Tesis presentada para la obtención del título de
Magíster en Investigación de Operaciones

Análisis y estudio de Complejidad del Problema de Fragmentación de Grafos

Autora: Lic. Natalia Castro

Director Académico: Dr. Ing. Pablo Romero

Directores de Tesis: Dr. Ing. Franco Robledo
Dr. Ing. Pablo Romero

Tribunal: Dra. Simone Martins
Dr. Ing. Claudio Risso
Dr. Gerardo Rubino

Facultad de Ingeniería

Universidad de la República

19 de Diciembre de 2017

Para Rodrigo y Katia.

Agradecimientos

Gracias...

a Pablo y a Franco; por haber hecho posible la realización de este trabajo, por toda su dedicación y correcciones, por saber motivarme, por escucharme cuando lo necesité y por darme ánimo en los momentos difíciles.

a Mathias, por ayudarme a dar el paso inicial en el camino de regresar al estudio.

a todos los profesores; que han sabido despertar mi curiosidad, me han ayudado a enfrentarme a nuevos desafíos y me han permitido aprender muchísimas cosas nuevas.

a todos los familiares y amigos, que de diversas formas me dieron ánimo y me apoyaron durante la realización de este trabajo. Particularmente, a Analía, Carolina, Ana, Mariana, Fabián, Sylvia, Álvaro, Chabela y una mención especial para mi padre.

a Katia, porque ser su madre cambió mi vida. Me hizo reflexionar mucho sobre cual es el ejemplo que le quiero dar. Volver a estudiar tuvo que ver en gran parte con eso. Quiero que algun día ella pueda elegir, sin tener miedo, una carrera que le apasione y la haga feliz.

a Rodrigo, porque desde el primer día me ha dado su apoyo incondicional en todo lo que hago y porque sin su sostén esta tesis no hubiera sido posible.

Resumen

El objeto de estudio de esta tesis es un problema de optimización combinatoria denominado Problema de Fragmentación de Grafos (GFP). Inspirado por el modelado de epidemias su aplicación puede extenderse a otras clases de desastres, por ejemplo, catástrofes naturales. En esta tesis se estudia el Problema de Fragmentación de Grafos desde el punto de vista de sus propiedades teóricas.

El sistema que será afectado se modela como una red, donde un nodo expuesto al desastre inmediatamente lo propaga a sus vecinos. El objetivo del GFP es elegir una estrategia de inmunización de forma en que se minimice el número esperado de muertes causadas. Se presenta el problema mostrando además su relación con el modelo de epidemias clásico SIR y con otro problema teórico de grafos denominado Component Order Connectivity problem. Se prueba que el problema de decisión asociado al GFP pertenece a la clase de problemas $\mathscr{NP}$-completos y también un resultado fuerte de inaproximabilidad que muestra que no existe un algoritmo aproximado de tiempo polinomial para la resolución del GFP con factor menor a $\frac{5}{3}$, a menos que $\mathscr{P} = \mathscr{NP}$.

Por último en contraste con el anterior resultado de inaproximabilidad se hallan estrategias de tiempo polinomial para la mejor inmunización en algunas familias especiales de grafos, a saber, ciclos, grafos acíclicos y grafos bipartitos.

Palabras Clave: Problema de Fragmentación de Grafos, Complejidad Computacional, Algoritmos de Aproximación.

Esta tesis ha dado como producto las siguientes publicaciones:

- Aprile, M., Castro, N., Robledo, F., Romero, P. “Analysis and Complexity of Node-Immunization under Natural Disasters.” Proceedings of the 13th International Conference on Design of Reliable Communication Networks, 8-10 march, 2017. Munich, Germany. Pages 1 - 8. URL: http://ieeexplore.ieee.org/document/7993440/
- Castro, N., Ferreira, G., Robledo, F., Romero, P. “Graph Fragmentation Problem for Natural Disaster Management.” The Third International Conference on Machine learning, Optimization and big Data - MOD 2017, 14 - 17 september, 2017. Volterra, Tuscany, Italy. Proceedings to be published in Lecture Notes in Computer Science, Springer, Vol. 10710, DOI 10.1007/978-3-319-72926-8, 2018.

A su vez, se ha sometido a referato el siguiente artículo:

- Aprile, M., Castro, N., Ferreira, G., Piccini J., Robledo, F., Romero, P. “Graph Fragmentation Problem: Analysis and Synthesis.” Submitted, International Transactions in Operational Research. 2017.

Índice

Capítulo 1

Fundamentos

En este capítulo se presentan conceptos necesarios para introducir el problema de interés abordado en esta tesis. En la Sección 1.1 se definen conceptos básicos de la Teoría de Grafos [12]. En las Secciones 1.2 y 1.3 se realiza una pequeña recopilación de definiciones y resultados necesarios referentes a la Teoría Computacional [11] y a Algoritmos de Aproximación [25] respectivamente.

1.1 Teoría de grafos

El nacimiento del concepto de grafo se remonta al siglo XVIII cuando Leonhard Euler modeló el famoso problema de los puentes de Königsberg, que consistía en encontrar un camino que recorriera los siete puentes del río Pregel en la ciudad de Königsberg de modo que se recorrieran todos los puentes pasando una sola vez por cada uno de ellos.

A partir de Euler el modelado mediante grafos fue desarrollándose y convirtiéndose en una herramienta de trabajo para ciencias tan diferentes como la Física, la Química o la Economía entre otras. La teoría de grafos está íntimamente relacionada con otras ramas de la Matemática, como por ejemplo la Teoría de Conjuntos, el Análisis Numérico, la Probabilidad y es la base conceptual en el tratamiento de problemas combinatorios.

La eficacia de los grafos se basa en su gran poder de abstracción y su capacidad de representar relaciones (de orden, precendencia, etc) lo que facilita el modelado y la resolución de problemas. Hay muchas situaciones en las cuales el modelado más conveniente de los datos se realiza utilizando grafos, por ejemplo la representación de carreteras, calles, redes de telecomunicaciones, redes eléctricas, internet, planificación de tareas, etapas de un proceso industrial. Gracias a la Teoría de Grafos se han desarrollado

gran variedad de algoritmos y métodos de resolución eficaces que permiten tomar mejores decisiones en diversos problemas.

A continuación se presentan algunas definiciones y propiedades que serán utilizadas a lo largo de esta tesis.

Definición 1.1 *Un **grafo** es un par $G = (V,E)$, donde $V \neq \emptyset$ es el conjunto de vértices o nodos y $E \subseteq V \times V$ es el conjunto de aristas o enlaces, que conectan vértices entre sí. Si $(u,v) \in E$ significa que el vértice u está conectado con el vértice v. Dependiendo de si el orden de los vértices en las aristas importa se tiene:*

- ***Grafo dirigido**: el vértice u conectado con el vértice v no implica que el vértice v esté conectado con el vértice u, es decir, $(u,v) \neq (u,v)$.*
- ***Grafo no dirigido**: para toda arista $(u,v) \in E$ se cumple $(u,v) = (u,v)$.*

Definición 1.2 *Un **subgrafo** $G' = (V',E')$ es un grafo donde $V' \subseteq V$ y $E' \subseteq E$.*

Definición 1.3 *Un grafo G es **simple** si es un grafo no dirigido sin lazos ($(u,u) \notin E$)).*

Definición 1.4 *Dos **vértices** $u,v \in V$ son **adyacentes** si $e = (u,v) \in E$. Se dice que la arista e es incidente en los vértices u y v.*

Definición 1.5 *El **grado** de un vértice v es la cantidad de aristas que inciden en él y se denota $gr(v)$.*

Definición 1.6 *Un **camino** es una secuencia ordenada de vértices adyacentes $P = (v_1, \ldots, v_n)$. Tal **camino** es **simple** si no repite vértices.*

Definición 1.7 *Sea $G = (V,E)$ un grafo simple. Un grafo es **conexo** si todo par de vértices $u,v \in V$ admiten un camino que los une.*

Recuérdese que un conjunto S es maximal respecto de una determinada propiedad P si S satisface P y todo conjunto R que contiene propiamente a S no satisface P. Luego,

Definición 1.8 *Una **componente conexa** de un grafo es un subgrafo conexo maximal.*

Definición 1.9 *Un **ciclo** es un camino simple donde el vértice inicial y el final son el mismo.*

Definición 1.10 *Un **ciclo hamiltoniano** en un grafo es un ciclo que contiene todos sus vértices. Diremos que G es un grafo hamilltoniano si admite un ciclo hamiltoniano.*

Definición 1.11 *Un grafo se dice que es **acíclico** si todos sus posibles caminos son simples (no existen ciclos).*

Definición 1.12 *Un **árbol** es un grafo no dirigido y acíclico.*

Definición 1.13 *Un **bosque** es un grafo donde cada componente conexa es un árbol, es decir es un conjunto de árboles.*

En el capítulo 4 se trabajará con grafos bipartitos por ello se presentan las siguientes definiciones y resultados:

Definición 1.14 *Un grafo $G = (V,E)$ es **bipartito** si $V = V_1 \cup V_2$ y su conjunto de aristas verifica que $E \subseteq V_1 \times V_2$.*

Definición 1.15 *Un **conjunto independiente de vértices** en un grafo $G = (V,E)$ es un conjunto $S \subseteq V$ tal que para ningún par de vértices en S existe alguna arista que los conecte.*

Definición 1.16 *Un **conjunto independiente de aristas** en un grafo G es un conjunto de aristas sin vértices en común.*

Definición 1.17 *Un **emparejamiento** en un grafo G es un subconjunto $A \subseteq E$ de aristas independientes. Si el emparejamiento tiene el mayor número posible de aristas se denomina **emparejamiento máximo** o de cardinal máximo.*

Definición 1.18 *Un **cubrimiento de vértices** de un grafo G es un conjunto de vértices tales que cada arista del grafo es incidente en al menos un vértice del conjunto.*

Teorema 1.1 (Teorema de König) *En todo grafo bipartito, el cardinal del emparejamiento máximo coincide con el cubrimiento de vértices de tamaño mínimo.*

Teorema 1.2 (Teorema de Menger) *Sean $G = (V,E)$ un grafo y $A,B \subseteq V$ subconjuntos de vértices. El mínimo número de nodos necesarios para separar A de B en G es igual al máximo número de caminos nodo-disjuntos de A a B en G.*

Las demostraciones correspondientes a estos teoremas pueden ser consultadas en [12].

1.2 Complejidad Computacional

La teoría de la complejidad computacional estudia los recursos requeridos durante la ejecución de un algoritmo para resolver un problema. Los recursos comúnmente estudiados son el tiempo y la memoria utilizada para poder resolver el problema. Algunos problemas que son computables precisan un uso de recursos demasiado grande para permitir su cálculo en la práctica. Es entonces de particular interés poder distinguir problemas que son tratables de otros que no lo son.

Existen diferentes tipos de problemas computacionales, por ejemplo, problemas de decisión, problemas de conteo, problemas de optimización, etc.

Definición 1.19 *Un **problema de decisión** es aquel donde dada una entrada se debe dar una "respuesta" del tipo SI o NO. Esto corresponde a verificar si la entrada satisface o no una cierta propiedad.*

La separación en clases de complejidad es una manera de clasificar los problemas, para esto es usual tomar como referencia la máquina de Turing. Una máquina de Turing es un dispositivo que manipula símbolos sobre una cinta de acuerdo a una tabla de reglas. Esta puede ser adaptada para simular la lógica de un algoritmo. A partir de esto se obtienen las siguientes definiciones para dos grandes clases:

- $\mathscr{P}$ es la clase de problemas que admiten solución en tiempo polinomial con la entrada, por una máquina de Turing determinista.
- $\mathscr{NP}$ es la clase de problemas de decisión que admiten verificación de una instancia positiva en tiempo polinomial.

A modo de ejemplo se muestra a continuación un problema perteneciente a la clase $\mathscr{NP}$.

Ejemplo 1.1 *Dado un grafo simple G, considérese el problema de decidir si G es hamiltoniano. El candidato a solución es un subgrafo $C \subseteq G$. Se observa que el problema del ciclo hamiltoniano pertenece a la clase $\mathscr{NP}$. De hecho, basta con verificar que C es un ciclo que contiene a todos los vértices de G.*

En la Teoría de la Complejidad, se utiliza como una herramienta central la noción de Reducción de problemas, que corresponde a la posibilidad de transformar en tiempo polinómico un problema en otro. El objetivo es poder clasificar los problemas de acuerdo a su dificultad relativa. Un problema **A** es reducible a otro **B** si existe una función inyectiva $f : \mathbf{A} \to \mathbf{B}$ que traduce toda instancia de **A** en alguna instancia de **B** en una cantidad

polinomial de operaciones. Intuitivamente, si **A** se reduce a **B**, entonces **B** es al menos tan difícil como **A**. Si **A** y **B** fuesen mutuamente reducibles, entonces su dificultad sería la misma.

La reducción entre algunos problemas se da por simple equivalencia. Para ejemplificar, considérense los dos problemas que siguen:

Ejemplo 1.2 *Para un grafo $G = (V,E)$ y un entero k,*

- *el **Problema del Conjunto Independiente** consiste en determinar si existe un conjunto de vértices $S \subseteq V$ con $|S| \geq k$ y tal que para cada arista como máximo uno de sus extremos esté en S.*
- *el **Problema del Cubrimiento de Vértices** consiste en determinar si existe un conjunto de vértices $S \subseteq V$ con $|S| \leq k$ y tal que para cada arista por lo menos uno de sus extremos esté en S.*

Obsérvese que es posible probar que S es un conjunto independiente si y sólo si $V \setminus S$ es un cubrimiento de vértices. Un bosquejo de la prueba es,

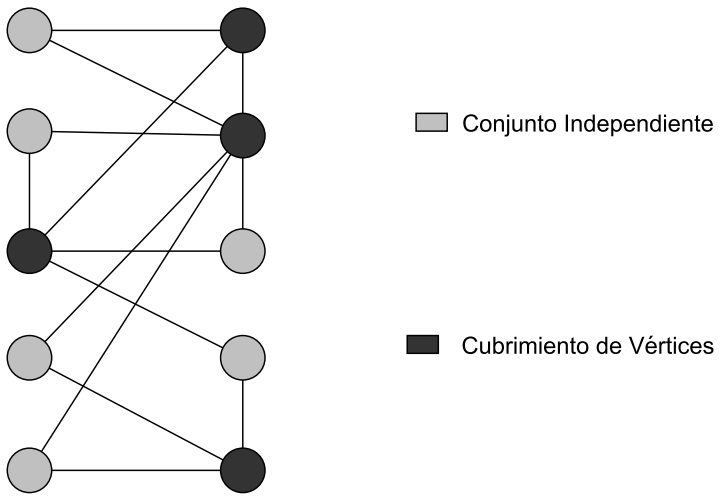

Fig. 1.1 Conjunto Independiente vs. Cubrimiento de Vértices

- *Si S es un conjunto independiente. Sea (u,v) una arista, como S es independiente, entonces $u \notin S$ o $v \notin S$. Luego, $u \in V \setminus S$ o $v \in V \setminus S$ y esto implica que $V \setminus S$ es un cubrimiento de vértices.*

- *Si $V \setminus S$ es un cubrimiento de vértices. Sean dos nodos $u, v \in S$. Obsérvese que $(u,v) \notin E$ pues $V \setminus S$ es un cubrimiento de vértices. Entonces, no hay nodos de S que estén unidos por una arista y S es un conjunto independiente.*

Se probó entonces que el problema del Conjunto Independiente y el del Cubrimiento de Vértices son igualmente difíciles, por ser reducibles polinomialmente uno al otro. Es conocido que ambos pertenecen a la clase de problemas $\mathscr{NP}$-completos [14] que se definirá a continuación.

Se definen a su vez las siguientes clases de problemas,

Definición 1.20 *Para una clase $\mathscr{C}$ cualquiera,*

- *Un problema es $\mathscr{C}$-difícil si es al menos tan difícil como cualquier problema de $\mathscr{C}$.*
- *Un problema es $\mathscr{C}$-completo si es $\mathscr{C}$-difícil y además pertenece a la clase $\mathscr{C}$.*

Aplicando estas definiciones a la clase $\mathscr{NP}$ se obtiene que, un problema es $\mathscr{NP}$-completo si pertenece a la clase $\mathscr{NP}$ y todo problema en $\mathscr{NP}$ se reduce a él en tiempo polinomial. Un problema es $\mathscr{NP}$-difícil si existe un problema $\mathscr{NP}$-completo que se reduce a él.

Utilizando la noción de reducción, se desarrolla una técnica para probar que un problema X es $\mathscr{NP}$-Completo. Se prueba, primero, que el problema X pertenece a la clase $\mathscr{NP}$. Luego se elige un problema A que sea $\mathscr{NP}$-difícil y se halla una reducción $f : A \to X$. En 1971, Stephen Cook demostró que el problema MAX-SAT es $\mathscr{NP}$-Completo [6], lo que definió el primer problema de esta clase. En 1972, Richard Karp utilizó la reducibilidad de Stephen Cook para dar la lista de los primeros 21 problemas $\mathscr{NP}$-Completos [13]. Uno de estos 21 problemas es el del Ciclo Hamiltoniano mencionado en el Ejemplo 1.1.

De las definiciones presentadas es claro que $\mathscr{P} \subseteq \mathscr{NP}$. La pregunta de si $\mathscr{P} = \mathscr{NP}$, todavía está abierta y es fundamental en la Teoría de la Complejidad Computacional. Aún cuando no está demostrado, hay una inclinación a creer que las clases $\mathscr{P}$ y $\mathscr{NP}$ no son iguales. Esto implica que existirían muchos problemas para los que no es posible determinar un algoritmo de tiempo polinomial para su resolución y por lo tanto, recurrir a identificar casos particulares más sencillos de resolver, métodos heurísticos o algoritmos aproximados para resolverlos son opciones frecuentes.

1.3 Algoritmos de Aproximación

En la sección anterior se presentaron los problemas de decisión y sus clases de complejidad, para ellos todo instancia tendrá una respuesta del tipo SI o NO. Este tipo de problemas son inapropiados para introducir conceptos de aproximación, precisamente porque la aproximación requiere tener la posibilidad de estar más cerca o más lejos de un cierto valor y esto sólo es posible cuando la respuesta al problema se encuentra dentro de un cierto rango. Es necesario entonces presentar un nuevo tipo de problema, el problema de optimización.

Definición 1.21 *Un **Problema de Optimización Combinatoria** consiste en la mininización de una función $f(x)$, donde x pertenece a un conjunto finito denominado región factible.*

Obsérvese que todo problema de maximización en un conjunto finito es también un Problema de Optimización Combinatoria, pues es equivalente la maximización de $f(x)$ a la minimización de $-f(x)$. El objetivo de un problema de optimización es para cualquier instancia dada hallar la solución que alcanza el objetivo (minimización o maximización) para el costo. Para un problema de optimización anotaremos $OPT(I)$ para referirnos al valor de una solución óptima para el problema en consideración para la instancia I. Análogamente a la clase $\mathscr{NP}$ para los problemas de decisión es posible definir la clase de problemas de $\mathscr{NP}$-optimización como sigue,

Definición 1.22 *Un problema es de $\mathscr{NP}$**-optimización** si es un problema de optimización que además verifica:*

- *la factibilidad de una solución puede ser reconocida en tiempo polinomial,*
- *todas las soluciones tienen largo polinomial con respecto a la entrada,*
- *el costo de una solución puede ser calculado en tiempo polinomial.*

Notése que todo problema de $\mathscr{NP}$-optimización puede convertirse en un problema de decisión de la clase $\mathscr{NP}$. Por ejemplo, para un problema de minimización se pregunta, ¿se cumple que $OPT(I) \leq q$? La solución óptima sirve como certificado de un respuesta SI y por lo tanto el problema análogo de decisión estará en la clase $\mathscr{NP}$.

Muchos problemas conocidos son computacionalmente difíciles de resolver, particularmente $\mathscr{NP}$-difíciles, lo que implica que no existe algoritmo de resolución de tiempo polinomial, a menos que $\mathscr{P} = \mathscr{NP}$. Luego, es necesario encontrar otras

alternativas de resolución. En algunos casos una opción viable es obtener soluciones aproximadas. Idealmente, se quiere tener una garantía de que tan cerca del óptimo estará dicha solución. A un algoritmo para el que existe una garantía respecto de la cercanía con el valor óptimo lo llamamos Algoritmo de Aproximación.

Definición 1.23 *Para un problema de minimización un **Algoritmo de Aproximación** de factor $\alpha > 1$ es un algoritmo que para todas las instancias I del problema produce una solución con costo $ALG(I)$ cuyo valor es como máximo $\alpha.OPT(I)$. Esto es,*

$$\frac{ALG(I)}{OPT(I)} \leq \alpha.$$

Análogamente puede darse la definición para un problema de maximización. La calidad de la solución dada por el algoritmo de aproximación está dada por el valor α, que se denomina **radio o factor de aproximación**.

Típicamente las reducciones de tiempo polinomial entre algoritmos transforman soluciones óptimas en soluciones óptimas, sin embargo, no preservan soluciones cercanas al óptimo como lo son las obtenidas por un algoritmo de aproximación. Ciertamente todos los problemas $\mathscr{NP}$-completos son igualmente difíciles desde el punto de vista de hallar soluciones óptimas pero, hallar soluciones aproximadas puede ser más fácil o más difícil dependiendo del problema en cuestión. A la hora de trabajar con reducciones polinomiales existen distintas posibilidades, se define a continuación una clase de reducción que preserva el factor de aproximación.

Definición 1.24 *Sean P_1 y P_2 dos problemas de minimización. Una reducción que preserva el factor de aproximación de P_1 a P_2 consiste en dos algoritmos de tiempo polinomial, f y g tales que:*

- *para cada instancia I_1 de P_1, $I_2 = f(I_1)$ es una instancia de I_2 tal que $OPT(I_2) \leq OPT(I_1)$, y*
- *para cada solución t de I_2, $s = g(I_1,t)$ es una solución de I_1 tal que $obj(I_1,s) \leq obj(I_2,t)$.*

Este concepto de reducibilidad ha sido utilizado para mostrar que una variedad de problemas $\mathscr{NP}$-difíciles pueden ser reducidos unos a otros preservando la calidad de la aproximación. En otras oportunidades se deseará probar que un problema no puede ser aproximado a menos de cierto factor, como es el caso del problema de estudio de esta Tesis. A continuación se

propone un ejemplo para mostrar un método general utilizado para probar resultados de inaproximabilidad.

Ejemplo 1.3 *Considérese el* ***Problema del Viajante de Comercio (TSP)****. Dado un conjunto de ciudades con costos de transporte entre ellas, hallar el recorrido (tour) más barato que permita visitarlas todas una vez regresando al punto de partida. Formalmente, en términos de grafos, se enuncia: dado un grafo completo no dirigido* $G = (V,E)$ *con costos enteros no negativos* $c(u,v)$ *para cada arista* $(u,v) \in E$*, hallar un ciclo hamiltoniano en G de costo mínimo.*

A continuación se prueba que el TSP es un problema inaproximable en su caso general, lo que se enuncia el siguiente Teorema:

Teorema 1.3 *Si* $\mathscr{P} \neq \mathscr{NP}$*, para cualquier constante* $\alpha \geq 1$*, no existe algoritmo de aproximación de tiempo polinomial con radio* α *para el TSP.*

Demostración: La idea consiste en hacer una reducción desde el problema del Ciclo Hamiltoniano presentado en el Ejemplo 1.1.
Sea $G = (V,E)$ *una instancia del problema del Ciclo Hamiltoniano. Se construye* $G' = (V,E')$ *un grafo completo con costos para cada arista* (u,v) *definidos por:*

$$c(u,v) = \begin{cases} 1, \text{ si } (u,v) \in E \\ \alpha.|V| + 1, \text{ si no} \end{cases}$$

La figura muestra un posible grafo G con su respectivo G′,

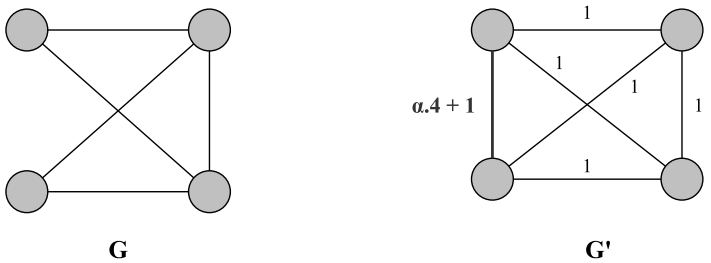

Fig. 1.2 Construcción de G'

Obsérvese que:

- *Si G tiene un ciclo hamiltoniano H entonces* (G',c) *contiene un tour de costo* $|V|$*.*

- *Si G no tiene un ciclo hamiltoniano, entonces todo tour T debe utilizar una arista que no pertenece a E y por lo tanto el costo del tour cumple:*

$$c(T) \geq (\alpha|V|+1)+(|V|-1) = (\alpha+1).|V|$$

Existe un salto de $(\alpha+1)$ *entre los costos de los tours que sólo usan aristas de G y aquellos que no. Entonces, un algoritmo de aproximación de factor* α *del TSP en* G' *permite hallar, si existe, un ciclo hamiltoniano en G. Es decir, que un algoritmo de tiempo polinomial para el TSP permitiría resolver en tiempo polinomial el Problema del Ciclo Hamiltoniano, que como ya se mencionó en la Sección 1.2 pertenece a la clase de problemas* $\mathscr{NP}$*-completos. Esta contradicción implica que el Problema del Viajante de Comercio es inaproximable.*

□

Capítulo 2

Problema de Estudio

El tema de estudio de esta tesis es un problema teórico de partición de grafos inspirado por el modelado de epidemias. En la Sección 2.1 se motiva la definición del problema y se describe brevemente el proceso de investigación. En la Sección 2.2 se define el Problema de Inmunización en Redes, del cual se derivará el Problema de Fragmentación de Grafos, que se presentará formalmente en la Sección 2.3.1. En la Sección 2.3.2 se presenta un problema relacionado con el del problema de Tesis.

2.1 Motivación y Trabajos relacionados

Es deseable que la investigación académica intente buscar soluciones para las problemáticas que surgen en la sociedad. Desde el siglo pasado existe un gran interés por la aplicación de métodos cuantitativos en el ámbito de la biología. Los modelos matemáticos han cobrado gran importancia, por ejemplo, en el área de la epidemiología, donde los modelos se utilizan como herramienta conceptual con la cual poder describir, explicar y hasta en algunos casos, predecir comportamientos o resultados específicos. Por su parte las pérdidas causadas por las catástrofes naturales van en aumento desde los años 80. Los riesgos de desastres están aumentando también, principalmente, como resultado de la creciente exposición de las personas y los activos a fenómenos naturales extremos.

Se puede observar que tanto los procesos epidémicos como los desastres naturales tienen puntos en común a la hora de utilizar modelos cuantitativos. Pensando en términos de prevención, en ambos casos la intención será la de fragmentar la zona que resultará dañada de forma de conseguir minimizar el impacto global. Esta es la idea que define el problema estudiado en esta Tesis, denominado Problema de Fragmentación de Grafos (GFP).

El modelo matemático que define el GFP fue propuesto por primera vez en 2015, en el artículo [20]. En dicha publicación se propone una resolución golosa y se brindan los primeros resultados experimentales. En 2016 se realizó una primera prueba de la complejidad computacional del problema y se mostraron las primeras propiedades como problema combinatorio [21], [23], [22]. A principios de 2017 se logró demostrar la inaproximabilidad del problema a factores inferiores a $\frac{5}{3}$, a menos que $\mathscr{P} = \mathscr{NP}$ y en contraste, se desarrolló un algoritmo para la resolución cuando las redes representadas no poseen ciclos [2]. Posteriormente, en [5], se formuló un modelo exacto para la resolución del problema, que es útil para casos de prueba de tamaño pequeño por correr en tiempo exponencial.

A continuación se presenta el Problema de Inmunización en Redes, modelo proveniente del mundo de la epidemiología, del cual derivó el nacimiento del Problema de Fragmentación de Grafos. Luego, en la Sección 2.3 se hace la presentación formal del GFP y se presenta también su relación con otro problema ya existente en la literatura.

2.2 Problema de Inmunización en Redes

Probablemente el modelo más estudiado en el modelado de epidemias sea el clásico SIR. En este los individuos se dividen en tres clases: vulnerables (S, "susceptible"), infectados (I, "infected") o eliminados (R, "removed"). La última clase representa a aquellos individuos que son inmunes a la enfermedad por haberse recuperado de ella. Se supone que los individuos infectados tienen contactos aleatorios entre sí y con los individuos pertenecientes a las otras clases a una tasa promedio β. La recuperación de la enfermedad se da a una tasa promedio γ. Si un individuo vulnerable tiene contacto con uno infectado, el vulnerable se transforma en infectado.
Si consideramos una población de gran tamaño, con n individuos y $s(t) = \frac{S(t)}{n}$, $i(t) = \frac{I(t)}{n}$ y $r(t) = \frac{R(t)}{n}$ son las respectivas proporciones de individuos en cada clase en tiempo t, el modelo epidémico SIR puede describirse mediante el siguiente sistema no lineal de ecuaciones diferenciales:

$$\frac{\partial s}{\partial t} = -\beta is, \frac{\partial i}{\partial t} = \beta is - \gamma i, \frac{\partial r}{\partial t} = \gamma i \tag{2.1}$$

La principal deficiencia del modelo SIR es que proviene de asumir una población infinita con contactos aleatorios y probabilidades de contagio idénticas para todos los individuos. Incluso sus propios creadores, Lowell Reed y Wade Frost, dudaban del valor

científico del modelo por lo que decidieron no publicarlo [9].
Diversos autores consideran que el modelo SIR no es suficientemente realista. Como indican Fromont et al. [10] en las poblaciones reales la estructura del grafo que representa las relaciones entre individuos será altamente heterogénea y como hace notar Andersson [1] hasta un pequeño cambio en la homogeneidad puede tener importantes efectos en la propagación de una epidemia. Otros modelos han sido propuestos, por ejemplo, por Santhanam et al. [24] donde se utilizan redes en las que los nodos representan a los individuos y la epidemia se propaga por las aristas. Autores como Newman et al. [18], [16], [17] o Ball et al. [4], [3] reafirman la idea de que los individuos no deben representarse con contactos aleatorios y por lo tanto la representación mediante modelos más realistas (población finita, contactos entre vecinos, etc.) que utilizan redes se vuelve necesaria.

Se define entonces el Problema de Inmunización en Redes, originalmente propuesto en la tesis del Dr. J. Piccini [19]. El estudio analítico del problema es complejo, por lo tanto en la publicación [20] se utiliza un tratamiento mediante heurísticas. También allí se introduce un caso extremal denominado Problema de Fragmentacion de Grafos (GFP), que es el objeto de estudio de esta Tesis y se presenta formalmente en la Sección 2.3.

Sea $G = (V,E)$ un grafo simple donde los nodos representan individuos y las aristas representan las relaciones entre ellos. Tomamos el tiempo dentro del dominio de los números naturales, $T = \mathbb{N}$ y para $t = 0$ se cumple:

- Un subconjunto $U \subset V$ es inmunizado (eliminado) y un proceso epidémico basado en el modelo SIR toma lugar en el subgrafo G' inducido por $V' = V \setminus U$.
- Un nodo $x_0 \in V'$ es elegido con equiprobabilidad, x_0 es el primer nodo infectado.
- Los nodos restantes en $V' \setminus \{x_0\}$ son vulnerables a la enfermedad.

En tiempo t cada nodo infectado $v \in V$ propaga la enfermedad a cada uno de sus vecinos vulnerables w con un cierto perfil de probabilidad $p(t,\mu,v,w)$. Esta probabilidad p depende del portador v, del nodo vulnerable w, del tiempo t y del nivel de virulencia μ. Se asume que dicha probabilidad tiende a uno cuando el nivel de virulencia tiende a infinito, es decir:

$$\lim_{\mu\to\infty} p(t,\mu,v,w) = 1 \tag{2.2}$$

Una vez que un nodo v vulnerable (S) se infecta, pasa a pertenecer a la clase de nodos infectados (I) por un tiempo determinado por una variable aleatoria X_v con media finita. Luego, se convierte en vulnerable nuevamente. Sea $\{I_t\}_{t\in\mathbb{N}}$ el proceso estocástico que

cuenta el número de individuos infectados. El Problema de Inmunización en Redes es el siguiente problema de optimización combinatoria,

Definición 2.1 *El objetivo del **Problema de Inmunización en Redes** es elegir un conjunto de nodos U para inmunizar, que minimice el máximo para el proceso* $\{I_t\}_{t\in\mathbb{N}}$*, sujeto a una restricción de presupuesto B:*

$$\begin{aligned} &\min_{U\subseteq V} \max_{t\in\mathbb{N}} E(I_t) \\ &s.t. |U| \leq B \end{aligned} \tag{2.3}$$

Cuando μ es infinitamente grande, la epidemia se transforma en una pandemia donde los contactos propagan la enfermedad con seguridad. Como ya se mencionó antes este problema se denomina Problema de Fragmentación de Grafos (GFP) y se introdujo en [20]. Allí se estudian propiedades teóricas elementales pero no se caracteriza la complejidad computacional del problema. En la publicación [2], demostramos que el GFP pertenece al conjunto de problemas de optimización $\mathscr{NP}$-Difíciles, brindando también un resultado de inaproxi-mabilidad, y optimalidad en grafos acíclicos. El lector puede encontrar una síntesis de los avances obtenidos con respecto al problema en [5]. En dicha publicación se incluye una resolución exacta de tiempo exponencial y conexiones con la Teoría de Juegos.

2.3 Problema de Fragmentación de Grafos

2.3.1 Definición del Problema

Consideremos una población representada por un grafo $G = (V,E)$ y un presupuesto B natural tal que $0 \leq B \leq |V|$. Se eligen B nodos para inmunizar, es decir, se eliminan dichos B nodos del grafo G y se obtiene un subgrafo G', de forma que los nodos eliminados no pueden ser afectados por el desastre. La naturaleza elige un nodo v de G' aleatoriamente para ser afectado y mata a todos los nodos pertenecientes a la misma componente conexa que v. Obsérvese que este es un caso particular del Problema de Inmunización en Redes, mencionado en la Sección 2.2, en donde μ es infinitamente grande, $p = 1$ y por lo tanto todos los contactos propagan la enfermedad.

La idea es minimizar el número esperado de muertes. Si G' tiene n nodos y k componentes conexas de ordenes $n_1,\ldots,n_k$, la probabilidad de elegir la componente i es $p_i = \frac{n_i}{n}$. Luego, el número esperado de muertes es $E(G') = \sum_{i=1}^{k} n_i p_i$. El objetivo del

Problema de Fragmentación de Grafos (GFP) es elegir el conjunto de nodos a inmunizar de forma que se minimice el número esperado de muertes, esto es:

$$\begin{aligned} &\min_{U \subseteq V} \sum_{i=1}^{k} \frac{n_i^2}{n}, \\ &s.t. |U| = B \end{aligned} \tag{2.4}$$

Notése que para una instancia fija (G,B) el denominador n es constante, por lo tanto el problema resulta ser el de minimizar la norma euclídea del vector $\vec{n} = (n_1, \ldots, n_k)$, lo que se denomina Minimización Restringida de la Norma Euclídea (CENM):

$$\begin{aligned} &\min_{U \subseteq V} \| \vec{n}_{G-U} \|^2, \\ &s.t. |U| = B, \end{aligned} \tag{2.5}$$

donde $\vec{n}_{G \setminus U} = (n_1, \ldots, n_k)$ es el vector de órdenes de las componentes conexas de $G' = G \setminus U$.

Intuitivamente elegir el mejor conjunto de nodos para inmunizar parece una tarea difícil. El GFP surgió justamente con el fin de formalizar la evidencia de que los problemas de inmunización en epidemiología son de alta complejidad. En el próximo Capítulo se establece formalmente la complejidad computacional del problema.

2.3.2 Problema relacionado

Existe en la literatura una relajación natural al Problema de Fragmentación de Grafos, que para una misma instancia de entrada con grafo G y presupuesto B tiene como objetivo eliminar B nodos de G de manera que el tamaño de la mayor componente conexa del grafo resultante sea el mínimo posible. [8] Este problema toma la abreviatura COC, proveniente de su denominación en inglés "Component Order Connectivity Problem".

Se puede apreciar que en el COC se desea calcular sólo el número de muertes que ocurren en el peor caso, mientras que en el GFP se desea el valor esperado. Esta es una métrica alternativa para analizar la vulnerabilidad de la red. La siguiente proposición muestra la relación entre los dos problemas.

Proposición 2.1 *Si GFP(G,B) y COC(G,B) representan los valores óptimos de los respectivos problemas, se cumple:*

$$GFP(G,B) \leq COC(G,B) \leq \sqrt{n.GFP(G,B)} \tag{2.6}$$

Demostración: Sea $S \subseteq V$ tal que $G \setminus S$ tiene k componentes conexas de tamaños $n_1 \geq \ldots \geq n_k$. Entonces, como $\frac{n_i}{n_1} \leq 1$ para todo $i = 1, \ldots, k$ se verifica que:

$$\sum_{i=1}^{k} \frac{n_i^2}{n_1} = n_1 + n_2.\frac{n_2}{n_1} + \ldots + n_k.\frac{n_k}{n_1} \leq n_1 + n_2 + \ldots + n_k = n$$

Despejando se obtiene $\sum_{i=1}^{k} \frac{n_i^2}{n} \leq n_1$. Si S es el conjunto para el cual se da el valor óptimo del COC, entonces $n_1 = COC(G,B)$ y S es una solución factible para el GFP. Luego, se cumple la primera parte de la desigualdad:

$$GFP(G,B) \leq \sum_{i=1}^{k} \frac{n_i^2}{n} \leq COC(G,B)$$

Para la segunda parte de la desigualdad, obsérvese que $n_1^2 \leq \sum_{i=1}^{k} n_i^2$ implica $n_1 \leq \sqrt{n.\sum_{i=1}^{k} \frac{n_i^2}{n}}$ Si S es el conjunto para el cual se da el valor óptimo del GFP se cumple:

$$COC(G,B) \leq n_1 \leq \sqrt{n.GFP(G,B)}$$

lo que concluye la prueba. □

Capítulo 3

Complejidad Computacional del GFP

En este capítulo se formaliza la noción de que resolver el GFP de forma óptima es una tarea difícil. Se prueba que elegir la mejor estrategia de inmunización es un problema $\mathscr{NP}$-difícil y también, que no existe un algoritmo de aproximación para el GFP de factor menor a $\frac{5}{3}$, a menos que $\mathscr{P} = \mathscr{NP}$.

3.1 Intratabilidad del GFP

Mediante una reducción a un problema de la lista de 21 problemas $\mathscr{NP}$- Completos propuestos por Richard Karp [13] se probará la intratabilidad del GFP. Para ello es necesario definir el problema de decisión correspondiente al GFP.

Definición 3.1 *Dado un grafo simple $G = (V,E)$, un entero positivo B y un número real k. La **versión de decisión del GFP** consiste en determinar si existe un conjunto de nodos U con $|U| \leq B$ tal que el número esperado de muertes en $G \setminus U$ sea como máximo k.*

Para probar que el GFP en su versión de decisión pertenece a la clase de problemas $\mathscr{NP}$-completos es suficiente probar que pertenece a dicha clase para algún valor de k. Luego, se define el problema de decisión asociado al GFP para $k = 1$, denominado problema de mejor inmunización, y se prueba su pertenencia a la clase de problemas $\mathscr{NP}$-completos.

Definición 3.2 *Dado un grafo simple $G = (V,E)$ y un entero positivo B. El problema de **mejor inmunización** consiste en determinar si existe una solución factible $G \setminus U$ para el GFP con número esperado de muertes igual a* 1.

Se elige el problema de hallar un cubrimiento de vértices de cardinal mínimo. Es bien conocido que este problema pertenece a la clase de problemas $\mathscr{NP}$-completos [11]. Se da su definición a continuación:

Definición 3.3 *Dado un grafo simple $G = (V,E)$ y un entero positivo k. El problema de hallar un* ***cubrimiento de vértices de cardinal mínimo*** *consiste en determinar si existe un conjunto de nodos U con cardinal $|U| \leq k$ y tal que toda arista es incidente a algún nodo de U.*

En el siguiente Teorema se muestra la pertenencia del problema de mejor inmunización a la clase de problemas $\mathscr{NP}$-completos, utilizando para ello el problema de cubrimiento de vértices de cardinal mínimo definido antes.

Teorema 3.1 *El problema de Mejor inmunización pertenece a la clase de problemas $\mathscr{NP}$-Completos.*

Demostración: Considérese una instancia arbitraria (G,k) para el problema del cubri-miento de vértices de cardinal mínimo. Se construye una instancia del problema de mejor inmunización (G,B) con igual grafo G y presupuesto $B = k$. Obsérvese que el GFP alcanza un número esperado de muertes igual a 1 si y sólo si todos los nodos de un grafo factible $G' = G \setminus U$ están aislados. Todos los nodos de G' estarán aislados si U es un cubrimiento de vértices con cardinal $|U| \leq k$. En consecuencia, el problema de mejor inmunización es al menos tan difícil como el problema del cubrimiento de vértices de cardinal mínimo y por lo tanto es $\mathscr{NP}$-difícil.
Para finalizar la prueba, nótese que la función objetivo del GFP puede evaluarse en tiempo polinomial. Un algoritmo eficiente, por ejemplo, consiste en aplicar iterativamente búsqueda en profundidad (*Depth First Search*) para hallar el número de nodos perteneciente a cada componente conexa. Luego, el problema de mejor inmunización pertenece a los problemas de decisión de la clase $\mathscr{NP}$ y esto implica que es $\mathscr{NP}$-Completo. □

Recuérdese que un problema de optimización combinatoria es de $\mathscr{NP}$-optimización si existen algoritmos que permitan tanto evaluar la función objetivo como verificar la factibilidad de una solución dada en tiempo polinomial [25]. A su vez para que un problema de $\mathscr{NP}$-optimización sea $\mathscr{NP}$-difícil es necesario que su correspondiente versión de decisión sea un problema perteneciente a los de la clase $\mathscr{NP}$-completos. Luego, como corolario del teorema 3.1 se obtiene que el GFP es un problema de $\mathscr{NP}$-optimización $\mathscr{NP}$-difícil.

3.2 Inaproximabilidad del GFP

En esta sección se prueba la inexistencia de una algoritmo de aproximación para el GFP de factor menor a $\frac{5}{3}$, a menos que $\mathscr{P} = \mathscr{NP}$. Para esto se utiliza el siguiente problema de decisión:

Definición 3.4 *Dado un grafo simple $G = (V,E)$, un conjunto de terminales $K \subseteq V$ y un entero positivo B. El problema de hallar un* ***conjunto separador de*** *k* ***terminales*** *consiste en determinar si existe un conjunto separador $U \subseteq V \setminus K$ tal que $|U| \leq B$ y todo nodo terminal pertenece a una componente conexa distinta en $G \setminus U$.*

Obsérvese que para $k = 2$ este problema pertenece a la clase $\mathscr{P}$, un algoritmo de resolución en tiempo polinomial es el de Ford y Fulkerson. Por su parte, es conocido que para todo $k \geq 3$ el problema pertenece a los de la clase $\mathscr{NP}$-Completos [7].

En el siguiente Teorema se prueba la inaproximabilidad del GFP, reduciendo desde el problema de hallar un conjunto separador de $k = 3$ terminales.

Teorema 3.2 *Aproximar el GFP a menos de $\frac{5}{3} - \varepsilon$ para todo $\varepsilon > 0$ es $\mathscr{NP}$-Difícil.*

Demostración: Considérese una instancia arbitraria (G,K,B) para el problema de hallar un conjunto separador de 3 terminales, con $G \supset K = \{t_1,t_2,t_3\}$. Creamos una instancia corres-pondiente para el GFP mediante una reducción polinomial $(G,K,B) \rightarrow (G^{(K)},B)$, donde $G^{(K)}$ es un grafo idéntico a G en el que los nodos terminales t_1,t_2,t_3 fueron sustituidos por cliques K_1,K_2,K_3 de tamaño $M = cn$ cada una. La constante c se definirá a conveniencia posteriormente en la prueba. Cada nodo de cada clique se conectará a los nodos restantes de G de la misma forma en la que se conectaba el terminal sustituido. Un ejemplo de la reducción se puede observar en la Figura 3.1 de la página siguiente.

La idea intuitiva de esta reducción es que las cliques asociadas a los terminales son las "partes pesadas" del grafo y por lo tanto minimizar la función objetivo del GFP implicará asegurarnos de que terminen en componentes conexas distintas del grafo resultante luego de la inmunización.

Supongamos que existe un algoritmo de aproximación $\mathbf{A}$ para el GFP de factor $\left(\frac{5}{3} - \varepsilon\right)$ para algún $\varepsilon > 0$. Si se aplica el algoritmo $\mathbf{A}$ en la entrada $(G^{(K)},B)$ se obtiene como salida un conjunto de nodos S a inmunizar y un valor para la función objetivo $\mathbf{A}_{G^{(K)}}$. El número

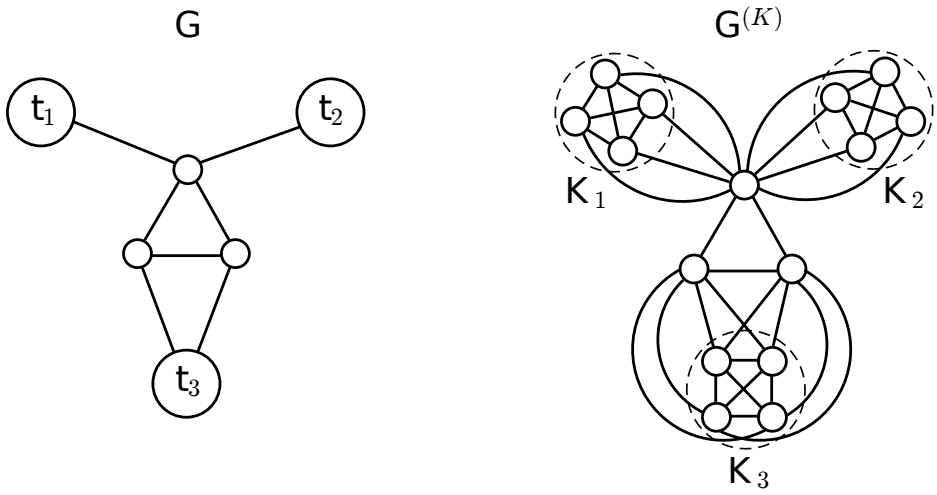

Fig. 3.1 Ejemplo de la reducción propuesta

de nodos resultante en $G^{(K)} \setminus S$ será $\hat{n} = n + 3M - 3 - B$.

El valor objetivo calculado por el algoritmo **A** será como mínimo $\frac{(2M)^2+M^2}{\hat{n}} = \frac{5M^2}{\hat{n}}$ si dos de las cliques pertenecen a la misma componente luego de la inmunización. Por lo tanto, existen dos posibilidades que se plantean en las siguientes afirmaciones:

1. Si $\mathbf{A}_{G^{(K)}} < \frac{5M^2}{\hat{n}}$ la respuesta al problema de hallar un conjunto separador para t_1, t_2, t_3 será SI y dicho certificado positivo puede obtenerse en tiempo polinomial.

2. Si $\mathbf{A}_{G^{(K)}} \geq \frac{5M^2}{\hat{n}}$ la respuesta al problema de hallar un conjunto separador será NO.

Esto implica que es posible resolver el problema de hallar un conjunto separador utilizando el algoritmo **A** y por lo tanto, permite llegar a una contradicción. Se probaran ambas afirmaciones a continuación.

1. Para demostrar esta afirmación se consideran las diferentes maneras en las que podría estar conformado el conjunto S.

   - Si el conjunto S no incluye vértices pertenecientes a las cliques,

     $$S \subseteq V(G^{(K)}) \setminus \{K_1, K_2, K_3\} = V(G) \setminus \{t_1, t_2, t_3\},$$

     entonces S es un conjunto de nodos que separa a los tres terminales t_1, t_2, t_3. Si no las separará existiría una componente en $G^{(K)}$ con al menos dos cliques y en tal caso el valor objetivo correspondiente sería $\mathbf{A}_{G^{(K)}} \geq \frac{(2M)^2+M^2}{\hat{n}} = \frac{5M^2}{\hat{n}}$, lo que contradice la hipótesis de la afirmación.

Si existe intersección no vacía de S con las cliques deberá haber presentes nodos de más de una de ellas, pues elegir nodos de una única clique para eliminar no contribuye a desconectar el grafo resultante.

- Si el conjunto S incluye vértices pertenecientes a las cliques de forma que eliminar S de $G^{(K)}$ nos da una componente C tal que $C \cap K_i \neq \emptyset$ para todo $i \in \{1,2,3\}$. Se tendrá que $|C| \geq 3M - B \geq 3M - n$ y entonces,

$$\mathbf{A}_{G^{(K)}} \geq \frac{|C|^2}{\hat{n}} > \frac{5M^2}{\hat{n}}, \text{si se elige } c \geq 2.$$

  Luego, se contradice la hipótesis de la afirmación y este caso no es posible.

- Si el conjunto S incluye vértices pertenecientes a las cliques de forma que eliminar S de $G^{(K)}$ nos da una componente C que intersecta a dos de ellas y a una no. Sin pérdida de generalidad puede suponerse $C \cap K_i \neq \emptyset$ para $i \in \{1,2\}$ y $C \cap K_3 = \emptyset$. Sean $B_i = |S \cap K_i|$ para $i \in \{1,2,3\}$ y $t = |C \cap V(G)|$, entonces $|C| = 2M - B_1 - B_2 + t$. Existe otra componente que contiene vértices de K_3 y su tamaño será al menos $M - B_3$. Ahora, debe cumplirse que $t \leq B_1 + B_2 + B_3$ pues sino se verifica,

$$\mathbf{A}_{G^{(K)}} \geq \frac{(2M - B_1 - B_2 + t)^2 + (M - B_3)^2}{\hat{n}} > \frac{(2M + B_3)^2 + (M - B_3)^2}{\hat{n}} > \frac{5M^2}{\hat{n}}.$$

  Luego, $S' = S \cup (C \cap V(G)) \setminus \bigcup_{i=1}^{3} K_i$ es un conjunto de nodos que separa a los tres terminales t_1, t_2, t_3 y es un certificado positivo para G.

2. Para demostrar esta afirmación elegiremos el valor de c convenientemente. Como nuestro problema es de minimización y $\mathbf{A}$ es un algoritmo de aproximación de factor $\left(\frac{5}{3} - \varepsilon\right)$ tenemos:

$$\frac{\mathbf{A}_{G^{(K)}}}{GFP(G^{(K)}, B)} \leq \frac{5}{3} - \varepsilon.$$

   Luego, por hipótesis de la afirmación, se cumple:

$$GFP(G^{(K)}, B) \geq \frac{1}{\frac{5}{3} - \varepsilon} \mathbf{A}_{G^{(K)}} \geq \frac{1}{\frac{5}{3} - \varepsilon} \frac{5M^2}{\hat{n}}.$$

   Si se elige c de forma que se verifique:

$$\frac{1}{\frac{5}{3} - \varepsilon} \frac{5M^2}{\hat{n}} > \frac{2M^2 + (M + n - B)^2}{\hat{n}} \tag{3.1}$$

se obtiene que los tres terminales no pueden ser separados y la respuesta al problema de hallar un conjunto separador es NO. El valor $\frac{2M^2+(M+n-B)^2}{\hat{n}}$ es el mayor valor que puede obtenerse teniendo a las tres cliques en diferentes componentes. Un posible valor para c de forma que se verifique la ecuación 3.1 es por ejemplo $c = \frac{55}{\varepsilon}$, lo que puede comprobarse realizando los cálculos.

En conclusión se probó que el algoritmo $\mathbf{A}_{G^{(K)}}$ permite resolver el problema de hallar un conjunto separador para tres terminales y esto contradice el hecho de que el problema pertenece a los de la clase $\mathscr{NP}$-completos. Luego, es $\mathscr{NP}$-difícil aproximar el GFP a un factor $\left(\frac{5}{3} - \varepsilon\right)$ para todo $\varepsilon > 0$. □

3.3 Conclusiones

En las secciones anteriores se mostró que el problema de fragmentación de grafos pertenece a la categoria de problemas de $\mathscr{NP}$-optimización $\mathscr{NP}$-difíciles. Más aún se probó la inaproximabilidad del problema, mostrando que no existe un algoritmo de aproximación para el GFP con factor menor a $\frac{5}{3}$, a menos que $\mathscr{P} = \mathscr{NP}$.
Esto promueve el desarrollo de metaheurísticas que permitan hallar soluciones aproximadas y la búsqueda de algoritmos que resuelvan el problema para casos pequeños o para familias especiales de grafos en forma exacta.

Vale destacar que el Problema de Fragmentación de Grafos es objeto de estudio de otras tesis de posgrado. En la tesis doctoral del Dr. Juan Piccini [19] se desarrolla una metodología GRASP enriquecida con una técnica de post-optimización denominada Path-Relinking para abordar el problema de estudio. La tesis doctoral en curso de la MSc. Ing. Graciela Ferreira propone métodos exactos de tiempo sobrepolinomial para el Problema de Fragmentación de Grafos, mientras que la tesis en curso de la Lic. Nicole Rosenstock apunta a superar el rendimiento del GRASP previamente desarrollado, combinando diversas estructuras de vecindad.

En esta tesis se propone resolver el problema para familias especiales de grafos, explotanto características particulares en cada caso. En el próximo capítulo se trabajará en forma general con grafos acíclicos (caminos, árboles y bosques) y en forma restringida con grafos bipartitos. Para resolver el Problema de Fragmentación de Grafos en el caso de acíclicos se propone un algoritmo basado en programación dinámica, mientras que para el caso de grafos bipartitos se resuelve en forma exacta exigiendo condiciones especiales.

Capítulo 4

Resolución exacta del GFP en casos especiales.

En el capítulo anterior se mostró que no existe un algoritmo de aproximación para el GFP de factor menor a $\frac{5}{3}$, a menos que $\mathscr{P} = \mathscr{NP}$. En contraste, en este Capítulo se mostrará que es posible resolver de forma exacta el GFP para familias especiales de grafos. Todos los resultados presentados en este Capítulo fueron publicados en [2].

4.1 Propiedades del GFP.

Se puede pensar el GFP en términos de la minimización de la norma de un vector cuyas coordenadas son los órdenes de las componente conexas de un grafo, específicamente el Problema de Minimización de la Norma Euclídea Restringida o *CENM*, mencionado en la Sección 2.3. Las siguientes Proposiciones muestran propiedades del GFP, que pueden demostrarse a partir de propiedades elementales del espacio euclídeo.

Proposición 4.1 *El óptimo para el CENM restringido al hiperplano* $\mathscr{F} = \{\vec{n} \in \mathbb{R}^k : \|\vec{n}\|_1 = n\}$ *se alcanza en el vector de coordenadas idénticas* $n_i = \frac{n}{k}$, $\forall i = 1, \ldots, k$.

Demostración: El vector normal del hiperplano $\mathscr{F}$ tiene todas sus coordenadas idénticas. El óptimo para *CENM* es precisamente la proyección ortogonal, cuya dirección está dada por el vector normal. □

Este resultado sugiere un método para hallar el valor óptimo del GFP en casos especiales. Si el número de componentes conexas del subgrafo resultante luego del proceso

de inmunización está fijo, la solución óptima se obtendrá equilibrando el número de nodos pertenecientes a cada componente. Esto puede verse como sigue:

Proposición 4.2 *Sean los vectores de coordenadas enteras positivas* $\vec{n} = (n_1, \ldots, n_k)$ *tales que* $n_1 \geq n_2 \geq \ldots \geq n_k$. *Se cumple una de las siguientes afirmaciones:*

- *Si* $n_1 \geq n_k + 2$, *el vector* $\vec{u} = \vec{n} - \delta_1 + \delta_k$ *tiene menor norma euclídea que* $\vec{n}$, *donde* δ_i *es el i-ésimo vector unitario.*
- *Si no,* $\vec{n}$ *es el mejor vector de coordenadas enteras de* $\mathbb{R}^k$ *para el CENM dentro de aquellos de norma* 1.

Demostración: En el primer caso se tiene que:

$$\|\vec{n}\| - \|\vec{u}\|^2 = n_1^2 - (n_1 - 1)^2 + n_k^2 - (n_k + 1)^2 = 2(n_1 - n_k) - 2 > 0.$$

En otro caso, si $n_1 - n_k \leq 1$ no existe mejora posible entre los vectores de norma 1. □

4.2 Resolución exacta.

4.2.1 Resolución exacta en caminos elementales y ciclos.

La primera familia de grafos en donde se mostrará la resolución exacta del GFP es la de caminos elementales. Sean $P_n = \{v_1, \ldots, v_n\}$ un camino elemental con n nodos y $B < n$ un presupuesto fijo, para resolver el problema de inmunización en P_n se utilizará la Proposición 4.2.

Obsérvese que si $B \geq \lfloor \frac{n}{2} \rfloor$ es posible inmunizar nodos intercalados de P_n de forma de obtener valor óptimo unitario.

En el caso en que $B < \lfloor \frac{n}{2} \rfloor$ se puede asumir, sin pérdida de generalidad, que la solución óptima no elige para inmunizar nodos adyacentes ni nodos que se encuentren en los extremos del camino. Pues si se considera una solución que inmuniza nodos adyacentes v_i, v_{i+1} y no a uno de los siguientes k nodos que están en el camino se puede observar que el costo total de la solución aumenta. Supongamos que se elige para inmunizar v_{i+2} en lugar de v_{i+1}, como en la Figura 4.1, entonces el costo total disminuye en $k^2 - (k-1)^2 - 1 \geq 0$.

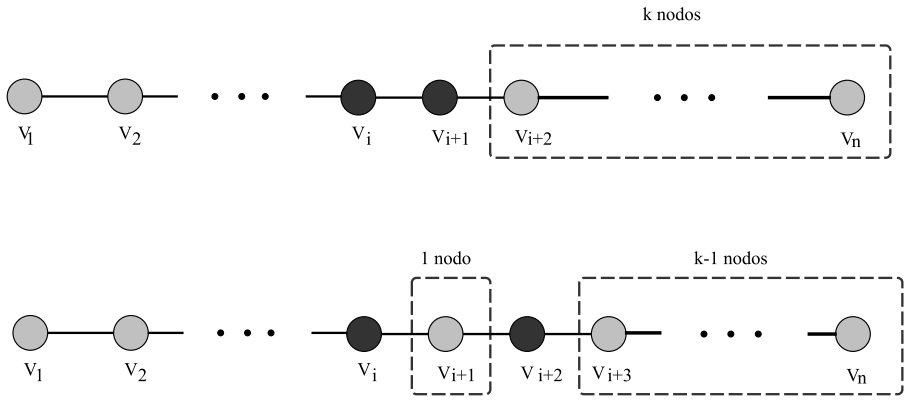

Fig. 4.1 Distintas elecciones de Inmunización.

Luego, en la estrategia óptima de inmunización se eliminan B nodos y se generan $B+1$ componentes conexas. Tales componentes deben tener un número de nodos equilibrado. Con esto se quiere decir que la cantidad de nodos de una componente respecto a otra debe diferir como máximo en uno, lo que se logra de la siguiente manera:

Proposición 4.3 *Sean los enteros d y r definidos por la regla de división euclídea, es decir, tales que* $0 \leq r < B+1$ *y* $n-B = d(B+1)+r$. *El costo mínimo para el GFP se alcanza en el grafo* G' *que tiene r caminos elementales* P_{d+1} *y* $(B+1-r)$ *caminos elementales* P_d.

Demostración: La diferencia de tamaño entre dos componentes de G' es como máximo 1. Luego, el vector de tamaños $\overrightarrow{n}_{G'}$ cumple con la Proposición 4.2. □

Como Corolario se obtiene que es posible resolver de forma exacta el GFP también para la familia de Ciclos.

Corolario 4.1 *La solución óptima del GPF se puede hallar en tiempo polinomial si el grafo de entrada es un ciclo.*

Demostración: La estrategia óptima de inmunización en un ciclo es comenzar inmunizando un nodo arbitrario. Luego de eliminarlo se obtiene un camino elemental y se procede como en la Proposición 4.3. □

4.2.2 Resolución exacta en árboles y bosques.

En esta sección se mostrará que la solución óptima del GFP puede hallarse en tiempo polinomial si el grafo de entrada es un árbol. En el caso de la topología de árbol se verá

que es posible resolver en forma exacta el GFP mediante la técnica de Programación Dinámica [15]. A continuación se introducen la terminología y las definiciones necesarias para definir la recurrencia correspondiente con dicha técnica.

Dado un árbol T, se elige un nodo como raíz r y se ordenan sus nodos en niveles respecto de su distancia a r. Decimos que u es el padre de v, si u se encuentra un nivel por encima de v y decimos que v es descendiente de u.
Dado un grafo conexo $T' \subseteq T$ decimos que T' es una rama de T y la raíz de T' es su nodo de nivel más alto. Si una rama contiene a todos los descendientes de su raíz v diremos que es una rama completa y la anotaremos T_v.

La idea central para resolver el problema consiste en resolver en ramas completas a bajo nivel para luego resolver recursivamente sobre la unión de dichas ramas hasta cubrir todo el árbol. Para cada rama del árbol será necesario conocer la solución óptima como una función F de dos parámetros:

- b, el número de vértices de la rama que serán eliminados, y
- c, el tamaño de la componente conexa que contiene a la raíz de la rama luego de eliminar los b vértices elegidos antes.

Sea una rama T' con raíz r. El algoritmo elegirá distintos conjuntos $S \subseteq V(T')$ de b nodos para eliminar. Esto definirá subgrafos $T' \setminus S$ que tendrán en cada caso un cierto número de componentes conexas. Si la raíz r no fue eliminada existirá en $T' \setminus S$ una componente conexa que contiene a r y la anotaremos $C_{(r)}$. Anotaremos $C_1, \ldots, C_k$ para las restantes (supongamos k) componentes conexas que no contienen a la raíz. La función F se define formalmente como sigue.

Definición 4.1 *Con la notación anterior, $F(b,c)$ es* ***el menor valor de*** $\sum_{i=1}^{k} \mid C_i \mid^2$ *con* $\mid S \mid = b$ *y* $\mid C_{(r)} \mid = c$.

Obsérvese que para algunas combinaciones de valores (b,c) puede no ser posible satisfacer el requisito $\mid C_{(r)} \mid = c$, para esos valores se impone $F(b,c) = \infty$. También cabe destacar que en el caso $c = 0$, pedir $\mid C_{(r)} \mid = 0$ es equivalente a imponer que la raíz r sea eliminada del grafo.

Es importante notar que por definición de F el valor óptimo del GFP en la entrada (T,B) es igual a $\frac{1}{n}.\min_c\{F(B,c)+c^2\}$, donde $n = \mid V(T) \mid -B$ y $c \in \{0,\ldots,n\}$. A continuación se

presenta un ejemplo del cálculo de F para un árbol de 5 nodos del cual se desean eliminar $b = 2$.

Ejemplo 4.1 *Consideremos el grafo T representado en la Figura 4.2.*

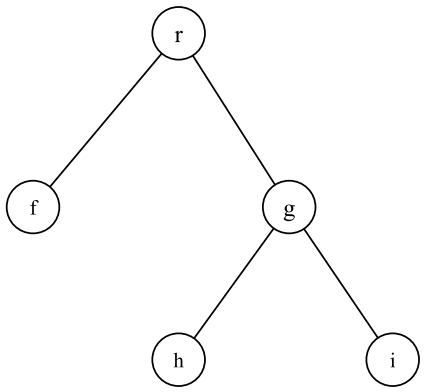

Fig. 4.2 Árbol de 5 nodos.

- *Primero se calcula* $F(2,0)$. *Esto es, se eliminarán* 2 *nodos y la componente conexa que contiene a la raíz r deberá tener tamaño* 0 *(la raíz será eliminada). Existen varias posibilidades:*
  - *Si se elige eliminar* $S = \{r, f\}$, *se tendrá que* $T \setminus S$ *es un grafo conexo (C) de* 3 *nodos y por lo tanto el valor de* $| C |^2 = (3)^2 = 9$.
  - *Si se elige eliminar* $S = \{r, i\}$ *o* $S = \{r, h\}$, *se tendrá que* $T \setminus S$ *es un grafo con dos componentes conexas* (C_1, C_2) *de* 1 *y* 2 *nodos respectivamente y por lo tanto, el valor de* $| C_1 |^2 + | C_2 |^2 = (1)^2 + (2)^2 = 5$.
  - *Si se elige eliminar* $S = \{r, g\}$, *se tendrá que* $T \setminus S$ *es un grafo con tres componentes conexas* (C_1, C_2, C_3) *todas de* 1 *nodo y por lo tanto el valor de* $| C_1 |^2 + | C_2 |^2 + | C_3 |^2 = (1)^2 + (1)^2 + (1)^2 = 3$.

  Luego, el valor de $F(2,0) = \min\{3,5,9\} = 3$ *y el conjunto S correspondiente es* $S = \{r, g\}$.

- *Ahora se calcula* $F(2,1)$. *Esto es, se eliminarán dos nodos y la componente conexa que contiene a la raíz r deberá tener tamaño* 1. *Se puede observar que existe una*

única posibilidad para hacerlo, que es eliminar $S = \{f, g\}$. Esto define un grafo $T \setminus S$ con una componente conexa que contiene a la raíz $C_{(r)}$ de tamanõ 1 y otras dos componentes conexas (C_1,C_2) también de tamaño 1. Por lo tanto, el valor de $\mid C_1 \mid^2 + \mid C_2 \mid^2 = (1)^2 + (1)^2 = 2$. Es decir, $F(2,1) = 2$ y el conjunto S correspondiente es $S = \{f, g\}$.

- *Luego se calcula $F(2,2)$, es decir, se eliminarán 2 nodos y se desea obtener una componente conexa que contenga al vértice de tamanõ 2. Para esto, es posible eliminar el conjunto $S = \{g, h\}$ o $S = \{g, i\}$. En ambos casos se define un grafo $T \setminus S$ con una componente conexa que contiene a la raíz $C_{(r)}$ de tamanõ 2 y otra componente conexa C_1 de tamaño 1. Por lo tanto, el valor de $\mid C_1 \mid^2 = (1)^2 = 1$. Dado que ambos conjuntos S devuelven el mismo valor este será también el valor de $F(2,2)$. Entonces $F(2,2) = 1$ y el conjunto S correspondiente puede ser tanto $S = \{g, h\}$ como $S = \{g, i\}$.*

- *Por último se calcula $F(2,3)$, es decir se eliminarán 2 nodos y la componente conexa que contiene a la raíz r deberá tener tamaño 3. Para esto es posible eliminar uno de los siguiente conjuntos $S = \{h, i\}$, $S = \{f, h\}$ o $S = \{f, i\}$. En todos los casos se define un grafo $T \setminus S$ con una única componente conexa que contiene a la raíz $C_{(r)}$ de tamaño 3. Luego, $F(2,3) = 0$ y el conjunto S correspondiente puede ser $S = \{h, i\}$, $S = \{f, h\}$ o $S = \{f, i\}$.*

A continuación se demuestran las proposiciones necesarias para definir la recurrencia que permitirá calcular F en un árbol partiendo del valor de F ya calculado en ramas completas del mismo.

Proposición 4.4 *Sean u y v dos nodos tales que u es padre de v. Si F ya está calculada para una rama completa T_v, entonces se puede calcular F para T_u en tiempo polinomial.*

Demostración: Sea T_v una rama completa y u el padre de v como en la Figura 4.3

Anotamos F_u y F_v a la función F calculada en T_u y T_v respectivamente. Mostraremos que F_u puede calcularse a partir de F_v de la siguiente forma:

$$F_u(b,c) = \begin{cases} F_v(b, c-1), \text{ si } c > 0 \\ \min\limits_{c'} \{F_v(b-1, c') + (c')^2\}, \text{ si } c = 0 \end{cases}$$

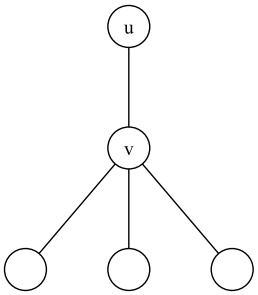

Fig. 4.3 Árbol con nodo padre u y nodo hijo v.

donde $c' \in \{0,1,\ldots,\mid V(T_v) \mid\}$. Dividimos la prueba en diferentes casos:

- Si $c > 0$ y $F_v(b,c-1) = M < \infty$.
  Claramente, eliminar de T_u el mismo conjunto de vértices que dió el valor M en T_v dará el mismo valor, luego $F_u(b,c) \leq M$. A su vez, si existiese un conjunto de vértices S que permitiera obtener un valor estrictamente menor que M, como $c > 0$, u no estaría en S y eso haría que se cumpliera $F_v(b,c-1) < M$, lo que es un absurdo.

- Si $c > 0$ y $F_v(b,c-1) = \infty$.
  En este caso se debe cumplir $F_v(b,c-1) = \infty$. De hecho, si eliminando b vértices de T_u se pudiera obtener $\mid C_{(u)} \mid = c > 0$, la misma elección de vértices para eliminar en T_v debería generar una componente conexa $C_{(v)}$ de tamaño $c-1$, lo que nos llevaría a un absurdo.

- Si $c = 0$. Obsérvese que $F_u(b,c) < \infty$ a menos que $b = 0$. Sea S el conjunto que da el valor óptimo para $F_u(b,c)$, como $c = 0$ tenemos que $u \in S$. Si c' es el tamaño de la componente conexa de $T_u \setminus S$ que contiene a v tenemos que se verifica $F_v(b,c) = F_v(b-1,c') + (c')^2$. Tomar el mínimo sobre todos los posibles valores de c' nos permite determinar el valor correspondiente que nos dará el óptimo además de su correspondiente conjunto S'. Luego, tomamos $S = S' \cup \{u\}$ y esto hace que se cumpla la igualdad.

□

Proposición 4.5 *Sean dos ramas T_1 y T_2 con el mismo nodo raíz. Si ya se calculó F para T_1 y para T_2, se puede calcular F para $T_1 \cup T_2$ en tiempo polinomial.*

Demostración: Sean T_1 y T_2 dos ramas con el mismo nodo raíz v y supongamos que tenemos calculada F para T_1 y T_2, lo que anotamos F_1 y F_2 respectivamente. Sea $T_3 = T_1 \cup T_2$

y F_3 la función F calculada en T_3. Mostraremos que F_3 puede calcularse de la siguiente forma:

$$F_3(b,c) = \begin{cases} \min\limits_{b_1,c_1}\{F_1(b_1,c_1) + F_2(b-b_1, c-c_1+1)\}, \text{ si } c > 0 \\ \min\limits_{b_1}\{F_1(b_1,0) + F_2(b-b_1+1, 0)\}, \text{ si } c = 0 \end{cases}$$

donde $b_1 \in \{0,\ldots,b\}$ y $c_1 \in \{1,\ldots,c\}$.
Dividimos la prueba en diferentes casos:

- Si $c > 0$ y $F_3(b,c) = \infty$.
  Es necesario probar que para toda elección b_1, c_1 tendremos alguno de los valores $F_1(b_1,c_1)$ o $F_2(b-b_1, c-c_1+1)$ igual a infinito. Supongamos, por absurdo, que ninguno vale infinito, entonces deben existir conjuntos $S_1 \subseteq T_1$ y $S_2 \subseteq T_2$ tales que $|S_i| = b_i$ para $i \in \{1,2\}$ y que determinan componentes conexas que contienen a v que verifican $|C^1_{(v)}| = c_1$ y $|C^2_{(v)}| = c - c_1 + 1$ respectivamente. Luego, como $c > 0$, se tiene que $v \notin S_i$ para $i \in \{1,2\}$, es decir que S_1 y S_2 son disjuntos y por lo tanto $|S_1 \cup S_2| = b$. Esto nos lleva a una contradicción, pues T_3 debería verificar $|C_{(v)}| = |C^1_{(v)}| + |C^2_{(v)}| - 1 = c_1 + c - c_1 + 1 - 1 = c$, lo que implicaría que el valor de $F_3(b,c)$ no es infinito.

- Si $c > 0$ y $F_3(b,c) = M < \infty$.
  Sea $S \subseteq V(T_3)$ un conjunto que nos dé el valor M para $F_3(b,c)$. Entonces, los conjuntos $S \cap V(T_i)$ con $i \in \{1,2\}$ darán valor M_i con $i \in \{1,2\}$ para sus respectivas F_i, de forma que se verifica $M_1 + M_2 = M$. Estableciendo $b_1 = |S \cap V(T_1)|$ y $c_1 = |C_{(v)} \cap T_1|$ se cumple que $F_3(b,c) = \min\limits_{b_1,c_1}\{F_1(b_1,c_1) + F_2(b-b_1, c-c_1+1)\}$.

- Si $c = 0$, obsérvese que $F_3(b,c)$ será finito siempre que $b \neq 0$. En tal caso existirá M tal que $F_3(b,c) = M$. Luego, similarmente al caso anterior podemos concluir que se cumple la igualdad.

□

Las relaciones de recurrencia definidas en las Proposiciones 4.4 y 4.5 permiten definir un algoritmo para el calculo del óptimo del GFP en un grafo con topología de árbol.

Para un árbol T y un presupuesto B el algoritmo iniciará considerando ramas $T_1,\ldots,T_m$ a bajo nivel de forma que sean disjuntas dos a dos o sólo compartan la raíz. Las raíces de estas ramas se tomarán todas al mismo nivel y cada hoja de T deberá estar exactamente en una única rama. Para cada T_i el algoritmo calculará el valor de $F(b,c)$ para $b = 0,\ldots,B$ y

$c = 0, \ldots, |V(T_i)|$ y asociará a cada valor finito obtenido un conjunto S de vértices a eliminar. Se almacenarán tanto el valor de $F(b,c)$ como el conjunto S correspondiente. Luego, recursivamente, se calculará F para ramas de tamaño cada vez mayor. Para esto se agregarán raíces o se unirán ramas hasta terminar cubriendo todo el árbol T.

Formalmente, en una cierta etapa del algoritmo se tiene una lista de ramas para la que F ya fue calculado. Para cada rama T' se hace lo que sigue, si T' comparte su raíz con otra rama T'' se calcula F para $T' \cup T''$, se borran de la lista las ramas T' y T'' y se agrega $T' \cup T''$; si T' no comparte su raíz con ninguna otra rama de la lista, se calcula F para $T' \cup \{r'\}$ donde r' es el padre de la raíz de T' y se reemplaza en la lista T' por $T' \cup \{r'\}$. Nótese que el segundo caso sólo puede suceder cuando T' es una rama completa. De esta manera nos aseguramos de que el proceso nos lleve a una lista con una sola rama final, T. Es importante observar que para una rama cualquiera T', siempre existiran valores de b, c para los cuales $F(b,c)$ será finito, lo que implica que por lo menos existirá un conjunto guardado para T' y podrá ser utilizado en las iteraciones siguientes.

Cada iteración del cálculo de F toma un tiempo del orden $\mathbf{O}(B.|V(T_i)|)$ para la rama T_i que esté siendo considerada en ese momento y se observa que la cantidad de evaluaciones de la función F es lineal en el número de nodos del árbol T. Por lo tanto, el algoritmo completará el cálculo de F en el árbol T en tiempo polinomial. También, se obtendrá el conjunto de vértices a ser eliminados de manera eficiente a partir de los conjuntos guardados en iteraciones previas. Con todos los valores calculados de F para T la solución óptima para el GFP se obtiene hallando:

$$OPT = \frac{1}{n}.\min_{c}\{F(B,c) + c^2\}, \tag{4.1}$$

donde $n = |V(T)| - B$ y $c \in \{0, \ldots, n\}$.

Para completar esta sección se incluye un ejemplo del cálculo del valor óptimo del GFP, mediante la aplicación del algoritmo definido previamente, para un árbol de 8 nodos del cual se desean eliminar $B = 2$.

Ejemplo 4.2 *Se quiere calcular el valor óptimo del GFP para el árbol de la Figura 4.4 con un presupuesto $B = 2$.*

Se consideran las ramas T_1 y T_2 de T como puede apreciarse en la Figura.

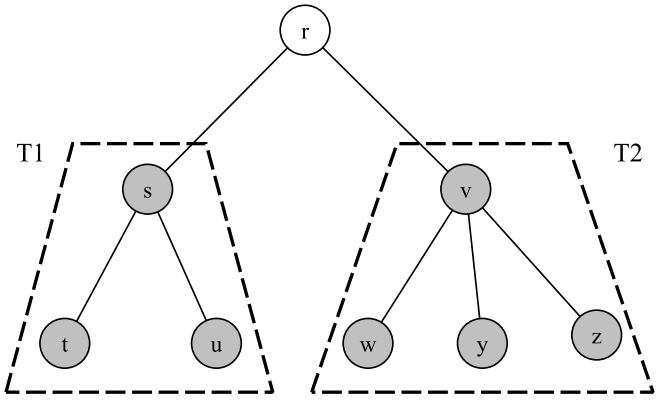

Fig. 4.4 Algoritmo de resolución del GFP en árboles.

- *Inicialmente se calcula F para T_1, anotamos F_1. Los valores de $F_1(b,c)$ para cada $b \in \{0,1,2\}$ y sus respectivos conjuntos S se presentan en la siguiente tabla:*

b	c	$F_1(b,c)$	*Conjunto* S_1
0	*3*	*0*	$\emptyset$
1	*0*	*2*	$\{s\}$
1	*1*	∞	$\nexists$
1	*2*	*0*	$\{t\}$ *o* $\{u\}$
2	*0*	*1*	$\{s,u\}$ *o* $\{s,t\}$
2	*1*	*0*	$\{t,u\}$

Nótese que las combinaciones de b,c que no se presentan en la tabla corresponden con $F(b,c) = \infty$.

- *Se calcula F para T_2, anotamos F_2. Los valores de $F_2(b,c)$ para $b \in \{0,1,2\}$ y sus respectivos conjuntos S se presentan en la siguiente tabla:*

b	c	$F_2(b,c)$	*Conjunto* S_2
0	*4*	*0*	$\emptyset$
1	*0*	*3*	$\{v\}$
1	*3*	*0*	$\{w\}$ *o* $\{y\}$ *o* $\{z\}$
2	*0*	*2*	$\{v,w\}$ *o* $\{v,y\}$ *o* $\{v,z\}$
2	*2*	*0*	$\{w,y\}$ *o* $\{w,z\}$ *o* $\{y,z\}$

Nuevamente las combinaciones de b,c que no se presentan en la tabla corresponden con $F(b,c) = \infty$.

Luego, se agrega la raíz r a T_1 y T_2 para formar los grafos T_3 y T_4 respectivamente, como se puede observar en la Figura 4.5.

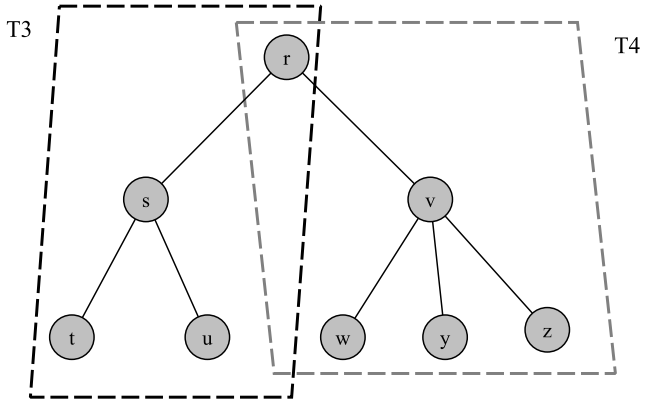

Fig. 4.5 Algoritmo de resolución del GFP en árboles.

- *A continuación, utilizando la Proposición 4.4, se calcula F para T_3 y anotamos F_3. Los valores de $F_3(b,c)$ para cada $b \in \{0,1,2\}$ y sus respectivos conjuntos S se presentan en la siguiente tabla:*

b	c	$F_3(b,c)$	*Conjunto* S_3
0	*4*	*0*	$\emptyset$
1	*0*	*9*	$\{r\}$
1	*1*	*2*	$\{s\}$
1	*2*	∞	$\nexists$
1	*3*	*0*	$\{t\}$ *o* $\{u\}$
2	*0*	*2*	$\{r,s\}$
2	*1*	*1*	$\{s,u\}$ *o* $\{s,t\}$
2	*2*	*0*	$\{t,u\}$

Nótese que el valor de F_3 para todas las combinaciones (b,c) que no aparecen en la tabla toman el valor ∞.

- *Nuevamente, utilizando la Proposición 4.4, se calcula F para T_4 y anotamos F_4. Los valores de $F_4(b,c)$ para cada $b \in \{0,1,2\}$ y sus respectivos conjuntos S se presentan*

en la siguiente tabla:

b	c	$F_4(b,c)$	*Conjunto* S_4
0	5	0	$\emptyset$
1	0	16	$\{r\}$
1	1	3	$\{v\}$
1	2	∞	$\nexists$
1	3	∞	$\nexists$
1	4	0	$\{w\}$ *o* $\{y\}$ *o* $\{z\}$
2	0	3	$\{r,v\}$
2	1	2	$\{v,w\}$ *o* $\{v,y\}$ *o* $\{v,z\}$
2	2	∞	$\nexists$
2	3	0	$\{w,y\}$ *o* $\{w,z\}$ *o* $\{y,z\}$
2	4	∞	$\nexists$

Nótese que el valor de F_4 *para todas las combinaciones* (b,c) *que no aparecen en la tabla toman el valor* ∞*.*

Por último, dado que los grafos T_3 *y* T_4 *comparten la raíz, utilizando la Proposición 4.5, se puede calcular* F *en el grafo* $T = T_3 \cup T_4$ *como en la Figura 4.6*

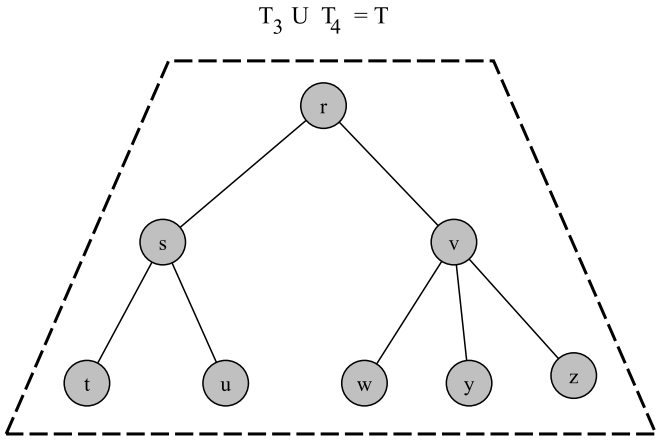

Fig. 4.6 Algoritmo de resolución del GFP en árboles.

Como se desea encontrar el valor óptimo del GFP cuando el presupuesto es $B = 2$ *es necesario calcular* $F(2,c)$ *para* $c \in \{0,1,2,3,4,5,6\}$*. Esto se muestra en las siguientes tablas:*

- *Para $F(2,0)$, por Proposición 4.5, se tiene que:*

$$F(2,0) = \min_{b_1 \in \{0,1,2\}} \{F_3(b_1,0) + F_4(3-b_1,0)\}.$$

Los valores correspondientes de F_3 y F_4 se muestran en la siguiente tabla:

b_1	$F_3(b_1,0)$	*Conjunto* S_3	$3-b_1$	$F_4(3-b_1,0)$	*Conjunto* S_4
0	∞	$\nexists$	*3*	*1*	$\{v,w,y\}$
1	*9*	$\{r\}$	*2*	*3*	$\{r,v\}$
2	*2*	$\{s,r\}$	*1*	*16*	$\{r\}$

Luego, $F(2,0) = \min\{9+3, 2+16\} = 12$ y el conjunto $S = \{r,v\}$.

- *Para $F(2,1)$, por Proposición 4.5, se tiene que:*

$$F(2,1) = \min_{b_1 \in \{0,1,2\}} \{F_3(b_1,1) + F_4(2-b_1,1)\}.$$

Los valores correspondientes de F_3 y F_4 se muestran en la siguiente tabla:

b_1	$F_3(b_1,1)$	*Conjunto* S_3	$2-b_1$	$F_4(2-b_1,1)$	*Conjunto* S_4
0	∞	$\nexists$	*2*	*2*	$\{v,w\}$ *o* $\{v,y\}$ *o* $\{v,z\}$
1	*2*	$\{s\}$	*1*	*3*	$\{v\}$
2	*1*	$\{s,u\}$ *o* $\{s,t\}$	*0*	∞	$\nexists$

Luego, $F(2,1) = 2+3 = 5$ y el conjunto $S = \{s,v\}$.

- *Para $F(2,2)$, por Proposición 4.5, se tiene que:*

$$F(2,2) = \min_{b_1,c_1} \{F_3(b_1,c_1) + F_4(2-b_1,3-c_1)\},$$

donde $b_1 \in \{0,1,2\}$ y $c_1 \in \{1,2\}$. Los valores correspondientes de F_3 y F_4 se muestran en la siguiente tabla:

b_1	c_1	$F_3(b_1,c_1)$	*Conjunto* S_3	$2-b_1$	$3-c_1$	$F_4(2-b_1,3-c_1)$	*Conjunto* S_4
0	*1*	∞	$\nexists$	*2*	*2*	∞	$\nexists$
0	*2*	∞	$\nexists$	*2*	*1*	*2*	$\{v,w\}$ *o* $\{v,y\}$ *o* $\{v,z\}$
1	*1*	*2*	$\{s\}$	*1*	*2*	∞	$\nexists$
1	*2*	∞	$\nexists$	*1*	*1*	*3*	$\{v\}$
2	*1*	*1*	$\{s,u\}$ *o* $\{s,t\}$	*0*	*2*	∞	$\nexists$
2	*2*	*0*	$\{t,u\}$	*0*	*1*	∞	$\nexists$

Luego, $F(2,2) = \infty$.

- *Para* $F(2,3)$*, por Proposición 4.5, se tiene que:*

$$F(2,3) = \min_{b_1,c_1}\{F_3(b_1,c_1) + F_4(2-b_1,4-c_1)\},$$

donde $b_1 \in \{0,1,2\}$ *y* $c_1 \in \{1,2,3\}$. *Los valores correspondientes de* F_3 *y* F_4 *se muestran en la siguiente tabla:*

b_1	c_1	$F_3(b_1,c_1)$	*Conjunto* S_3	$2-b_1$	$4-c_1$	$F_4(2-b_1,4-c_1)$	*Conjunto* S_4
0	*1*	∞	$\nexists$	*2*	*3*	*0*	$\{\}$
0	*2*	∞	$\nexists$	*2*	*2*	∞	$\nexists$
0	*3*	∞	$\nexists$	*2*	*1*	*2*	$\{v,w\}$ *o* $\{v,y\}$ *o* $\{v,z\}$
1	*1*	*2*	$\{s\}$	*1*	*3*	∞	$\nexists$
1	*2*	∞	$\nexists$	*1*	*2*	∞	$\nexists$
1	*3*	*0*	$\{t\}$ *o* $\{u\}$	*1*	*1*	*3*	$\{v\}$
2	*1*	*1*	$\{s,u\}$ *o* $\{s,t\}$	*0*	*3*	∞	$\nexists$
2	*2*	*0*	$\{t,u\}$	*0*	*2*	∞	$\nexists$
2	*3*	∞	$\nexists$	*0*	*1*	∞	$\nexists$

Luego, $F(2,3) = 0+3 = 3$ *y el conjunto* $S = \{t,v\}$ *o* $S = \{u,v\}$.

- *Para* $F(2,4)$*, por Proposición 4.5, se tiene que:*

$$F(2,4) = \min_{b_1,c_1}\{F_3(b_1,c_1) + F_4(2-b_1,5-c_1)\},$$

donde $b_1 \in \{0,1,2\}$ *y* $c_1 \in \{1,2,3,4\}$. *Los valores correspondientes de* F_3 *y* F_4 *se muestran en la siguiente tabla:*

b_1	c_1	$F_3(b_1,c_1)$	*Conjunto* S_3	$2-b_1$	$5-c_1$	$F_4(2-b_1,5-c_1)$	*Conjunto* S_4
0	*1*	∞	$\nexists$	*2*	*4*	∞	$\nexists$
0	*2*	∞	$\nexists$	*2*	*3*	*0*	$\{w,y\}$, $\{w,z\}$ *o* $\{y,z\}$
0	*3*	∞	$\nexists$	*2*	*2*	∞	$\nexists$
0	*4*	*0*	$\emptyset$	*2*	*1*	*2*	$\{v,w\}$, $\{v,y\}$ *o* $\{v,z\}$
1	*1*	*2*	$\{s\}$	*1*	*4*	*0*	$\{w\}$, $\{y\}$ *o* $\{z\}$
1	*2*	∞	$\nexists$	*1*	*3*	∞	$\nexists$
1	*3*	*0*	$\{t\}$ *o* $\{u\}$	*1*	*2*	∞	$\nexists$
1	*4*	∞	$\nexists$	*1*	*1*	*3*	$\{v\}$
2	*1*	*1*	$\{s,u\}$ *o* $\{s,t\}$	*0*	*4*	∞	$\nexists$
2	*2*	*0*	$\{t,u\}$	*0*	*3*	∞	$\nexists$
2	*3*	∞	$\nexists$	*0*	*2*	∞	$\nexists$
2	*4*	∞	$\nexists$	*0*	*1*	∞	$\nexists$

Luego, $F(2,4) = \min\{0+2, 2+0\} = 2$ *y el conjunto S puede ser uno de los que siguen:* $\{v,w\}$, $\{v,y\}$, $\{v,z\}$, $\{s,w\}$, $\{s,y\}$ *o* $\{s,z\}$.

- *Para* $F(2,5)$*, por Proposición 4.5, se tiene que:*

$$F(2,5) = \min_{b_1,c_1}\{F_3(b_1,c_1) + F_4(2-b_1, 6-c_1)\},$$

donde $b_1 \in \{0,1,2\}$ *y* $c_1 \in \{1,2,3,4,5\}$. *Los valores correspondientes de* F_3 *y* F_4 *se muestran en la siguiente tabla:*

b_1	c_1	$F_3(b_1,c_1)$	*Conjunto* S_3	$2-b_1$	$6-c_1$	$F_4(2-b_1,6-c_1)$	*Conjunto* S_4
0	*1*	∞	$\nexists$	*2*	*5*	∞	$\nexists$
0	*2*	∞	$\nexists$	*2*	*4*	∞	$\nexists$
0	*3*	∞	$\nexists$	*2*	*3*	*0*	$\{w,y\}$, $\{w,z\}$ *o* $\{y,z\}$
0	*4*	*0*	$\emptyset$	*2*	*2*	∞	$\nexists$
0	*5*	∞	$\nexists$	*2*	*1*	*2*	$\{v,w\}$, $\{v,y\}$ *o* $\{v,z\}$
1	*1*	*2*	$\{s\}$	*1*	*5*	∞	$\nexists$
1	*2*	∞	$\nexists$	*1*	*4*	*0*	$\{w\}$, $\{y\}$ *o* $\{z\}$
1	*3*	*0*	$\{t\}$ *o* $\{u\}$	*1*	*3*	∞	$\nexists$
1	*4*	∞	$\nexists$	*1*	*2*	∞	$\nexists$
1	*5*	∞	$\nexists$	*1*	*1*	*3*	$\{v\}$
2	*1*	*1*	$\{s,u\}$ *o* $\{s,t\}$	*0*	*5*	*0*	$\emptyset$
2	*2*	*0*	$\{t,u\}$	*0*	*4*	∞	$\nexists$
2	*3*	∞	$\nexists$	*0*	*3*	∞	$\nexists$
2	*4*	∞	$\nexists$	*0*	*2*	∞	$\nexists$
2	*5*	*0*	$\emptyset$	*0*	*1*	∞	$\nexists$

Luego, $F(2,5) = 1+0 = 1$ *y el conjunto S puede ser* $\{s,u\}$ *o* $\{s,t\}$.

- *Por último, para* $F(2,6)$*, por Proposición 4.5, se tiene que:*

$$F(2,6) = \min_{b_1,c_1}\{F_3(b_1,c_1) + F_4(2-b_1,7-c_1)\},$$

donde $b_1 \in \{0,1,2\}$ *y* $c_1 \in \{1,2,3,4,5,6\}$. *Los valores correspondientes de* F_3 *y* F_4 *se muestran en la siguiente tabla:*

b_1	c_1	$F_3(b_1,c_1)$	*Conjunto* S_3	$2-b_1$	$7-c_1$	$F_4(2-b_1,7-c_1)$	*Conjunto* S_4
0	*1*	∞	$\nexists$	*2*	*6*	∞	$\nexists$
0	*2*	∞	$\nexists$	*2*	*5*	∞	$\nexists$
0	*3*	∞	$\nexists$	*2*	*4*	∞	$\nexists$
0	*4*	*0*	$\emptyset$	*2*	*3*	*0*	$\{w,y\}$, $\{w,z\}$ *o* $\{y,z\}$
0	*5*	∞	$\nexists$	*2*	*2*	∞	$\nexists$
0	*6*	∞	$\nexists$	*2*	*1*	*2*	$\{v,w\}$, $\{v,y\}$ *o* $\{v,z\}$
1	*1*	*2*	$\{s\}$	*1*	*6*	∞	$\nexists$
1	*2*	∞	$\nexists$	*1*	*5*	∞	$\nexists$
1	*3*	*0*	$\{t\}$ *o* $\{u\}$	*1*	*4*	*0*	$\{w\}$, $\{y\}$ *o* $\{z\}$
1	*4*	∞	$\nexists$	*1*	*3*	∞	$\nexists$
1	*5*	∞	$\nexists$	*1*	*2*	∞	$\nexists$
1	*6*	∞	$\nexists$	*1*	*1*	*3*	$\{v\}$
2	*1*	*1*	$\{s,u\}$ *o* $\{s,t\}$	*0*	*6*	∞	$\nexists$
2	*2*	*0*	$\{t,u\}$	*0*	*5*	*0*	$\emptyset$
2	*3*	∞	$\nexists$	*0*	*4*	∞	$\nexists$
2	*4*	∞	$\nexists$	*0*	*3*	∞	$\nexists$
2	*5*	∞	$\nexists$	*0*	*2*	∞	$\nexists$
2	*6*	∞	$\nexists$	*0*	*1*	∞	$\nexists$

Luego, $F(2,6)=0$ *y el conjunto S puede ser uno de los que siguen:* $\{w,y\}$, $\{w,z\}$, $\{y,z\}$, $\{t,w\}$, $\{t,y\}$, $\{t,z\}$, $\{u,w\}$, $\{u,y\}$, $\{u,z\}$ *o* $\{t,u\}$.

- *Para finalizar el cálculo del óptimo del GFP se utiliza la Ecuación 4.1 como sigue:*

$$OPT = \frac{1}{6}.\min_{c}\{F(2,c)+c^2\}$$

Esto es,

$$OPT = \frac{1}{6}.\min\{12+(0)^2, 5+(1)^2, \infty+(2)^2, 3+(3)^2, 2+(4)^2, 1+(5)^2, 0+(6)^2\}$$

Entonces,

$$\boldsymbol{OPT} = \frac{1}{6}.(5+(1)^2) = \boxed{1},$$

lo que se corresponde con elegir el conjunto ***S={s,v}*** *para ser eliminado.*

Como Corolario del procedimiento descrito previamente para árboles es posible ver que la resolución exacta en tiempo polinomial se extiende también a bosques.

Si se tiene un grafo acíclico arbitrario (un bosque), cada componente del grafo será un árbol. Supongamos que anotamos a las componentes $T_1, T_2, \ldots, T_k$ y B es el presupuesto fijo. La estrategia óptima para $T' = T_1 \cup T_2$ puede obtenerse utilizando el algoritmo anterior y todas las particiones naturales de B tales que $B = B_1 + B_2$. Repitiendo para la unión $T' \cup T_3$ y así sucesivamente hasta completar $T_1 \cup \ldots \cup T_k$ se resuelve el problema. Alternativamente podríamos resolverlo como se propone en el siguiente Corolario:

Corolario 4.2 *La solución óptima del GFP puede hallarse en tiempo polinomial para todos los grafos acíclicos.*

Demostración: Se conectan todas las raíces de los árboles existentes a un nuevo nodo y se aplica el algoritmo definido antes con valor $F(B+1,0)$. □

4.2.3 Resolución exacta en grafos bipartitos.

En esta sección se analiza la estrategia de inmunización para la familia de grafos bipartitos. Se probará el siguiente resultado:

Proposición 4.6 *El óptimo del GFP puede hallarse en tiempo polinomial si el grafo de entrada es bipartito y el presupuesto B es mayor o igual al tamaño del emparejamiento máximo.*

Demostración: Considérese un grafo bipartito $G = (A,E)$ donde $A = A_1 \cup A_2$ y $E \subseteq A_1 \times A_2$.

Recuérdese que para grafos bipartitos el Teorema de König (Teorema 1.1) establece que el cardinal mínimo de un cubrimiento de vértices es igual al tamaño de un emparejamiento máximo. Este tamaño puede hallarse utilizando el Teorema de Menger (Teorema 1.2). Es decir, conectando todos los nodos de A_1 a una fuente s y todos los de A_2 a un sumidero t y hallando el mayor número L de caminos $s-t$ arista-disjuntos. Para ello se aplica el Algoritmo de Ford y Fulkerson con capacidades unitarias en todas las aristas.

Una vez que se obtiene un emparejamiento máximo M_L de tamaño L se puede construir un cubrimiento de vértices correspondiente de la siguiente forma:

El emparejamiento máximo M_L particiona las aristas de G en aquellas utilizadas en el emparejamiento E_M y aquellas que no E_U, luego $E = E_M \cup E_U$. Se define T el conjunto de vértices de A_1 que no son extremos de ninguna arista de E_M, además de todos los vértices que pueden ser alcanzados desde ellos utilizando un camino elemental que toma aristas de E_U y E_M alternadamente. Es decir, para cada vértice de A_1 que no pertenece a ninguna arista del emparejamiento, se agregan en T todos los vértices que aparecen en un camino alternante con aristas fuera y dentro del emparejamiento.
Luego, el conjunto $C = (A_1 \setminus T) \cup (A_2 \cap T)$ define un cubrimiento de vértices correspondiente de cardinal mínimo. La figura a continuación muestra un ejemplo de la construcción anterior:

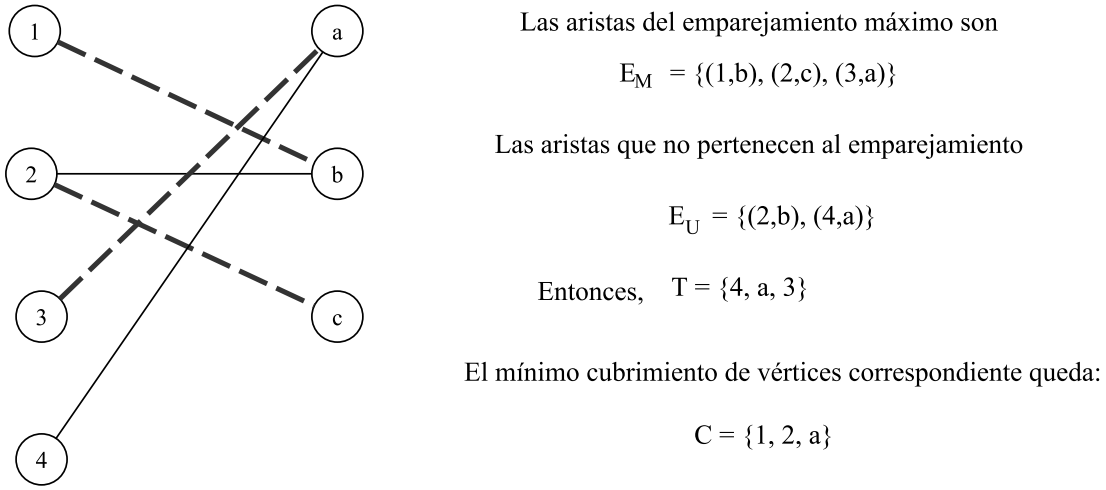

Fig. 4.7 Construcción del Cubrimiento de vértices de cardinal mínimo.

Si el presupuesto B es como mínimo L, entonces todos los nodos del cubrimiento de vértices podrán ser hallados en tiempo polinomial e inmunizados. □

Capítulo 5

Conclusiones Finales y Trabajo a Futuro

En esta tesis se estudió el problema de optimización combinatoria, denominado Problema de Fragmentación de Grafos (GFP). Inspirado por el modelado de epidemias su aplicación se extiende a otras clases de desastres, por ejemplo, catástrofes naturales. El Pro-blema de Fragmentación de Grafos se presentó desde el punto de vista de sus propiedades teóricas.

En el tratamiento del GFP el sistema que será afectado se modeló como una red, donde un nodo expuesto al desastre inmediatamente lo propaga a sus vecinos. El objetivo del GFP consiste en elegir una estrategia de inmunización de forma en que se minimice el número esperado de muertes causadas. Se presentó la relación del problema con el modelo de clásico para representación de epidemias SIR y con otro problema teórico de grafos denominado Component Order Connectivity problem (COC). Se probó que el problema de decisión asociado al GFP pertenece a la clase de problemas $\mathscr{NP}$-completos y también un resultado fuerte de inaproximabilidad que muestra que no existe un algoritmo aproximado de tiempo polinomial para la resolución del GFP con factor menor a $\frac{5}{3}$, a menos que $\mathscr{P} = \mathscr{NP}$.
Por último, en contraste con el anterior resultado de inaproximabilidad, fue posible hallar estrategias de tiempo polinomial para la mejor inmunización en algunas familias especiales de grafos, a saber, ciclos, grafos acíclicos y grafos bipartitos.

Como trabajo futuro se observan dos líneas de investigación importantes a seguir. La primera es mejorar heurísticas para la resolución del GFP, como ser Greedy y GRASP presentadas en [20]. La otra, más desafiante pero claramente el motivo más importante para esta investigación es la aplicación en escenarios de la vida real.

Bibliografía

[1] Andersson, H. (1997). Epidemics in a population with social structures. *Mathematical Biosciences*, 140(2):79 – 84.

[2] Aprile, M., Castro, N., Robledo, F., and Romero, P. G. (2017). Analysis of node-resilience strategies under natural disasters. In *International Conference on Design of Reliable Communication Networks 2017 (DRCN 2017)*, pages 93–100, Munich, Germany.

[3] Ball, F. and Sirl, D. (2013). Acquaintance vaccination in an epidemic on a random graph with specified degree distribution. *J. Appl. Probab.*, 50(4):1147–1168.

[4] Ball, F., Sirl, D., and Trapman, P. (2010). Analysis of a stochastic {SIR} epidemic on a random network incorporating household structure. *Mathematical Biosciences*, 224(2):53 – 73.

[5] Castro, N., Ferreira, G., Robledo, F., and Romero, P. (2017). Graph fragmentation problem for natural disaster management. In *Proceedings of the Third International Conference on Machine Learning, Optimization and Big Data. Springer LNCS*, Cham. Springer International Publishing.

[6] Cook, S. A. (1971). The complexity of theorem-proving procedures. In *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, STOC '71, pages 151–158, New York, NY, USA. ACM.

[7] Dahlhaus, E., Johnson, D. S., Papadimitriou, C. H., Seymour, P. D., and Yannakakis, M. (1992). The complexity of multiway cuts (extended abstract). In *Proceedings of the Twenty-fourth Annual ACM Symposium on Theory of Computing*, STOC '92, pages 241–251, New York, NY, USA. ACM.

[8] Drange, P. G., Dregi, M., and van 't Hof, P. (2016). On the computational complexity of vertex integrity and component order connectivity. *Algorithmica*, 76(4):1181–1202.

[9] Fine, P. E. (1977). A commentary on the mechanical analogue to the reed-frost epidemic model. *American journal of epidemiology*, 106(2):87–100.

[10] Fromont, E., Artois, M., and Pontier, D. (1996). Cat population structure and circulation of feline viruses. *Acta oecologica*, 17(6):609–620.

[11] Garey, M. R. and Johnson, D. S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA.

[12] Harary, F. (1969). *Graph Theory*. Addison-Wesley Series in Mathematics. Addison Wesley.

[13] Karp, R. M. (1972). Reducibility among combinatorial problems. In Miller, R. E. and Thatcher, J. W., editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press.

[14] Kleinberg, J. and Tardos, E. (2005). *Algorithm Design*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

[15] Nemhauser, G. (1966). *Introduction To Dynamic Programming*. John Wiley & Sons Inc., 5th edition.

[16] Newman, M. E. J. (2002). Spread of epidemic disease on networks. *Physical Review E*, 66(1):1–12.

[17] Newman, M. E. J. (2003). The structure and function of complex networks. *SIAM Review*, 45(2):167–256.

[18] Newman, M. E. J., Strogatz, S. H., and Watts, D. J. (2001). Random graphs with arbitrary degree distributions and their applications. *Physical Review E*, 64(2):1–17.

[19] Piccini, J. (2016). *Static Reliability and Resilience in Dynamic Systems*. PhD thesis, Facultad de Ingeniería, Universidad de la República , Uruguay.

[20] Piccini, J., Robledo, F., and Romero, P. (2015). *Node-Immunization Strategies in a Stochastic Epidemic Model*, pages 222–232. Springer International Publishing, Cham.

[21] Piccini, J., Robledo, F., and Romero, P. (2016a). Analysis and complexity of pandemics. In *2016 8th International Workshop on Resilient Networks Design and Modeling (RNDM)*, pages 224–230.

[22] Piccini, J., Robledo, F., and Romero, P. (2016b). Graph fragmentation problem. In *Proceedings of 5th the International Conference on Operations Research and Enterprise Systems - Volume 1: ICORES*, pages 137–144. INSTICC, SciTePress.

[23] Piccini, J., Robledo, F., and Romero, P. (2018). Complexity among combinatorial problems from epidemics. *International Transactions in Operational Research*, 25(1):295–318.

[24] Santhanam, G. R., Suvorov, Y., Basu, S., and Honavar, V. (2011). Verifying intervention policies to counter infection propagation over networks: A model checking approach. In *Proceedings of the National Conference on Artificial Intelligence*, pages 1408–1414.

[25] Vazirani, V. V. (2001). *Approximation Algorithms*. Springer-Verlag New York, Inc., New York, NY, USA.