ es un $\gamma_{2t}^g(G)$ -conjunto. De manera similar, a partir de S se obtiene $S' = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8\}$, que es un $\gamma_{3t}^g(G)$ -conjunto.

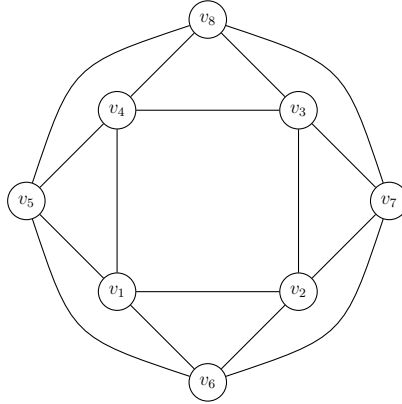


Figura 5.6: Grafo G con $\gamma_{1t}^g(G) = 4$, $\gamma_{2t}^g(G) = 6$ y $\gamma_{3t}^g(G) = 8$

5.6. Conclusiones

En este capítulo se estudió el número de k -dominación global total en grafos generales. En particular, se presentaron nuevas cotas superiores e inferiores utilizando la conectividad algebraica en grafos. También se estableció una relación entre el número de k -dominación global total del grafo originalmente dado G y los transformados. Luego se derivó una relación explícita entre un $\gamma_{kt}^g(G)$ -conjunto y un $\gamma_{(k+1)t}^g(G)$ -conjunto, lo que permitió obtener otra cota superior para $\gamma_{kt}^g(G)$ de manera recurrente, comenzando desde $k = 1$. Se dió un ejemplo de

un grafo G para el cual se proporciona un $\gamma_{kt}^g(G)$ -conjunto para cada $k = 2, \dots, \text{mín}\{\delta, n - \Delta - 1\}$. Para trabajos futuros, sería una tarea desafiante modificar el método propuesto para proporcionar un $\gamma_{kt}^g(G)$ -conjunto para una clase más amplia de grafos.

Conclusiones Generales

La investigación realizada logró cumplir el objetivo general de *estudiar las propiedades matemático-computacionales asociadas a problemas de la teoría de dominación en grafos*, logrando resultados significativos en la resolución de problemas de dominación y en el desarrollo de algoritmos heurísticos y exactos. A continuación, se presentan las conclusiones generales con los principales resultados y aportes, estructurados en función de los objetivos específicos planteados.

1. *Estudio de propiedades matemático-computacionales en problemas de dominación en grafos*: A lo largo del trabajo, se abordaron y analizaron problemas de dominación en grafos como la dominación mínima, dominación global, dominación global total, y k -dominación global total, que pertenecen a la clase de problemas NP -hard. En particular, se estudiaron sus propiedades estructurales y sus limitaciones computacionales. La tesis confirmó la alta complejidad de estos problemas en diversas familias de grafos y, en algunos casos, se identificaron propiedades específicas que permitieron derivar soluciones óptimas en ciertas clases de grafos. También se establecieron nuevas cotas teóricas para el número de dominación global total y k -dominación global total, aportando al conocimiento sobre la estructura de soluciones óptimas y la complejidad de estos problemas en contextos específicos.
2. *Desarrollo de algoritmos eficientes para problemas de dominación en grafos*: Se desarrollaron y probaron algoritmos heurísticos y exactos para resolver los problemas de conjunto dominante mínimo, dominación global y dominación global total. Para el conjunto dominante mínimo, se presentó un enfoque de dos etapas que combinó un algoritmo voraz con una fase de purificación, lo que resultó en soluciones de alta calidad en términos de tiempo de ejecución y precisión. En el caso de la dominación global

y global total, se diseñaron algoritmos de aproximación y formulaciones basadas en Programación Lineal Entera (ILP). Al combinar estas formulaciones con solvers como CPLEX, los algoritmos lograron resolver instancias de referencia con hasta miles de vértices, confirmando su eficiencia en términos de escalabilidad y precisión. Además, los métodos heurísticos propuestos alcanzaron la solución óptima en un alto porcentaje de las instancias probadas, y se utilizaron cotas superiores e inferiores para mejorar la precisión de los resultados en casos donde la solución óptima no se conocía.

3. *Identificación de familias de grafos para soluciones óptimas y pruebas de algoritmos en instancias de tamaño real:* Se identificaron varias familias de grafos en las que los algoritmos desarrollados garantizan soluciones óptimas o cercanas al óptimo. Para el problema de dominación global y dominación global total, se verificaron experimentalmente las soluciones obtenidas en instancias grandes, confirmando que los algoritmos propuestos son capaces de mantener un alto rendimiento y precisión. Las pruebas realizadas en instancias de referencia confirmaron que los métodos heurísticos alcanzaban soluciones de alta calidad en tiempos de ejecución razonables. Estas pruebas fortalecen la aplicabilidad de los algoritmos propuestos en grafos de gran escala y demuestran que pueden resolver instancias de tamaño real de forma eficiente.
4. *Obtención de nuevas cotas teóricas para el número de dominación global total y k -dominación global total:* La investigación aportó nuevas cotas superiores e inferiores para el número de dominación global total ($\gamma_t^g(G)$) y el número de k -dominación global total ($\gamma_{kt}^g(G)$). En el estudio de la k -dominación global total, se estableció una relación recurrente entre el número de k -dominación y el de $(k+1)$ -dominación, lo cual permitió derivar cotas que pueden calcularse de manera incremental. Además, se mostró cómo estas relaciones pueden utilizarse para obtener cotas efectivas en familias de grafos específicas y proporcionar soluciones de alta calidad para valores grandes de k .

En conclusión, esta investigación ha proporcionado una base sólida para el estudio y la resolución de problemas de dominación en grafos, desarrollando tanto enfoques teóricos como algoritmos prácticos que destacan por su aplicabilidad y efectividad en instancias de gran escala. Los métodos y algoritmos propuestos ofrecen soluciones eficientes para varios problemas de dominación, contribuyendo con nuevas técnicas y cotas teóricas que enriquecen tanto

la teoría de grafos como la práctica en optimización. Las pruebas experimentales realizadas confirmaron la eficacia de estas soluciones, resaltando la utilidad de los enfoques descritos para resolver problemas complejos en una variedad de contextos.

Problemas abiertos

A partir de los resultados obtenidos, surgen varias líneas de investigación futuras. Se recomienda para futuras investigaciones, los siguientes problemas abiertos:

1. Extender las técnicas desarrolladas a nuevas clases de grafos con restricciones adicionales, como grafos de alta conectividad o con ciclos específicos.
2. Mejorar la escalabilidad de los algoritmos exactos para problemas de dominación global total y k -dominación global total, buscando optimizar aún más el tiempo de cómputo en instancias de gran tamaño.
3. Adaptar el algoritmo para el $\gamma_{kt}^g(G)$ y pueda aplicarse en grafos con propiedades estructurales más diversas, contribuyendo así a una mayor generalización de los resultados alcanzados en esta tesis.

Referencias

- [1] AKHBARI, M. H., ESLAHCHI, C., RAD, N. J., AND HASNI, R. Some remarks on global total domination in graphs. *Appl. Math. E-Notes* 15 (2015), 22–28.
- [2] ALON, N., AND GUTNER, S. Linear time algorithms for finding a dominating set of fixed size in degenerated graphs. *Algorithmica* 54 (2009), 544–556.
- [3] ALVARADO, J. D., DANTAS, S., AND RAUTENBACH, D. Perfectly relating the domination, total domination, and paired domination numbers of a graph. *Discrete Mathematics* 338, 8 (2015), 1424–1431.
- [4] AMOS, D., ASPLUND, J., BRIMKOV, B., AND DAVILA, R. The slater and sub-k-domination number of a graph with applications to domination and k-domination. *Discussiones Mathematicae Graph Theory* 40, 1 (2020), 209–225.
- [5] BALAJI, S., AND SASILATHA, T. Detection of denial of service attacks by domination graph application in wireless sensor networks. *Cluster Computing* 22, 6 (2019), 15121–15126.
- [6] BALASUNDARAM, B., AND BUTENKO, S. Graph domination, coloring and cliques in telecommunications. In *Handbook of optimization in telecommunications*. Springer, 2006, pp. 865–890.
- [7] BASILIO, L. A., BERMUDO, S., LEAÑOS, J., AND SIGARRETA, J. M. β -differential of a graph. *Symmetry* 9, 10 (2017), 205.
- [8] BENTERT, M., VAN BEVERN, R., AND NIEDERMEIER, R. Inductive k-independent graphs and c-colorable subgraphs in scheduling: a review. *Journal of Scheduling* 22, 1 (2019), 3–20.

- [9] BERGE, C. *The Theory of Graphs and its Applications*. Methuen, London, 1958.
- [10] BERMUDO, S., HERNÁNDEZ-GÓMEZ, J. C., AND SIGARRETA, J. M. On the total k -domination in graphs. *Discussiones Mathematicae: Graph Theory* 38, 1 (2018).
- [11] BERMUDO, S., JALEMSKAYA, D. L., AND SIGARRETA, J. M. Total 2-domination in grid graphs. *Utilitas Mathematica* 110 (2019).
- [12] BERMUDO, S., MARTÍNEZ, A. C., MIRA, F. A. H., AND SIGARRETA, J. M. On the global total k -domination number of graphs. *Discrete Applied Mathematics* 263 (2019), 42–50.
- [13] BERMUDO, S., SÁNCHEZ, J., AND SIGARRETA, J. M. Total k -domination in cartesian product graphs. *Periodica Mathematica Hungarica* 75 (2017), 255–267.
- [14] BERTOSSI, A. A. Dominating sets for split and bipartite graphs. *Information processing letters* 19, 1 (1984), 37–40.
- [15] BIGGS, N., BIGGS, N. L., AND NORMAN, B. *Algebraic graph theory*, vol. 67. Cambridge university press, 1993.
- [16] BILLAUT, J.-C., DELLA CROCE, F., SALASSA, F., AND T’KINDT, V. No-idle, no-wait: when shop scheduling meets dominoes, eulerian paths and hamiltonian paths. *Journal of Scheduling* 22, 1 (2019), 59–68.
- [17] BOUAMAMA, S., AND BLUM, C. An improved greedy heuristic for the minimum positive influence dominating set problem in social networks. *Algorithms* 14, 3 (2021), 79.
- [18] BOYACI, A., AND MONNOT, J. Weighted upper domination number. *Electronic Notes in Discrete Mathematics* 62 (2017), 171–176.
- [19] BRAEKERS, K., RAMAEKERS, K., AND VAN NIEUWENHUYSE, I. The vehicle routing problem: State of the art classification and review. *Computers & Industrial Engineering* 99 (2016), 300–313.
- [20] BRIGHAM, R. C., AND DUTTON, R. D. Factor domination in graphs. *Discrete Mathematics* 86, 1 (1990), 127–136.

- [21] BRIMKOV, B., MIKESELL, D., AND SMITH, L. Connected power domination in graphs. *Journal of Combinatorial Optimization* 38, 1 (2019), 292–315.
- [22] BURTSEVA, L., MDPI, Y. S., ET AL. *Algorithms for Scheduling Problems*. MDPI, 2018.
- [23] CABRERA MARTÍNEZ, A., HERNÁNDEZ-GÓMEZ, J. C., INZA, E. P., AND SIGARRETA, J. M. On the total outer k -independent domination number of graphs. *Mathematics* 8, 2 (2020), 194.
- [24] CABRERA MARTÍNEZ, A., HERNÁNDEZ MIRA, F. A., SIGARRETA ALMIRA, J. M., AND YERO, I. G. On computational and combinatorial properties of the total co-independent domination number of graphs. *The Computer Journal* 62, 1 (2019), 97–108.
- [25] CAMPAN, A., TRUTA, T. M., AND BECKERICH, M. Fast dominating set algorithms for social networks. In *Maics* (2015), pp. 55–62.
- [26] CARBALLOSA, W., RODRÍGUEZ, J. M., SIGARRETA, J. M., AND VAKHANIA, N. f -polynomial on some graph operations. *Mathematics* 7, 11 (2019), 1074.
- [27] CARO, Y., RODITTY, Y., ET AL. A note on the k -domination number of a graph. *International Journal of Mathematics and Mathematical Sciences* 13, 1 (1990), 205–206.
- [28] CHELLALI, M., FAVARON, O., HANSBERG, A., AND VOLKMANN, L. k -domination and k -independence in graphs: A survey. *Graphs and Combinatorics* 28, 1 (2012), 1–55.
- [29] CHEN, J., HE, K., DU, R., ZHENG, M., XIANG, Y., AND YUAN, Q. Dominating set and network coding-based routing in wireless mesh networks. *IEEE Transactions on Parallel and Distributed Systems* 26, 2 (2015), 423–433.
- [30] CHIARELLI, N., HARTINGER, T. R., LEONI, V. A., PUJATO, M. I. L., AND MILANIČ, M. New algorithms for weighted k -domination and total k -domination problems in proper interval graphs. *Theoretical computer science* 795 (2019), 128–141.
- [31] CHINOS, E., AND VAKHANIA, N. Polynomially solvable and np-hard special cases for scheduling with heads and tails. *Recent Advances in Mathematics and Computational Science.(MCSS 16). Barcelona, Spain* (2016), 141–145.

- [32] CHVATAL, V. A greedy heuristic for the set-covering problem. *Mathematics of operations research* 4, 3 (1979), 233–235.
- [33] COCKAYNE, E., GOODMAN, S., AND HEDETNIEMI, S. A linear algorithm for the domination number of a tree. *Information Processing Letters* 4, 2 (1975), 41–44.
- [34] COCKAYNE, E. J., DAWES, R., AND HEDETNIEMI, S. T. Total domination in graphs. *Networks* 10, 3 (1980), 211–219.
- [35] COLOMBI, M., MANSINI, R., AND SAVELSBERGH, M. The generalized independent set problem: Polyhedral analysis and solution approaches. *European Journal of Operational Research* 260, 1 (2017), 41–55.
- [36] DAVIDSON, P. P., BLUM, C., AND LOZANO, J. A. The weighted independent domination problem: Integer linear programming models and metaheuristic approaches. *European Journal of Operational Research* 265, 3 (2018), 860–871.
- [37] DESORMEAUX, W. J., GIBSON, P. E., AND HAYNES, T. W. Bounds on the global domination number. *Quaestiones Mathematicae* 38, 4 (2015), 563–572.
- [38] DESORMEAUX, W. J., HAYNES, T. W., HENNING, M. A., AND YEO, A. Total domination in graphs with diameter 2. *Journal of Graph Theory* 75, 1 (2014), 91–103.
- [39] DETTLAFF, M., LEMAŃSKA, M., RODRÍGUEZ-VELÁZQUEZ, J. A., AND ZUAZUA, R. On the super domination number of lexicographic product graphs. *Discrete Applied Mathematics* 263 (2019), 118–129.
- [40] DINH, T. N., SHEN, Y., NGUYEN, D. T., AND THAI, M. T. On the approximability of positive influence dominating set in social networks. *Journal of Combinatorial Optimization* 27, 3 (2014), 487–503.
- [41] DOREIAN, P., AND CONTI, N. Social context, spatial structure and social network structure. *Social networks* 34, 1 (2012), 32–46.
- [42] ENCISO, R. I., AND DUTTON, R. D. Global domination in planar graphs. *J. Combin. Math. Combin. Comput* 66 (2008), 273–278.

- [43] ERLEBACH, T., AND VAN LEEUWEN, E. J. Domination in geometric intersection graphs. In *Latin American Symposium on Theoretical Informatics* (2008), Springer, pp. 747–758.
- [44] EUBANK, S., KUMAR, V. A., MARATHE, M. V., SRINIVASAN, A., AND WANG, N. Structural and algorithmic aspects of massive social networks. In *Proceedings of the fifteenth annual ACM-SIAM symposium on Discrete algorithms* (2004), pp. 718–727.
- [45] FAN, N., AND WATSON, J.-P. Solving the connected dominating set problem and power dominating set problem by integer programming. In *International conference on combinatorial optimization and applications* (2012), Springer, pp. 371–383.
- [46] FERNAU, H., RODRÍGUEZ-VELÁZQUEZ, J., AND SIGARRETA, J. Global powerful r -alliances and total k -domination in graphs. *Util. Math* 98 (2015), 127–147.
- [47] FIEDLER, M. A property of eigenvectors of nonnegative symmetric matrices and its application to graph theory. *Czechoslovak mathematical journal* 25, 4 (1975), 619–633.
- [48] FINK, J. F., AND JACOBSON, M. S. n -domination in graphs. In *Graph theory with applications to algorithms and computer science* (1985), John Wiley & Sons, Inc., pp. 283–300.
- [49] FLAKE, G. W., LAWRENCE, S., AND GILES, C. L. Efficient identification of web communities. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining* (2000), pp. 150–160.
- [50] FOERSTER, K.-T. Approximating fault-tolerant domination in general graphs. In *2013 Proceedings of the Tenth Workshop on Analytic Algorithmics and Combinatorics (ANALCO)* (2013), SIAM, pp. 25–32.
- [51] FOLDES, S., AND HAMMER, P. L. Split graphs having dilworth number two. *Canadian Journal of Mathematics* 29, 3 (1977), 666–672.
- [52] FOMIN, F. V., GRANDONI, F., AND KRATSCH, D. A measure & conquer approach for the analysis of exact algorithms. *Journal of the ACM (JACM)* 56, 5 (2009), 1–32.

- [53] GAREY, M. R., AND JOHNSON, D. S. *Computers and intractability*, vol. 174. freeman San Francisco, 1979.
- [54] GASPERS, S., KRATSCH, D., LIEDLOFF, M., AND TODINCA, I. Exponential time algorithms for the minimum dominating set problem on some graph classes. *ACM Transactions on Algorithms (TALG)* 6, 1 (2009), 1–21.
- [55] GIBSON, M., AND PIRWANI, I. A. Approximation algorithms for dominating set in disk graphs. *arXiv preprint arXiv:1004.3320* (2010).
- [56] GODDARD, W., AND HENNING, M. A. Domination in planar graphs with small diameter. *Journal of Graph Theory* 40, 1 (2002), 1–25.
- [57] GUHA, S., AND KHULLER, S. Approximation algorithms for connected dominating sets. *Algorithmica* 20 (1998), 374–387.
- [58] GUPTA, P. Domination in graph with application. *Indian Journal of Research, Paripex, ISSN-2250-1991, Vol. No. 2, Issue No 3* (2013), 115–117.
- [59] GUZE, S. An application of the selected graph theory domination concepts to transportation networks modelling. *Scientific Journals of the Maritime University of Szczecin* (2017).
- [60] HAYNES, T. *Domination in Graphs: Advanced Topics Volume 2*. Routledge, 2017.
- [61] HAYNES, T., KNISLEY, D., SEIER, E., AND ZOU, Y. A quantitative analysis of secondary rna structure using domination based parameters on trees. *BMC bioinformatics* 7 (2006), 1–11.
- [62] HAYNES, T. W., HEDETNIEMI, S., AND SLATER, P. *Fundamentals of domination in graphs*. Marcel Dekker, Inc., New York, 1998.
- [63] HAYNES, T. W., HEDETNIEMI, S. M., HEDETNIEMI, S. T., AND HENNING, M. A. Domination in graphs applied to electric power networks. *SIAM Journal on Discrete Mathematics* 15, 4 (2002), 519–529.

- [64] HAYNES, T. W., HENNING, M. A., AND MYNHARDT, C. M. Global total domination in graphs. *Bulletin of the Institute of Combinatorics and its Applications* 24 (1998), 75–91.
- [65] HE, J., AND LIANG, H. Complexity of total $\{k\}$ -domination and related problems. In *Frontiers in Algorithmics and Algorithmic Aspects in Information and Management: Joint International Conference, FAW-AAIM 2011, Jinhua, China, May 28-31, 2011. Proceedings* (2011), Springer, pp. 147–155.
- [66] HEDETNIEMI, S. T., AND LASKAR, R. C. Bibliography on domination in graphs and some basic definitions of domination parameters. In *Annals of Discrete Mathematics*, vol. 48. Elsevier, 1991, pp. 257–277.
- [67] HERNÁNDEZ MIRA, F. A., PARRA INZA, E., SIGARRETA ALMIRA, J. M., AND VAKHANIA, N. Properties of the global total k -domination number. *Mathematics* 9, 5 (2021), 480.
- [68] HUNT III, H. B., MARATHE, M. V., RADHAKRISHNAN, V., RAVI, S. S., ROSENKRANTZ, D. J., AND STEARNS, R. E. N_c -approximation schemes for np-and pspace-hard problems for geometric graphs. *Journal of algorithms* 26, 2 (1998), 238–274.
- [69] INZA, E. P., VAKHANIA, N., ALMIRA, J. M. S., AND MIRA, F. A. H. Exact and heuristic algorithms for the domination problem. *European Journal of Operational Research* 313, 3 (2024), 926–936.
- [70] IWATA, Y. A faster algorithm for dominating set analyzed by the potential method. In *International Symposium on Parameterized and Exact Computation* (2012), Springer, pp. 41–54.
- [71] JOSHI, D. S., RADHAKRISHNAN, S., AND CHANDRASEKHARAN, N. The k -neighbor, r -domination problems on interval graphs. *European Journal of Operational Research* 79, 2 (1994), 352–368.
- [72] JOVANOVIĆ, R., SANFILIPPO, A. P., AND VOSS, S. Fixed set search applied to the clique partitioning problem. *European Journal of Operational Research* 309, 1 (2023), 65–81.

- [73] KELLEHER, L. L., AND COZZENS, M. B. Dominating sets in social network graphs. *Mathematical Social Sciences* 16, 3 (1988), 267–279.
- [74] KIKUNO, T., YOSHIDA, N., AND KAKUDA, Y. A linear algorithm for the domination number of a series-parallel graph. *Discrete Applied Mathematics* 5, 3 (1983), 299–311.
- [75] KULLI, V., AND JANAKIRAM, B. The total global domination number of a graph. *Indian J. pure appl. Math* 27, 6 (1996), 537–542.
- [76] KULLI, V. R. On n -total domination number in graphs. *Graph Theory, Combinatorics, Algorithms and Applications* (1991), 319–324.
- [77] KUO, T.-W. On the approximability and hardness of the minimum connected dominating set with routing cost constraint. *Theoretical Computer Science* 793 (2019), 140–151.
- [78] LIAO, C.-S., AND LEE, D.-T. Power domination problem in graphs. In *International Computing and Combinatorics Conference* (2005), Springer, pp. 818–828.
- [79] LICK, D. R., AND WHITE, A. T. k -degenerate graphs. *Canadian Journal of Mathematics* 22, 5 (1970), 1082–1096.
- [80] MARTÍNEZ, A. C., ALMIRA, J. M. S., AND YERO, I. G. On the independence transversal total domination number of graphs. *Discrete Applied Mathematics* 219 (2017), 65–73.
- [81] MEEK, D., AND GARY PARKER, R. A graph approximation heuristic for the vertex cover problem on planar graphs. *European Journal of Operational Research* 72, 3 (1994), 588–597.
- [82] MERRIS, R. Split graphs. *European Journal of Combinatorics* 24, 4 (2003), 413–430.
- [83] MIRA, F. A. H., INZA, E. P., ALMIRA, J. M. S., AND VAKHANIA, N. A polynomial-time approximation to a minimum dominating set in a graph. *Theoretical Computer Science* 930 (2022), 142–156.

- [84] NAZARI, M., OROOJLOOY, A., SNYDER, L., AND TAKÁC, M. Reinforcement learning for solving the vehicle routing problem. In *Advances in Neural Information Processing Systems* (2018), pp. 9839–9849.
- [85] NIEBERG, T., HURINK, J., AND KERN, W. Approximation schemes for wireless networks. *ACM Transactions on Algorithms (TALG)* 4, 4 (2008), 1–17.
- [86] ORE, O. Theory of graphs. In *Colloquium Publications* (1962), vol. 38, American Mathematical Society.
- [87] PANDA, B., AND GOYAL, P. Global total k-domination: Approximation and hardness results. *Theoretical Computer Science* 850 (2021), 1–19.
- [88] PANDA, B., AND GOYAL, P. Hardness results of global total k-domination problem in graphs. *Discrete Applied Mathematics* 319 (2022), 223–238.
- [89] PAREKH, A. K. Analysis of a greedy heuristic for finding small dominating sets in graphs. *Information processing letters* 39, 5 (1991), 237–240.
- [90] PARRA INZA, E. Random graph (1). *Mendeley Data V4* (2023).
- [91] PARRA INZA, E. Random graph. *Mendeley Data V6* (2024).
- [92] PARRA INZA, E. Random graph (1). *Mendeley Data V5* (2024).
- [93] PARRA INZA, E., SANDOVAL RAMÍREZ, A., HERNÁNDEZ GÓMEZ, J. C., AND CERDENARES LADRÓN DE GUEVARA, G. Análisis de la robustez de redes tróficas mediante el uso de conjuntos dominantes totales outer k-independientes. In *Modelación Matemática IV: Biomatemáticas, Epidemiología, Ingeniería*, 1 ed. Universidad Tecnológica de la Mixteca, Oaxaca, México, 2021, pp. 3–18.
- [94] PARRA INZA, E., VAKHANIA, N., SIGARRETA ALMIRA, J. M., AND HERNÁNDEZ-AGUILAR, J. A. Approximating a minimum dominating set by purification. *Algorithms* 17, 6 (2024), 258.
- [95] PARRA INZA, E., VAKHANIA, N., SIGARRETA ALMIRA, J. M., AND HERNÁNDEZ MIRA, F. A. Algorithms for the global domination problem. *Preprint submitted to Computers & Operations Research* (2023).

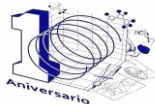
- [96] ROSENFELD, R., AND IRAZÁBAL, J. *Computabilidad, complejidad computacional y verificación de programas*, vol. 1. Universidad Nacional de La Plata – Editorial de la Universidad de La Plata, 2013.
- [97] SAMPATHKUMAR, E. The global domination number of a graph. *J. Math. Phys. Sci* 23, 5 (1989), 377–385.
- [98] SCHOENMAKERS, L. A. M. *A new algorithm for the recognition of series parallel graphs*. Centrum voor Wiskunde en Informatica Amsterdam, 1995.
- [99] SHAN, L., LI, H., AND ZHANG, Z. Domination number and minimum dominating sets in pseudofractal scale-free web and sierpiński graph. *Theoretical Computer Science* 677 (2017), 12–30.
- [100] SHAO, Z., KOSARI, S., RASHMANLOU, H., AND SHOAIB, M. New concepts in intuitionistic fuzzy graph with application in water supplier systems. *Mathematics* 8, 8 (2020), 1241.
- [101] SHCHEPIN, E. V., AND VAKHANIA, N. An optimal rounding gives a better approximation for scheduling unrelated machines. *Operations Research Letters* 33, 2 (2005), 127–133.
- [102] SIEBERTZ, S. Greedy domination on biclique-free graphs. *Information Processing Letters* 145 (2019), 64–67.
- [103] TYSHKEVICH, R. I., AND CHERNYAK, A. A. Canonical partition of a graph defined by the degrees of its vertices. *Izv. Akad. Nauk BSSR, Ser. Fiz.-Mat. Nauk* 5 (1979), 14–26.
- [104] VAKHANIA, N. A better algorithm for sequencing with release and delivery times on identical machines. *Journal of Algorithms* 48, 2 (2003), 273–293.
- [105] VAKHANIA, N. Efficient heuristics for scheduling with release and delivery times. *Heuristics and Hyper-Heuristics: Principles and Applications* (2017), 65.
- [106] VAKHANIA, N. Probabilistic quality estimations for combinatorial optimization problems. *Georgian Mathematical Journal* 25, 1 (2018), 123–134.

- [107] VAKHANIA, N. Scheduling a single machine with primary and secondary objectives. *Algorithms* 11, 6 (2018), 80.
- [108] VAKHANIA, N. Fast solution of single-machine scheduling problem with embedded jobs. *Theoretical Computer Science* 782 (2019), 91–106.
- [109] VAKHANIA, N., HERNANDEZ, J. A., ALONSO-PECINA, F., AND ZAVALA, C. A simple heuristic for basic vehicle routing problem. *J. Comput. Sci* 3 (2016), 39.
- [110] VAKHANIA, N., AND WERNER, F. Minimizing maximum lateness of jobs with naturally bounded job data on a single machine in polynomial time. *Theoretical Computer Science* 501 (2013), 72–81.
- [111] VAN ROOIJ, J. M., AND BODLAENDER, H. L. Exact algorithms for dominating set. *Discrete Applied Mathematics* 159, 17 (2011), 2147–2164.
- [112] VENKATARAMAN, Y., KALAISELVI, T., ANGELINA, J. J. R., AND JANAKIRAM, S. S. Dominating sets in protein-protein interaction networks. In *Joint 3rd International Conference on Bioinformatics and Data Science (ICBDS 2022)* (2023), Atlantis Press, pp. 5–14.
- [113] WAN, P.-J., ALZOUBI, K. M., AND FRIEDER, O. Distributed construction of connected dominating set in wireless ad hoc networks. *Mobile Networks and Applications* 9, 2 (2004), 141–149.
- [114] WANG, F., DU, H., CAMACHO, E., XU, K., LEE, W., SHI, Y., AND SHAN, S. On positive influence dominating sets in social networks. *Theoretical Computer Science* 412, 3 (2011), 265–269.
- [115] WANG, Q., NUTE, M., AND TREANGEN, T. J. Bakdrive: identifying a minimum set of bacterial species driving interactions across multiple microbial communities. *Bioinformatics* 39, Supplement_1 (2023), i47–i56.
- [116] WANG, Y., CAI, S., CHEN, J., AND YIN, M. A fast local search algorithm for minimum weight dominating set problem on massive graphs. In *IJCAI* (2018), pp. 1514–1522.

- [117] WANG, Y., CAI, S., AND YIN, M. Local search for minimum weight dominating set with two-level configuration checking and frequency based scoring function. *Journal of Artificial Intelligence Research* 58 (2017), 267–295.
- [118] WILSON, R. J. *Introduction to graph theory*. Fourth Edition, Addison Wesley, 1996.
- [119] WU, J. Extended dominating-set-based routing in ad hoc wireless networks with unidirectional links. *IEEE transactions on parallel and distributed systems* 13, 9 (2002), 866–881.
- [120] WU, J., WU, B., AND STOJMENOVIC, I. Power-aware broadcasting and activity scheduling in ad hoc wireless networks using connected dominating sets. *Wireless Communications and Mobile Computing* 3, 4 (2003), 425–438.



UNIVERSIDAD AUTÓNOMA DEL
ESTADO DE MORELOS



Instituto de
Investigación en
Ciencias
Básicas y
Aplicadas



INSTITUTO DE INVESTIGACIÓN EN CIENCIAS BÁSICAS Y APLICADAS

POSGRADO EN CIENCIAS

Cuernavaca, Mor., a 7 de febrero de 2025

DRA. LINA ANDREA RIVILLAS ACEVEDO
COORDINADORA DEL POSGRADO EN CIENCIAS

PRESENTE

Atendiendo a la solicitud para emitir DICTAMEN sobre la revisión de la tesis titulada: **PROPIEDADES MATEMÁTICO-COMPUTACIONALES ASOCIADAS A PROBLEMAS DE TEORÍA DE DOMINACIÓN**, que presenta el M.C.M. Ernesto Parra Inza (10046093) para obtener el título de Doctorado en Ciencias.

Director de tesis: Dr. Nodari Vakhania Maisuradze

Unidad Académica: Instituto de Investigación en Ciencias Básicas y Aplicadas (IICBA)

Nos permitimos informarle que nuestro voto es:

NOMBRE	DICTAMEN	FIRMA
Dra. Liudmila Lvovna Sabinina CInC-UAEM	APROBADO	Firma Electrónica
Dr. Antonio Daniel Rivera López CInC-UAEM	APROBADO	Firma Electrónica
Dr. José María Sigarreta Almira UAGro	APROBADO	Firma Electrónica
Dr. José Alberto Hernández Aguilar FCAel-UAEM	APROBADO	Firma Electrónica
Dr. Nodari Vakhania Maisuradze CInC-UAEM	APROBADO	Firma Electrónica
Dr. Ulises Morales Fuentes CInC-UAEM	APROBADO	Firma Electrónica
Dr. Omar Rosario Cayetano UAGro	APROBADO	Firma Electrónica



Av. Universidad 1001 Col. Chamilpa, Cuernavaca Morelos, México, 62209, Edificio 65
Tel. (777) 329 7011, 329 70, 00, Ext. 6011 / posgradoenciencias@uaem.mx

UAEM
RECTORÍA
2023-2029



UNIVERSIDAD AUTÓNOMA DEL
ESTADO DE MORELOS

Se expide el presente documento con firma electrónica UAEM, soportada por el certificado vigente a la fecha de su elaboración y con efectos plenos de conformidad con los LINEAMIENTOS EN MATERIA DE FIRMA ELECTRÓNICA PARA LA UNIVERSIDAD AUTÓNOMA DEL ESTADO DE MORELOS PUBLICADOS en el ÓRGANO INFORMATIVO UNIVERSITARIO "ADOLFO MENÉNDEZ SAMARÁ" número 117 de fecha 20 de abril de 2021.

Sello electrónico

JOSÉ MARÍA SIGARRETA ALMIRA | Fecha:2025-02-07 15:57:12 | FIRMANTE

u9DqEVC7IRNVYj5I45v0s9vIRIm6YNNJQ0dM6hgtuo0x0TFcenV2iG/D9qpDnvQWQSN61ANOWsGL8M8VnEX4Gy7zck+g3rJy0gNn2R/dhWBA0/KuFVSXwNxfyVKzVEpY6gR
W7hHc4slgkyNiCTPIWk5YSMP6eg2jleYebSGaHg6iMkYXYRR6jAiM5ymaTPzPHOnZ4S3JxdWYqkatjZydkkmdMopFR6V2RA5FzaDAbR+w18rz2TQpYhYXI97XkQsXSUrqwVh
aO3tzUWeLW8hiLQdvlW3UwFVl0sbN1bqzCg4PSEjxrJV3l8pO6EZFfDMw1ubgCcVeA/KHGCBd4A==

NODARI VAKHANIA MAISURADZE | Fecha:2025-02-07 21:04:13 | FIRMANTE

DNDvJsaLrZg9RFSHCyKIEEp7H05dH+xDTj8iA+9HbSMiD0qgFLAFaNsJu0l0jNFWs2DKab3FsUdFB+UCT9E+jxxaT6VuqFStzPVyVlvXOENpNP7mmLFWdzPclRnwLWf4DOaH
u9srCxGAXCK2Dmjz67Mhh7jv0D7Om8xl3eZE8Uke+7NI2nvDEI3ecixlT/KwWz0ETzPwo8piDsR2nctiVr1WOn64PwwuplJcjin+H7RM4aUXtABT1mlhiTRw3uAe7Ba8XPnRBu735
eHfbW48HLb2s62PjvNnGlbqAATIIIME59qXYsfkeDuY8KSpNbURpjVlttnN49UZfiqL6Q==

OMAR ROSARIO CAYETANO | Fecha:2025-02-08 15:10:22 | FIRMANTE

UUZ5wESTGXN+E0bYhgftZMastNewfEEM7ZC1lrKVZmlstEkmsb172ZBcdUDkLwUA4fMkpVLGKjlvO5PPkhkbs7QDLLqN8dRnngHcsbB3A8L24epgfh6mY8d6c0xtO2xqIYK8n/
qW9CfsZ2vgW1xTN/t/eZoenEnY7R1/SO4a9c3O6/dp3veCOB/X1xzgzs9h1sqr1uQKy8SVDMUJZ5uYpF8dCUFHZsnCG2UUVLcrrRQveEley+jlvVxZNLr7m1WaKa/ooFKuicXSLh
4PRWH8PocUCDz97MZhfQteLCi2Tcl+Jy3008maE9eMTliUWuxovcZJbBHOihdlNMcbttJg==

LIUDMILA LVOVNA SABININA | Fecha:2025-02-09 09:38:27 | FIRMANTE

WZ6SUTSPg6gadKyATteXbq8aqK3RHb1rQshmo3XOWTo0aSWsUEWcN7kDOH3zngjlvQ3q1/CeuoGcWyAoTObm9PCysqKKZ1n1QzL25OZlsR3SGK3FtGF9geGyTVLsB3Fv/
kRFuscKgFr+Xdl51EFaBozMaolR9rRlxFNQ+toVZEavnpnpaocqNsDX5XilRKx58h1q9Wf24SlaogpYs4Zn2AVRDPkZ+c+aomhJXUwy2BWckfuianRWWY2dlaWxIHsURV0AQXi
gjiQ+E06P1EJt611U2W5CFVdp2EQBqp+gmnMwNdqBXnkr77aDpnGDOCZkaQSwGbuH9luZaJIsNlgw==

ANTONIO DANIEL RIVERA LOPEZ | Fecha:2025-02-13 19:54:48 | FIRMANTE

TfdlrUTw1nWTkduJnVHfx5P5kBWwZ2UWZOP4xfDrdz2j4ZCPGxynlJBLc4nXIEiH5DithYiCVCrhaA41YmwyGHGL/hmz54RBAPnvdmRbjVFJFDig9tVdX8a2r2YInmHa4uB3o6z6u
NjToRYADIFHGgrjefn6BPDt8odtqX6Ei3THVf/9gVS/IzoErG2W+jbOUXjVbwlfMj5bTuFuyMzfseAmLddOUL12qS86BxZ+WqP90foj8X9Jm1cqIm2XYr4nF/zo2y97tuAe0YS66iCb9
DDbjJM94gJ3Pcv2/YQ86t0xf/2ktZgLkOtelzuRiYnlNphblFfzWnDfyp4V3lpw==

JOSE ALBERTO HERNANDEZ AGUILAR | Fecha:2025-02-14 14:17:19 | FIRMANTE

Z+v1CWdhN2kd6bluz5T7RCfyqerp41BPggpjmwTnhoEdK1H7cJFd/nOO6vV9RwnpZGgPqvWzyTe8hNbM0z8+Ffoymyr0ywNBdrRE2MzZvhE8y/ndnNtOVsMEj80yy8LuhW1qw
3xgQNA7TsSYVXD5n00na58F/Xc6lBLo0YnNmal9qCZqU8iw28Gp0LEWuPxZyGgdpofXzP/Xcrprv4FhJ9DXEg8iN3KNw4VTj4voUfmpqg0gRAWYWPQOtQbm7H6KboeEETatJ
GdhSlirtu0VjH69jWO6Xj1g3YK97SINcaEOb1RNTLSHHUpX4+crghTToQLCOofUANeX0mHGZBzg==

ULISES MORALES FUENTES | Fecha:2025-02-19 12:10:18 | FIRMANTE

s7zub1PKalli6KLLRjRDYxRNwbnc7hZHx3Njvx1DFNGnFcggbOHOXJhCVEdhWl6QDyxtHHwzggkky6a9Wlx68B8Hf9Nt7qFPkG3Poz38vi/pejkXOcwCSZfYkXI6f7+FFeAze0vu
UaROHsucVbP7cvdthO4f5a9PR3wy9nTG5JECVnB+d/fha6iHiJMYlthVcwfiAw9Ms1MrMO/HiEPfd4eruBD2Esn3Gn1/XNcJmfRcpRSNOfmMkL88BRMMQN4eE2TYkDYeYSB
mYM+bTgo0tctjKuk5laWUTfoaV7rUHjcrngkZsWNTFvz+OxXx+DzjskQ9MwVBlwg9Mw==

Puede verificar la autenticidad del documento en la siguiente dirección electrónica o
escaneando el código QR ingresando la siguiente clave:



yqF9k5E40

<https://efirma.uaem.mx/noRepudio/aAZc041D5aWH3lDgfUYLWnJkWXyirVp>



UAEM
RECTORÍA
2023-2029